VMware Aria Automation 8.18 API Programming Guide

July 2024

VMware Aria Automation 8.18

vmware by Broadcom

You can find the most up-to-date technical documentation on the VMware by Broadcom website at:

https://docs.vmware.com/

VMware by Broadcom
3401 Hillview Ave.
Palo Alto, CA 94304
www.vmware.com

Copyright © 2024 Broadcom. All Rights Reserved. The term “Broadcom” refers to Broadcom Inc. and/or its subsidiaries. For more information, go to https://www.broadcom.com. All trademarks, trade names, service marks, and logos referenced herein belong to their respective companies.

Contents

1 What are the VMware Aria Automation APIs and how do I use them 6

2 Getting Your Authentication Token 10

- Get Your Access Token 10
- Verify User Roles 12

3 Prerequisites for API Use Case Examples 15

4 Automation Assembler Tutorials 18

- Working with tags 18
  - Create a Kubernetes Zone with a Tag 24
- How do I retrieve provisioning request details 27
- How do I extract and edit zones associated with a project 31
- Deploying and Managing Resources 35
  - Create and Deploy a Machine Resource 35
  - Managing IP Addresses 40
    - Query for IP Addresses 40
    - Allocate IP Addresses 46
    - Release IP Addresses 50
  - Creating and Using a First Class Disk 53
    - Create a First Class Disk 53
    - Attach a First Class Disk 56
    - Manage First Class Disk Snapshots 60
  - Working with Azure Disk Snapshots 65
    - Create a Block Device 65
    - Create and Manage Azure Disk Snapshots 68
  - Update the Custom Properties of a Machine 73
  - Provision a VLAN Private Network 75
- How do I use a placement policy to spread VMs by memory 77
- Protecting Sensitive Data 82
  - How to provision a machine with sensitive data 83
  - Properties that Support Encryption 85
- Querying with the APIs 87
  - Using Pagination and Count 90
  - Filtering Resources by Region ID 91
  - Filtering for Machine Status 96
  - Filtering Operations for Projects 97

5 Setting up Automation Assembler using APIs 100

Adding Cloud Accounts 100

Add an Amazon Web Services Cloud Account 100

Add a vSphere Cloud Account 102

Add an NSX-T or NSX-V Cloud Account 109

Add a VMC Cloud Account with a Proxy 113

Add a Microsoft Azure Cloud Account 117

Add a Google Cloud Platform Cloud Account 119

Integrating with other applications 121

Create an Integration with Github 123

Delete an Integration 126

How do I import an IPAM package 129

6 Using Automation Assembler APIs to Build your Resource Infrastructure 136

Create a Cloud Zone 136

Create a Cloud Zone with a Folder 138

Create a Project to use in Automation Assembler 141

Add Users to Your Project 144

Add a Cloud Zone to Your Project 147

Create Flavor Mappings 150

Create Image Mappings 156

Working with Networks 159

Create Network Profiles 159

Create a Network Profile with Isolation 162

Create a Network Profile with Security Groups 168

Using the Network APIs 171

Creating Storage Profiles 171

Create an Amazon Web Services Storage Profile 171

Create a vSphere Storage Profile 175

Create a vSphere Storage Profile for a First Class Disk 180

Create a Microsoft Azure Storage Profile 183

Create a Microsoft Azure Storage Profile for a Managed Disk 187

7 Managing Your Projects 190

Create a Project with the Project Service API 190

Add Users to Your Project Using the Project Service API 192

8 Working with Blueprints/Cloud Templates 197

Create and Update a Cloud Template 197

Setting up Policies 201

Create an Approval Policy 201

How to Create Resource Quota Policies 206
Create a Deployment Limit Policy 211
Create a Content Sharing Policy 216
Version and Release a Cloud Template to a VMware Aria Automation Service Broker Catalog 223
Edit and Version a Custom Form 225
Remove a Cloud Template Version from a VMware Aria Automation Service Broker Catalog 232
Test Your Cloud Template Deployment 234
Deploy Your Cloud Template 236
Specify SCSI disk placement 240
How to Create Custom Naming Templates 244

9 Requesting a Deployment from a Catalog Item 248
Create a Catalog Source and List Discovered Items 248
Request Deployment 252
Create a Lease Policy 256

10 Working with Deployments and Resources 258
Deploy a Cloud Template with Contents Inline 258
Look up Deployment Details 263
Get Deployment Resource IDs 265
Change the Lease on Your Deployment 267
Delete Your Deployment 271
Reconfigure Load Balancer 275
Add a Disk to a Machine and Power It Off 280
Viewing Billable Objects 286
Onboarding machines 289
Onboard machines as a single deployment 289
How do I add a cloud template to my onboarding plan 294

11 Working with Pipelines 298
Create an Endpoint 298
Create and Enable a Pipeline 301
Run and Monitor your Pipeline 305

What are the VMware Aria Automation APIs and how do I use them

1

As a VMware Aria® Automation™ user or customer, you can perform VMware Aria Automation Assembler, VMware Aria Automation Service Broker, and VMware Aria Automation Pipelines functions programmatically by using REST API service calls.

API Services

VMware Aria Automation includes the following APIs. API documentation is available with the product. To access all Swagger specifications from a single landing page, go to `https://<your_FQDN>/automation/api-docs` where *your_FQDN* is the FQDN of your VMware Aria Automation appliance.

Table 1-1. VMware Aria Automation

Main Service	Service Name and Description
ABX	**ABX** Create or manage actions and their versions. Execute actions and flows.
Automation Service Broker	**Approvals** Enforce policies that control required approvals for a deployment or Day 2 action before the request is provisioned.
Automation Assembler	**Blueprint** Create, validate, and provision blueprints. **Note** Blueprints in the API are Automation Assembler Templates in the product.
Automation Service Broker	**Catalog** Access Automation Service Broker catalog items and catalog sources, including content sharing and the request of catalog items.

Table 1-1. VMware Aria Automation (continued)

Main Service	Service Name and Description
CMX	**CMX** When using Kubernetes with VMware Aria Automation , deploy and manage Kubernetes clusters and namespaces.
Automation Assembler	**Content Gateway** Connect to your infrastructure as code content in external content sources, such as Source Code Management providers.
Automation Service Broker and Automation Assembler	**Custom Forms** Define dynamic form rendering and customization behavior in Automation Service Broker and Automation Assembler.
User Profile	**Customization** Configure branding information.
Automation Service Broker	**Deployment** Access deployment objects and platforms or blueprints that have been deployed into the system. **Note** Blueprints in the API are Automation Assembler Templates in the product.
Automation Service Broker	**Deployment Metric** Aggregated metric values for the deployment objects.
Identity	**Identity** Authenticate and manage the authorization of VMware Aria Automation users.
Automation Assembler	**Infrastructure as a Service (IaaS)** Perform infrastructure setup tasks, including validation, and provisioning of resources in an iterative manner.
Migration	**Migration Assistant** Run assessments and access migration services. Supports migration for vRealize Automation 7.6 content, and for NSX-V to NSX-T migration.
Relocation	**Onboarding** Define policies and plans to bring existing VMs from any cloud under management.

Table 1-1. VMware Aria Automation (continued)

Main Service	Service Name and Description
Automation Orchestrator	**Orchestrator** Design, manage, and run workflows, actions, and policies to automate complex IT tasks.
Automation Orchestrator	**Orchestrator Gateway** Run workflows and actions to automate complex IT tasks.
Automation Pipelines	**Pipelines** Create and run pipelines for continuous delivery of your applications to production.
Automation Service Broker	**Policies** Interact with policies created in Automation Service Broker.
Project	**Projects** Provide visibility and isolation of provisioned resources for users with a project role. **Note** Swagger docs for the Platform and RBAC services are with the Projects Service.

Note VMware provides customers with a 12 month End-of-Life notice for any breaking changes to public APIs. Any breaking changes are announced in the release notes included with the VMware Aria Automation Documentation.

API versioning

It is highly recommended but not necessary to use API versioning. API versioning allows you to lock the API to a value and control when you upgrade to a new API version. If you do not use API versioning, the default behavior varies depending upon the API.

- For the IaaS APIs, the latest version is 2021-07-15. If you consume the IaaS APIs without versioning or if you assign a value other than 2021-07-15, requests use the version 2019-01-15.

  As a best practice, lock your IaaS API requests with the *apiVersion* query parameter assigned to 2021-07-15 so that you ensure a smooth transition to the latest version before the version 2019-01-15 reaches its end of life. See Chapter 6 Using Automation Assembler APIs to Build your Resource Infrastructure.

- For other APIs, you can specify any date you choose for the *apiVersion* query parameter. If you leave the value unspecified, requests use the latest API version by default. However backward compatibility is not preserved and if the API changes, you might encounter an unexpected change in the API response.

  As a best practice, use the *apiVersion* query parameter in your API requests and lock your API to the latest version listed in the Swagger specification. Then if a new API version is announced, you control when to opt-in to that version by changing the *apiVersion* query parameter to the new version value.

The following example shows how to use the *apiVersion* query parameter for the catalog API. The catalog API versions are: 2020-08-25, 2020-01-30, and 2019-01-15. Including the additional *apiVersion* query parameter locks the call to the API version that was in effect as of January 30, 2020 and through August 24, 2020.

```
GET https://appliance.domain.com/catalog/api/sources?apiVersion=2020-01-30
```

When you are ready to opt-in to the features released with the version dated 2020-08-25, change the value of the *apiVersion* query parameter.

```
GET https://appliance.domain.com/catalog/api/sources?apiVersion=2020-08-25
```

Setting the *apiVersion* query parameter to the latest version ensures that you will also get updates to the catalog API that occur after 2020-08-25. However, no breaking changes will occur until a new version is announced and you will only experience those changes if you change the value of the *apiVersion* query parameter to a date that is equivalent to the new version or later.

Note API versions do not change for every VMware Aria Automation release and are not the same for all services. To check API versions for the services you use, go to `https://<FQDN>/automation-ui/api-docs` and click the cards to open the Swagger specifications.

How Developers Use the VMware Aria Automation APIs

To make API service calls, you use a browser application or an HTTP client application to send requests and review responses. The following open-source applications are commonly used:

- cURL. http://curl.haxx.se
- `jq` parser. https://stedolan.github.io/jq/
- Postman application. https://www.getpostman.com/

To learn how to use the API, you start by getting an authentication token. Then you can perform steps outlined in the use cases in this guide. The use cases include `curl` commands in request examples. To use the commands, ensure that the `jq` command-line JSON processor is installed with `curl`. The `jq` parser ensures that responses are formatted for optimum readability. For information about `jq` installation, see https://stedolan.github.io/jq/.

Getting Your Authentication Token

2

In the REST API, VMware Aria Automation requires an HTTP authentication token in the request header. Getting your authentication token is a prerequisite for any use case.

The access token is the token you use to authenticate all API calls. To acquire an access token, you use the Identity Service and Infrastructure as a Service (IaaS) APIs. After you get the access token, use it to verify user roles.

Read the following topics next:

- Get Your Access Token for the VMware Aria Automation API
- Verify User Roles

Get Your Access Token for the VMware Aria Automation API

To get the token used to authenticate your session, you use the Identity Service API to get an API token. Then you use the API token as input to the IaaS API to get an access token.

The access token is valid for eight hours. If the token times out, request it again. The following procedure shows how to obtain the access token using both the Identity Service API and the IaaS API.

Prerequisites

- Secure a channel between the web browser and the VMware Aria Automation server. Open a browser and enter the URL such as: `https://appliance.domain.com`.

Procedure

1 Assign values to the variables for the hostname of your VMware Aria Automation appliance, your user name, and your password.

```
url='https://<your_FQDN>'
username='<your_username>'
password='<your_password>'
```

2 Use the Identity Service API to obtain the API token.

The API token is also known as the refresh token. It is valid for 90 days and can be used to generate a new access token when the access token expires.

Note You cannot revoke the refresh token.

```
api_token=`curl -X POST \
  "$url/csp/gateway/am/api/login?access_token" \
  -H 'Content-Type: application/json' \
  -H 'Accept: application/json' \
  -d '{
  "username": "'$username'",
  "password": "'$password'"
}' | jq -r .refresh_token`
```

3 Verify the API token variable is assigned.

The token is a compact string of characters as in the following example.

```
# echo $api_token
ABCutJJ7oEq7sWYD9qkpnlrzYqlFlSZmrWXYZrkpGswN8nzjmkWeNqn8QA1RfhQg
```

4 With the API token assigned, use the IaaS API to request the access token.

```
access_token=`curl -X POST \
  "$url/iaas/api/login" \
  -H 'Content-Type: application/json' \
  -H 'Accept: application/json' \
  -d '{
  "refreshToken": "'$api_token'"
}' | jq -r .token`
```

5 Verify the access token variable is assigned.

The access token is a long JSON Web Token as in the following example.

```
# echo $access_token
eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCIsImtpZCI6InNpZ25pbmdfMiJ9.eyJzdWIiOiJ2bXdhcmUuY29tOj...
...
tSQ74_XhszGifZe_gFdxw
```

After 25 minutes of inactivity, the access token times out and you must request it again. You can revoke an access token at any time.

Results

You have obtained the access token required to authenticate your API calls. This access token is valid for VMware Aria Automation users and is necessary when using tools that are integrated with VMware Aria Automation.

What to do next

Use the access token to verify user roles. See Verify User Roles .

Verify User Roles

To use the API, a VMware Aria Automation user must be an organization member with at least a user service role. You use the access token to verify user roles.

Prerequisites

Verify that you have an access token. See Get Your Access Token for the VMware Aria Automation API.

Procedure

1 Assign values to the variables for the hostname of your VMware Aria Automation instance, your user name, and your password.

```
url='http://<FQDN>'
username='<your_username>'
password='<your_password>'
```

2 Get your organization ID.

```
curl -X GET \
  "$url/csp/gateway/am/api/loggedin/user/orgs" \
  -H "csp-auth-token: $access_token"
```

3 Examine the response and assign the organization ID variable.

```
org_id='<your_org_id>'
```

4 Get your organization role.

```
curl -X GET \
   $url/csp/gateway/am/api/loggedin/user/orgs/$org_id/roles \
   -H "csp-auth-token: $access_token" | jq "."
```

The name field displays the organization role and must be either `org_owner` or `org_member`.

5 Get your service role.

```
curl -X GET \
   $url/csp/gateway/am/api/loggedin/user/orgs/$org_id/service-roles \
   -H "csp-auth-token: $access_token" | jq "."
```

The serviceRolesNames field displays the service role for each service and must be at least `user`.

Example: Verify User Roles

Using the access token previously obtained and assigned, verify user roles. See Get Your Access Token for the VMware Aria Automation API.

Assign variables.

```
# url='https://appliance.company.com'
# username='user@example.local'
# password='example_password'
```

Get your organization ID.

```
# curl -X GET \
  "$url/csp/gateway/am/api/loggedin/user/orgs" \
  -H "csp-auth-token: $access_token"
```

The response shows the organization ID.

```
{
  "refLinks": [
    "/csp/gateway/am/api/orgs/7f8c518a-65f5-494b-b714-f7e349957a30"
  ],
  "items": [
    {
      "name": "DEFAULT-ORG",
      "displayName": "DEFAULT-ORG",
      "refLink": "/csp/gateway/am/api/orgs/7f8c518a-65f5-494b-b714-f7e349957a30",
      "id": "7f8c518a-65f5-494b-b714-f7e349957a30",
      "metadata": null,
      "parentRefLink": null
    }
  ]
}
```

Assign the organization ID variable.

```
# org_id='7f8c518a-65f5-494b-b714-f7e349957a30'
```

Verify the organization role.

```
# curl -X GET \
   $url/csp/gateway/am/api/loggedin/user/orgs/$org_id/roles \
   -H "csp-auth-token: $access_token" | jq "."
```

The response shows that the organization role is `org_owner`.

```
{
    "refLink": "/csp/gateway/am/api/orgs/7f8c518a-65f5-494b-b714-f7e349957a30/roles/
52a6a411-2339-4bc3-91bc-62418977df11",
    "name": "org_owner",
    "displayName": "Organization Owner",
    "organizationLink": "/csp/gateway/am/api/orgs/7f8c518a-65f5-494b-b714-f7e349957a30"
  }
```

Verify the service role.

```
# curl -X GET \
   $url/csp/gateway/am/api/loggedin/user/orgs/$org_id/service-roles \
   -H "csp-auth-token: $access_token" | jq "."
```

A snippet of the response shows the Service Role Names for the Automation Assembler service. `cloud_admin` satisfies the minimum service role.

```
...
 {
      "serviceDefinitionLink": "/csp/gateway/slc/api/definitions/external/<service_id>",
      "serviceRoleNames": [
        "automationservice:cloud_admin"
      ]
    }
...
```

Prerequisites for API Use Case Examples

3

Before performing any task using API services, you must review the prerequisites. General prerequisites apply to all services. Prerequisites that are service-specific are common to all endpoints for the service, but may vary depending upon the service role for example, administrator or user.

See Organization and service user roles in VMware Aria Automation.

General prerequisites for all services

Before performing any task for any service, the following prerequisites must be satisfied:

- Verify that you have an active access token. See Chapter 2 Getting Your Authentication Token.
- Verify that the URL variable is assigned.

```
url='https://appliance.domain.com'
```

Prerequisites specific to API services

The following table lists prerequisites that are specific to the services with use cases in this guide.

Note Every service includes an API version variable. If you choose not to assign a value to the *apiVersion* or to assign it to a different value, review the information in API versioning.

Table 3-1. Prerequisites for VMware Aria Automation Service Use Cases

Product: Service	Service prerequisites	Role-specific prerequisites
Automation Assembler: Blueprint service To access the Swagger documentation for the Blueprint API, see `https://<your_FQDN>/blueprint/api/swagger/ui/`.	Verify that the Blueprint API version variable is assigned as in the following example. `api_version='2019-09-12'`	Verify that you are at least an organization member in VMware Aria Automation with a Automation Assembler administrator service role.
Automation Service Broker: Catalog service To access the Swagger documentation for the Catalog API, see `https://<your_FQDN>/deployment/api/swagger/swagger-ui.html?urls.primaryName=catalog`.	Verify that the Catalog API version variable is assigned as in the following example. `api_version='2020-08-25'`	Verify that you are at least an organization member in VMware Aria Automation with a Automation Service Broker administrator service role.
Automation Service Broker: Deployment service To access the Swagger documentation for the Deployment API, see `https://<your_FQDN>/deployment/api/swagger/swagger-ui.html?urls.primaryName=deployments`. **Note** For the Automation Assembler: Deployment service, use the prerequisites for the Automation Assembler: Infrastructure as a Service (IaaS) service.	Verify that the Deployment API version variable is assigned as in the following example. `api_version='2020-08-25'`	Verify that you are at least an organization member in VMware Aria Automation with a Automation Service Broker administrator service role.
Automation Assembler: Infrastructure as a Service (IaaS) service To access the Swagger documentation for the IaaS API, see `https://<your_FQDN>/iaas/api/swagger/ui/`.	Verify that the IaaS API version variable is assigned as in the following example. `api_version='2021-07-15'` If you do not assign a value to the *apiVersion* variable or you assign it to a value other than 2021-07-15, the IaaS API behavior will default to value of the previous version or 2019-01-15.	■ Verify that you are an organization owner in VMware Aria Automation with a Automation Assembler administrator service role. ■ If working with cloud accounts, verify that you have cloud administrator credentials. See Credentials required for working with cloud accounts in VMware Aria Automation.
All products: Onboarding service To access the Swagger documentation for the Onboarding API, see `https://<your_FQDN>/relocation/api/swagger/swagger-ui.html`	No version variable needed	Verify that you are at least an organization member in VMware Aria Automation with an Automation Assembler administrator service role.

Table 3-1. Prerequisites for VMware Aria Automation Service Use Cases (continued)

Product: Service	Service prerequisites	Role-specific prerequisites
Automation Pipelines: Pipelines service To access the Swagger documentation for the Pipelines API, see `https://<your_FQDN>/pipeline/api/swagger/swagger-ui.html`.	Verify that the Pipelines API version variable is assigned as in the following example. `api_version='2019-10-17'`	Verify that you are at least an organization member in VMware Aria Automation with an Automation Pipelines administrator service role.
Automation Service Broker: Policies service To access the Swagger documentation for the Policies API, see `https://<your_FQDN>/deployment/api/swagger/swagger-ui.html?urls.primaryName=policies`.	Verify that the Policies API version is assigned as in the following example. `api_version='2020-08-25'`	Verify that you are at least an organization member in VMware Aria Automation with a Automation Service Broker administrator service role.
All products: Project service To access the Swagger documentation for the Projects API, see `https://<your_FQDN>/project/api/swagger/swagger-ui.html`.	Verify that the Projects API version variable is assigned as in the following example. `api_version='2019-01-15'`	Verify that you are at least an organization member in VMware Aria Automation with an administrator service role.

Automation Assembler Tutorials

4

As a Automation Assembler administrator, you can use these tutorials to learn how to perform tasks programmatically using the IaaS APIs.

In addition to the steps in the tutorials, there is additional information in the guide. Links are provided to relevant topics.

Read the following topics next:

- Working with tags
- How do I retrieve provisioning request details
- How do I list and edit zones associated with a project
- Deploying and Managing Resources
- How do I use a placement policy to spread VMs by memory
- Protecting Sensitive Data
- Querying with the Automation APIs

Working with tags

As an Organization administrator with the Assembler admin service role, you can use the Tags endpoints to create, manage, or delete tags in Automation Assembler.

For information about working with tags using the UI, see How do I use tags to manage Automation Assembler resources and deployments.

Note

- Tag filters are case-insensitive.
- There is no way to query for unused tags using the IaaS API. However, you can submit a request to check for tag usage of specific tag IDs.

Prerequisites for working with tags

- Verify that all general prerequisites and prerequisites for the Automation Assembler Infrastructure as a Service (IaaS) service have been satisfied. See Chapter 3 Prerequisites for API Use Case Examples.

Filtering for tags

You can filter for tags based on a key, value, or origin and the tag filters are case-insensitive.

The following example shows how to filter for a tag with **`location`** in the key name.

```
curl -X GET \
  "$url/iaas/api/tags?$filter=key%20eq%20'location'&apiVersion=$api_version" \
  -H 'Content-Type: application/json' \
  -H "Authorization: Bearer $access_token" | jq "."
```

A sample response shows `"key": "location"`.

```
{
    "content": [
        {
            "key": "location",
            "value": "MUM",
            "id": "9aabbe23-d4bf-3d55-89f7-dc7e67317442"
        }
    ],
    "totalElements": 1,
    "numberOfElements": 1
}
```

To filter for the origin of a tag, you specify **`USER_DEFINED`** or **`DISCOVERED`** for the origin value. The following example filters for tags with the **`DISCOVERED`** origin.

```
curl -X GET \
  "$url/iaas/api/tags?$filter=origins.item%20eq%20%27DISCOVERED%27&apiVersion=$api_version" \
  -H 'Content-Type: application/json' \
  -H "Authorization: Bearer $access_token" | jq "."
```

Tag origins can also be both `USER_DEFINED` and `DISCOVERED` as in the following cases:

- If a tag's origin is `USER_DEFINED` and another tag with the same case-sensitive key/value pair is assigned to an endpoint resource from which Automation Assembler collects data, Automation Assembler adds the `DISCOVERED` origin.
- If a user assigns a tag with the `DISCOVERED` origin to any Automation Assembler resource such as a network profile, Automation Assembler also assigns the `USER_DEFINED` origin to the tag.

Creating tags

You create a tag with a name and an optional value. The following example shows how to create a tag named **`environment`** with the value **`prod`** that you could add to resources in your production environment.

```
curl -X POST \
  "$url/iaas/api/tags?apiVersion=$api_version"
  -H 'Content-Type: application/json'
  -H "Authorization: Bearer $access_token"
  -d '{
    "key" : "environment",
    "value" : "prod"
  }' | jq "."
```

The response provides the tag ID.

```
 {
    "key": "environment",
    "value": "prod",
    "id": "ca0ed5ef-72b5-3d9f-bc6d-0707defff540"
}
```

Listing and deleting tags

To delete a tag, you specify the tag ID. To get the tag ID, you list all tags in your infrastructure.

```
curl -X GET \
  "$url/iaas/api/tags?apiVersion=$api_version"
  -H 'Content-Type: application/json'
  -H "Authorization: Bearer $access_token"
  }' | jq "."
```

The response lists all the tag names, values, and IDs as in the following example.

```
{
    "content": [
        {
            "key": "location",
            "value": "MUM",
            "id": "9aabbe23-d4bf-3d55-89f7-dc7e67317442"
        }
        {
            "key": "Application",
            "value": "testing",
            "id": "57ead12-523a-3cd2-9447-06e757dcf382"
        }
    ],
    "totalElements": 2,
    "numberOfElements": 2
}
```

Before deleting a tag, a best practice is to check the tag usage of the ID that you plan to delete. The following example checks for usage of the tag with ID **9aabbe23-d4bf-3d55-89f7-dc7e67317442**.

```
curl -X POST \
  "$url/iaas/api/tags/tags-usage?apiVersion=$api_version" \
  -H 'Content-Type: application/json' \
  -H "Authorization: Bearer $access_token" \
  -d '{
    "tagIds" : ["9aabbe23-d4bf-3d55-89f7-dc7e67317442"]
  }' | jq "."
```

- If the tag is being used, the response shows where it is being used. This sample response shows that the tag is being used in a network profile.

```
{
    "documentLinks": [
        "/provisioning/resources/network-profiles/8615bbfe-ea90-420a-a1cf-8048ba4271d4"
    ],
    "documents": {
        "/provisioning/resources/network-profiles/8615bbfe-ea90-420a-a1cf-8048ba4271d4": {
            "id": "a30929ec-3b06-484e-8785-cc43619b5b01",
            "name": "23360",
            "customProperties": {
                "edgeClusterRouterStateLink": "/resources/routers/2d57f1c4-fbe0-46ce-
a061-3a5f1396f37b",
                "tier0LogicalRouterStateLink": "/resources/routers/9c965abe-9d69-40c0-99e0-
be5aa7e52097",
                "onDemandNetworkIPAssignmentType": "mixed"
            },
            "tagLinks": [
                "/resources/tags/ef254637-594d-3960-a642-f157f859a830",
                "/resources/tags/9aabbe23-d4bf-3d55-89f7-dc7e67317442"
            ],
            "expandedTags": [
                {
                    "tag": "23360\n"
                },
                {
                    "tag": "location\nMUM"
                }
            ],
            "documentVersion": 11,
            "documentKind":
"com:vmware:admiral:compute:profile:NetworkProfileService:NetworkProfile",
            "documentSelfLink": "/provisioning/resources/network-profiles/8615bbfe-
ea90-420a-a1cf-8048ba4271d4",
            "documentUpdateTimeMicros": 1696947968305000,
            "documentExpirationTimeMicros": 0
        }
    },
```

```
    "documentCount": 1,
    "documentVersion": 0,
    "documentUpdateTimeMicros": 0,
    "documentExpirationTimeMicros": 0}
```

- If the tag is not being used, the response shows no resource links.

```
{
    "documentLinks": [],
    "documents": {},
    "documentCount": 0,
    "documentVersion": 0,
    "documentUpdateTimeMicros": 0,
    "documentExpirationTimeMicros": 0
}
```

To delete the unused tag, specify the tag ID. The delete action first checks to ensure that the tag is not associated with a resource before deleting it. The following example deletes the tag with ID **9aabbe23-d4bf-3d55-89f7-dc7e67317442**.

```
curl -X DELETE \
  "$url/iaas/api/tags/9aabbe23-d4bf-3d55-89f7-dc7e67317442?apiVersion=$api_version" \
  -H 'Content-Type: application/json' \
  -H "Authorization: Bearer $access_token"  \
  }' | jq "."
```

Note To delete a tag that is associated with a resource, you can use the **ignoreUsage** query parameter as in the following example that forces the deletion of a tag with ID **9aabbe23-d4bf-3d55-89f7-dc7e67317442**. However, this removes all tag assignments from all the resources associated with the tag.

```
curl -X DELETE \
  "$url/iaas/api/tags/9aabbe23-d4bf-3d55-89f7-dc7e67317442?
ignoreUsage=true&apiVersion=$api_version" \
  -H 'Content-Type: application/json' \
  -H "Authorization: Bearer $access_token"  | jq "."
```

Updating tags

You can use the IaaS API to add, remove, or update the tags associated with a resource. Using the ID of the resource, you submit a request get the details about the resource including the tag definitions. Then you submit a request to update the resource with new tag key/value pairs.

The following example shows how to update tags for a vSphere datastore

List all vSphere datastores.

```
curl -X GET \
  "$url/iaas/api/fabric-vsphere-datastores?apiVersion=$api_version"
```

```
  -H 'Content-Type: application/json' \
  -H "Authorization: Bearer $access_token" | jq "."
```

Examine the response to find the ID of the datastore that you want to use and use the ID to get details about the datastore. The following request uses the datastore ID **002e0a62-846d-4fb2-a153-3dcd80e57ba9**.

```
curl -X GET \
  "$url/iaas/api/fabric-vsphere-datastores/002e0a62-846d-4fb2-a153-3dcd80e57ba9/?
apiVersion=$api_version"
  -H 'Content-Type: application/json' \
  -H "Authorization: Bearer $access_token" | jq "."
```

A snippet of the response lists the tags on the datastore.

```
...
    "tags": [
        {
            "key": "Team",
            "value": "E2E"
        },
        {
            "key": "Owner/Requestor",
            "value": "User1"
        },
        {
            "key": "Product",
            "value": "vRA"
        }
    ],
...
```

Update the tags for the datastore. The following example payload:

- Removes the **Team** and **Owner/Requestor** keys.
- Adds the **environment** key.
- Updates the **Product** key.

```
curl -X PATCH \
  "$url/iaas/api/fabric-vsphere-datastore/002e0a62-846d-4fb2-a153-3dcd80e57ba9?
apiVersion=$api_version" \
  -H 'Content-Type: application/json' \
  -H "Authorization: Bearer $access_token" \
  -d {
    "tags": [
        {
            "key": "environment",
            "value": "dev"
        },
        {
            "key": "Product",
```

```
            "value": "saltstack"
        }
    ],
}' | jq "."
```

Create a Kubernetes Zone with a Tag

As a Automation Assembler admin, you can use APIs to create a new tag and add it to a Kubernetes zone.

In the following procedure, you use the IaaS API to create a new tag. Then using the CMX API, you create a Kubernetes zone with a supervisor namespace on vSphere and assign the tag to the zone.

Note To create a Kubernetes zone with a vSphere cloud account, the account must be Tanzu-enabled. When you open a Tanzu-enabled vSphere cloud account in the UI, the cloud account appears with the label **Available for Kubernetes deployment**.

For information on Tanzu-enabled vSphere integration, see Use Tanzu supervisor clusters and namespaces in Automation Assembler

Prerequisites

- Verify that all general prerequisites and prerequisites for the Automation Assembler Infrastructure as a Service (IaaS) service have been satisfied. See Chapter 3 Prerequisites for API Use Case Examples.
- Verify that you have the cloud account ID for a vSphere cloud account. See Add a vSphere Cloud Account .

Procedure

1 Create a new tag with a key/value pair.

```
curl -X POST \
  $url/iaas/api/tags?apiVersion=$api_version \
  -H 'Content-Type: application/json' \
  -H "Authorization: Bearer $access_token" \
  -d '{
    "key": "'<your_tag_key>'",
    "value": "'<your_tag_value>'"
  }' | jq "."
```

The response includes a tag ID.

2 Assign the tag ID variable.

```
tag_id='<example-tagID-alphanumeric-string>'
```

3 Assign your vSphere cloud account ID to the cloud account ID variable.

```
cloud_account_id='<vsphere_cloud_account_ID>'
```

4 Use the cloud account ID to create the Kubernetes zone with the tag.

To associate the Kubernetes zone with a vSphere cloud account you specify **"providerType": "VSPHERE_NAMESPACES"**.

```
curl -X POST \
  "$url/cmx/api/resources/k8s-zones" \
  -H 'Content-Type: application/json' \
  -H "Authorization: Bearer $access_token" \
  -d '{
    "name": "<your_k8s_zone_name>",
    "providerId": "'$cloud_account_id'",
    "providerType": "VSPHERE_NAMESPACES",
    "tagIds": [
        "'$tag_id'"
    ]
}' | jq "."
```

Example: Create a Kubernetes Zone with a Tag

Create a new tag with key **example_tag_key** and value **example_tag_value**. Create a Kubernetes zone named **K8s-test-zone** with the tag. To create a Kubernetes zone, you must associate it with a cloud account configured for Automation Assembler such as a vSphere cloud account with ID `8d4646bd-d629-4526-9009-10e3d4b66e44`.

Assign variables.

```
url='https://appliance.domain.com'
$ api_version='2021-07-15'
```

Create a new tag with the key/value.

```
$ curl -X POST \
  "$url/iaas/api/tags?apiVersion=$api_version" \
  -H 'Content-Type: application/json' \
  -H "Authorization: Bearer $access_token" \
  -d '{
    "key": "example_tag_key",
    "value": "example_tag_value"
  }' | jq "."
```

Examine the response for the tag ID.

```
{
    "key": "example_tag_key",
    "value": "example_tag_value",
    "id": "a6f324e4-101c-33f6-ac51-d9aaaa020123"
}
```

Assign a value to the tag ID variable.

```
$ tag_id1= 'a6f324e4-101c-33f6-ac51-d9aaaa020123'
```

Assign your vSphere cloud account ID to the cloud account ID variable.

```
$ cloud_account_id='8d4646bd-d629-4526-9009-10e3d4b66e44'
```

Create the Kubernetes zone with the tag and the cloud account ID.

```
$ curl -X POST \
  "$url/cmx/api/resources/k8s-zones" \
  -H 'Content-Type: application/json' \
  -H "Authorization: Bearer $access_token" \
  -d '{
    "name": "k8s-test-zone",
    "providerId": "'$cloud_account_id'",
    "providerType": "VSPHERE_NAMESPACES",
    "tagIds": [
        "'$tag_id1'"
    ]
}' | jq "."
```

The response shows the Kubernetes zone.

```
{
    "id": "220a81ba-c8ca-4f01-97bc-6f497576d564",
    "createdMillis": 1674847375758,
    "updatedMillis": 1674847375758,
    "orgId": "20451685-2bf6-4ba4-9ea4-87ee61bd32f8",
    "name": "k8s-test-zone",
    "providerId": "8d4646bd-d629-4526-9009-10e3d4b66e44",
    "providerType": "VSPHERE_NAMESPACES",
    "tagIds": [
        "a6f324e4-101c-33f6-ac51-d9aaaa020123"
    ]
}
```

If you want to update the Kubernetes zone, assign the Kubernetes zone ID variable.

```
$ k8s_zone_id='220a81ba-c8ca-4f01-97bc-6f497576d564'
```

If you want to add another tag, assign another tag ID variable.

```
$ tag_id2='21ccbdd6-c849-37f3-a784-1f2452cf802d'
```

Update the Kubernetes zone to add the tag.

Note When updating the zone, the update deletes any tags that are not included in the request payload. So when making the request, you must include all the tags that will remain assigned to the zone after the update.

```
$ curl -X PUT \
  "$url/cmx/api/resources/k8s-zones/$k8s_zone_id" \
  -H 'Content-Type: application/json' \
  -H "Authorization: Bearer $access_token" \
```

```
  -d '{
    "id": "220a81ba-c8ca-4f01-97bc-6f497576d564",
    "createdMillis": 1674847375758,
    "updatedMillis": 1674847375758,
    "orgId": "20451685-2bf6-4ba4-9ea4-87ee61bd32f8",
    "name": "k8s-test-zone",
    "providerId": "'$cloud_account_id'",
    "providerType": "VSPHERE_NAMESPACES",
    "tagIds": [
        "'$tag_id1'",
        "'$tag_id2'"
    ]
}' | jq "."
```

How do I retrieve provisioning request details

To validate placement scenarios before deploying a cloud template, you use the IaaS API to retrieve the request graph for a simulated provisioning request.

Prerequisites for retrieving provisioning request details

- Verify that all general prerequisites and prerequisites for the Automation Assembler Infrastructure as a Service (IaaS) service have been satisfied. See Chapter 3 Prerequisites for API Use Case Examples.
- Assign an API version variable for the Blueprint API.

  ```
  api_version_blueprint='2019-09-12'
  ```

  Note The Automation Assembler Infrastructure as a Service (IaaS) service and the Automation Assembler Blueprint service have different API version values. You set the API version value for the Automation Assembler Infrastructure as a Service (IaaS) service when you satisfied the general prerequisites.

- Verify that you have the ID for the cloud template that you plan to deploy. See Create and Update a Cloud Template. You can also get the cloud template ID from the UI, by opening the cloud template listed on **Design > Templates** and inspecting the ID at the end of the URL.

Retrieving provisioning request details

This example shows how to get request details for a cloud template with the ID `c8197446-d636-4ed9-aa2b-796da98ad10c`. In the following procedure, you use the blueprint API to get the deployment ID and flow ID and provide those values as input for the IaaS API request graph endpoint.

Use the template ID to get the deployment ID and flow ID.

```
curl -X POST \
$url/blueprint/api/blueprint-requests?apiVersion=$api_version_blueprint \
-H 'Accept: application/json' \
```

```
-H "Authorization: Bearer $access_token" \
-d '{
    "deploymentId": null,
    "deploymentName": null,
    "description": null,
    "plan": false,
    "blueprintId": "c8197446-d636-4ed9-aa2b-796da98ad10c",
    "content": null,
    "simulate": true
} | jq "."
```

Examine the response to get the deployment ID and the flow ID.

```
{
    "id": "8b918325-21a7-4902-b464-f39f5fb1bb64",
    "createdAt": "2023-11-22T07:12:54.585430Z",
    "createdBy": "user@mycompany.com",
    "updatedAt": "2023-11-22T07:12:54.585430Z",
    "updatedBy": "user@mycompany.com",
    "orgId": "434f6917-4e34-4537-b6c0-3bf3638a71bc",
    "projectId": "267f8448-d26f-4b65-b310-9212adb3c455",
    "projectName": "testing",
    "deploymentId": "924fcbc3-2076-48c9-a5f5-64de7f1cbe3a",
    "requestTrackerId": "8b918325-21a7-4902-b464-f39f5fb1bb64",
    "deploymentName": "deployment_924fcbc3-2076-48c9-a5f5-64de7f1cbe3a",
    "reason": "Simulate",
    "plan": false,
    "destroy": false,
    "ignoreDeleteFailures": false,
    "simulate": true,
    "blueprintId": "c8197446-d636-4ed9-aa2b-796da98ad10c",
    "inputs": {},
    "status": "STARTED",
    "flowId": "8b918325-21a7-4902-b464-f39f5fb1bb64"
}
```

Use the deployment ID and flow ID to get the provisioning request details.

```
curl -X GET \
$url/iaas/api/request-graph?apiVersion=$api_version \
-H 'Accept: application/json' \
-H "Authorization: Bearer $access_token" \
-d '{
    "deploymentId": "924fcbc3-2076-48c9-a5f5-64de7f1cbe3a",
    "flowId": "8b918325-21a7-4902-b464-f39f5fb1bb64",
} | jq "."
```

Using the response to validate placement scenarios

The response provides task and project information for the simulated provisioning request.

The tasks section of the response lists the task ID and stages. To validate a placement scenario, the request must successfully transition through all stages and substages. Any stage that shows an error includes a failure message that you can use to troubleshoot the problem with the placement scenario.

In the following sample response, a snippet shows an error during the reserving stage. To validate the provisioning request, correct errors and rerun the API until all tasks finish successfully.


```
    "tasks": [
        {
            "id": "46414006-8f71-46f8-b965-eea0bacbba7e",
            "stages": [
                {
                    "transitionSource": {
                        "timestampMicros": 0
                    },
                    "taskSubStage": "CREATED",
                    "taskInfo": {
                        "stage": "STARTED",
                        "isDirect": false
                    },
                    "resourceLinks": [
                        "/iaas/api/machines/4323fd90-ad4e-4342-b9ca-62a9bb7193c9"
                    ],
                    "timestampMicros": 1702877437358001
                },
                ...
                    },
                    "taskSubStage": "ENHANCED",
                    "taskInfo": {
                        "stage": "STARTED",
                        "isDirect": false
                    },
                ...
                    },
                    "taskSubStage": "RESOURCE_COUNTED",
                    "taskInfo": {
                        "stage": "STARTED",
                        "isDirect": false
                    },
                ...
                    },
                    "taskSubStage": "RESERVING",
                    "taskInfo": {
                        "stage": "STARTED",
                        "isDirect": false
                    },
                ...
                    "taskSubStage": "ERROR",
                    "taskInfo": {
                        "stage": "STARTED",
                        "isDirect": false,
                        "failure": {
```

```
                            "message": "Cannot find matching image mappings for image: im1",
                            "statusCode": 0,
                            "errorCode": 0,
                            "serverErrorId": "ad4d91ea-4319-4f11-abcf-3601c08f3ecd",
                            "documentKind": "com:vmware:xenon:common:ServiceErrorResponse"
                        }
                    },
                ...
             }
        ]
```

The project section of the response provides reference information for the project associated with the template.

```
    "project": {
        "administrators": [],
        "members": [],
        "viewers": [],
        "supervisors": [],
        "zones": [
            {
                "zoneId": "8830680d-71d1-4dcb-8234-f22de4254c6b",
                "priority": 0,
                "maxNumberInstances": 0,
                "allocatedInstancesCount": 0,
                "memoryLimitMB": 0,
                "allocatedMemoryMB": 0,
                "cpuLimit": 0,
                "allocatedCpu": 0,
                "storageLimitGB": 0,
                "allocatedStorageGB": 0.0,
                "id": "8a342007-8e8a-432b-9a12-14ebb51e3c1a-8830680d-71d1-4dcb-8234-
f22de4254c6b"
            }
        ],
        "constraints": {},
        "operationTimeout": 0,
        "sharedResources": true,
        "placementPolicy": "DEFAULT",
        "customProperties": {},
        "name": "p",
        "description": "",
        "id": "8a342007-8e8a-432b-9a12-14ebb51e3c1a",
        "orgId": "ce1fb992-7495-48fd-8988-412658094f6b",
        "_links": {
            "self": {
                "href": "/iaas/api/request-graph/8a342007-8e8a-432b-9a12-14ebb51e3c1a"
            }
        }
    }
```

How do I list and edit zones associated with a project

To query and edit zones associated with a project, you use Project endpoints in the IaaS API.

Prerequisites for extracting zones associated with a project

- Verify that all general prerequisites and prerequisites for the Automation Assembler Infrastructure as a Service (IaaS) service have been satisfied. See Chapter 3 Prerequisites for API Use Case Examples.
- Verify that you have your project ID. If you do not have the ID, list all projects to find the name and ID of the project with the associated zones that you want to query or edit.

  ```
  curl -X GET -H 'Accept: application/json' -H "Authorization: Bearer $access_token" "$url/
  iaas/api/projects?apiVersion=$api_version" | jq "."
  ```

  Examine the response to find your project name and ID as in the following example snippet.

  ```
  ...
      ],
      "name": "project1",
      "description": "",
      "id": "6c2f2d0d-ecee-42e3-90be-7bb66d6da2f9",
      "orgId": "f098d692-e980-41a5-b349-83084fce1ea0",
  ...
  ```

Querying zones associated with a project

You can retrieve the first 100 cloud zones associated with a project without including a query option. To retrieve information about cloud zones that are not among the first 100 listed, you add query options to your request.

For a complete list of query options, see Querying with the Automation APIs.

This example assumes that you have more than 100 zones associated with the project ID `6c2f2d0d-ecee-42e3-90be-7bb66d6da2f9`. The following procedure shows how to use paging to get the second page of zones associated with the project.

For more information about pagination parameters, see Using Pagination and Count.

Assign the project ID variable.

```
project_id='6c2f2d0d-ecee-42e3-90be-7bb66d6da2f9'
```

Append query options `top=100` and `skip=100` to the request to retrieve cloud zones.

```
curl -X GET \
"$url/iaas/api/projects/$project_id/zones?apiVersion=$api_version&$top=100&skip=100" \
-H 'Accept: application/json' \
-H "Authorization: Bearer $access_token" \
| jq "."
```

The following response lists the second page of zones. Since there are 102 zones associated with the project, the second page lists two zones.


```
"content": [
   {
      "zoneId": "3cf514d6-0dfc-4941-95de-40b01d60e8d3",
      "priority": 0,
      "maxNumberInstances": 0,
      "allocatedInstancesCount": 2,
      "memoryLimitMB": 0,
      "allocatedMemoryMB": 1254,
      "cpuLimit": 0,
      "allocatedCpu": 2,
      "gpuLimit": 0,
      "allocatedGpu": 0,
      "storageLimitGB": 0,
      "allocatedStorageGB": 0.0,
      "id": "0966203d-63f5-41c7-8dcd-7c1833932ec4-3cf514d6-0dfc-4941-95de-40b01d60e8d3",
      "updatedAt": "2021-10-28",
      "orgId": "ce811934-ea1a-4f53-b6ec-465e6ca7d126",
      "_links": {
         "project": { "href": "/iaas/api/projects/6c2f2d0d-ecee-42e3-90be-7bb66d6da2f9" }
         }
   }   ,
   {
      "zoneId": "e4c56d64-a5bc-4656-bfc6-9f8009af66d3",
      "priority": 0,
      "maxNumberInstances": 0,
      "allocatedInstancesCount": 0,
      "memoryLimitMB": 0,
      "allocatedMemoryMB": 0,
      "cpuLimit": 0,
      "allocatedCpu": 0,
      "gpuLimit": 0,
      "allocatedGpu": 0,
      "storageLimitGB": 0,
      "allocatedStorageGB": 0.0,
      "id": "0966203d-63f5-41c7-8dcd-7c1833932ec4-e4c56d64-a5bc-4656-bfc6-9f8009af66d3",
      "updatedAt": "2022-01-07",
      "orgId": "ce811934-ea1a-4f53-b6ec-465e6ca7d126",
      "_links": {
         "project":{ "href": "/iaas/api/projects/6c2f2d0d-ecee-42e3-90be-7bb66d6da2f9" }
      }
   }
],
"totalElements": 2,
"numberOfElements": 2
```


Editing cloud zone assignments

You can edit the following cloud zone assignments in your project:

- Storage limit (GB)

- CPU limit
- Memory limit (MB)
- Maximum number of instances
- Provisioning priority

This example assumes that you have two zones associated with the project ID `094a2fab-7715-4844-94f9-71b45452da27`, one with provisioning priority 0 and one with provisioning priority 1. The following procedure shows how to edit the zone assignments to add a new zone with priority 1 and change an existing zone to priority 2.

Assign the project ID variable.

```
project_id='094a2fab-7715-4844-94f9-71b45452da27'
```

List the zones provisioned in your project.

```
curl -X GET \
"$url/iaas/api/projects/$project_id/zones?apiVersion=$api_version&$top=100&skip=100" \
-H 'Accept: application/json' \
-H "Authorization: Bearer $access_token" \
| jq "."
```

A snippet of the response shows two zone IDs:

- `3c2bbe36-bf8e-4484-9c31-ce552422aaf1`
- `8992bdf0-136f-401c-822a-e22dae67259b`

```
...
  }
     "zoneId": "3c2bbe36-bf8e-4484-9c31-ce552422aaf1",
     "priority": 0
     "maxNumberInstances": 0,
     "allocatedInstancesCount":0,
     ...
  },
  {
     "zoneId": "8992bdf0-136f-401c-822a-e22dae67259b",
     "priority": 1
     "maxNumberInstances": 0,
     "allocatedInstancesCount":0,
     ...
  }
}
...
```

Note When updating project zone assignments, the update deletes any cloud zones that are not included in the request payload. So when making the request, you must include zone IDs for all cloud zones that will remain in the project after the update, even if there is no change to the zone assignment specification.

The zone assignment update includes information for three zones in the project:

- For zone ID `3c2bbe36-bf8e-4484-9c31-ce552422aaf1`, maintain the zone with **`"priority": 0`**.
- For zone ID `8992bdf0-136f-401c-822a-e22dae67259b`, maintain the zone but change to **`"priority": 2`**.
- For zone ID `66067958-7e43-47f8-9bc9-0d32594c47e9`, add a new zone with **`"priority": 1`**.

```
curl -X PUT \
  "$url/iaas/api/projects/$project_id/zones?apiVersion=$api_version"
  -H 'Content-Type: application/json'
  -H "Authorization: Bearer $access_token"
  -d '{
   "zoneAssignmentSpecifications": [
        {
            "zoneId": "3c2bbe36-bf8e-4484-9c31-ce552422aaf1",
            "priority": 0
        },
        {
            "zoneId": "8992bdf0-136f-401c-822a-e22dae67259b",
            "priority": 2
        }.
        {
            "zoneId": "66067958-7e43-47f8-9bc9-0d32594c47e9",
            "priority": 1
        }
    }
  }' | jq "."
```

Examine the response.

```
{
"progress": 0,
"status": "INPROGRESS",
"name": "Project Zones Assignment Task",
"id": "a6241aeb-909e-4689-af5a-940b52d216ff",
"selfLink": "/iaas/api/request-tracker/a6241aeb-909e-4689-af5a-940b52d216ff"
}
```

Assign the selflink variable.

```
selfLink_id='a6241aeb-909e-4689-af5a-940b52d216ff'
```

Use the selfLink for tracking.

```
curl -X GET -H 'Content-Type: application/json' -H "Authorization: Bearer $access_token"
"$url/iaas/api/request-tracker/$selfLink_id?apiVersion=$api_version" | jq "."
```

The zone assignments are complete when the response includes `"status": "FINISHED"`.

```
{
   "progress": 100,
   "status": "FINISHED",
```

```
    "name": "Project Zones Assignment Task",
    "id": "a6241aeb-909e-4689-af5a-940b52d216ff",
    "selfLink": "/iaas/api/request-tracker/a6241aeb-909e-4689-af5a-940b52d216ff"
}
```

If you list the zones provisioned in your project again, a snippet of the response shows the newly added cloud zone and updated provisioning priorities.

```
...
  }
     "zoneId": "3c2bbe36-bf8e-4484-9c31-ce552422aaf1",
     "priority": 0
     "maxNumberInstances": 0,
     "allocatedInstancesCount":0,
     ...
  },
  {
     "zoneId": "8992bdf0-136f-401c-822a-e22dae67259b",
     "priority": 2
     "maxNumberInstances": 0,
     "allocatedInstancesCount":0,
     ...
  }
  {
     "zoneId": "66067958-7e43-47f8-9bc9-0d32594c47e9",
     "priority": 1
     "maxNumberInstances": 0,
     "allocatedInstancesCount":0,
     ...
  }
}
...
```

Deploying and Managing Resources

As a Automation Assembler administrator, you can use the IaaS APIs to deploy and manage cloud resources.

Create and Deploy a Machine Resource

To create a new resource such as a VM, you can use the resources API to make a POST request with a project ID. The deployment creates a new resource without using a cloud template.

This procedure shows how to provision a VM with a project and a cloud zone assigned to the project. The flavor and image for the VM must exist in your cloud account.

Prerequisites

- Verify that all general prerequisites and prerequisites for the Automation Assembler Infrastructure as a Service (IaaS) service have been satisfied. See Chapter 3 Prerequisites for API Use Case Examples.

- Assign an API version variable for the Deployment API.

```
api_version_deployment='2020-08-25'
```

Note The Automation Assembler Infrastructure as a Service (IaaS) service and the Automation Service Broker Deployment service have different API version values. You set the API version value for the Automation Assembler Infrastructure as a Service (IaaS) service when you satisfied the general prerequisites.

- Verify that you have the ID of the cloud account where you want to deploy the VM. See Adding Cloud Accounts.
- Verify that you know the cloud zone in your cloud account where you want the new VM to be deployed. See Create a Cloud Zone .
- Verify that you have the ID for a project that includes the cloud zone in your cloud account. See Add a Cloud Zone to Your Project.
- Verify that the flavor and image for the new VM exist in your cloud account.
- Verify that you know the resource type for the AWS machine, GCP machine, vSphere machine, or Azure machine that you plan to create. For a list of all resource types and request schema, see the link to the schema that's available from VMware Aria Automation Resource Schema Documentation.

Procedure

1 Assign the cloud account ID variable.

```
cloud_account_id='<your_cloud_account_id>'
```

2 Assign the project ID variable.

```
project_id='<your_project_id>'
```

3 List all cloud zones.

```
curl -X GET -H 'Content-Type: application/json' -H "Authorization: Bearer $access_token"
"$url/iaas/api/zones?apiVersion=$api_version"  | jq "."
```

4 Examine the response for the zone where you want to place your VM.

You use the **`externalRegionId`** to filter for fabric flavor and fabric image. You use the cloud zone ID for VM placement.

5 Assign the external region ID variable.

```
external_region_id='<your_external_region_id>'
```

6 Assign the cloud zone placement variable.

```
placement_name='/iaas/api/zone/<your_cloud_zone_id>'
```

7 To list the fabric images in your cloud account and zone, specify the cloud account ID and external region ID in the request.

```
curl -X GET -H 'Content-Type: application/json' -H "Authorization: Bearer $access_token" \
    "$url/iaas/api/fabric-images/?apiVersion=$api_version&"\
    '$filter'="(((externalRegionId eq '') or (externalRegionId eq '$external_region_id')) \
    and ((cloudAccountId ne '*') or cloudAccountId eq '$cloud_account_id')))" | jq "."
```

8 Examine the response to select a fabric image.

9 Assign the fabric image variable.

```
image_name='<your_image_name>'
```

10 To list the fabric flavors in your cloud account and zone, specify the cloud account ID and external region ID in the request.

```
curl -X GET -H 'Content-Type: application/json' -H "Authorization: Bearer $access_token" \
    "$url/iaas/api/fabric-flavors/?apiVersion=$api_version&"\
    '$filter'="(((externalRegionId eq '') or (externalRegionId eq '$external_region_id')) \
    and ((cloudAccountId ne '*') or cloudAccountId eq '$cloud_account_id')))" | jq "."
```

11 Examine the response to select a fabric flavor.

12 Assign the fabric flavor variable.

```
flavor_name='<your_flavor_name>'
```

13 Assign the resource type variable.

```
resource_type='<your_resource_type>'
```

14 Create and deploy the VM.

```
curl -X POST \
  $url/deployment/api/resources?apiVersion=$api_version_deployment \
  -H "Authorization: Bearer $access_token" \
  -H "Content-Type: application/json" \
  -d '{
    "name":"<your_resource_name>",
    "projectId": "'$project_id'",
    "type":"'$resource_type'",
    "properties":{
        "imageRef": "'$image_name'",
        "flavor": "'$flavor_name'",
        "placement": "'$placement_name'"
  },
}' | jq "."
```

15 Examine the response for the deployment ID.

16 Assign the deployment ID.

17 Get the status of the deployment.

```
curl -X GET \
  $url/deployment/api/deployments/$deployment_id?apiVersion=$api_version_deployment \
  -H "Authorization: Bearer $access_token" | jq "."
```

When the status shows `CREATE_SUCCESSFUL` the VM is deployed.

Example: Create and deploy a VM

With a cloud account ID and cloud zone in the cloud account, create and deploy an AWS machine named **`cloud_machine_1`** using the **`Cloud.AWS.EC2.Instance`** resource type.

Assign variables.

```
$ url='https://appliance.domain.com'
$ api_version='2021-07-15'
$ api_version_deployment='2020-08-25'
$ cloud_account_id='14e6b70c-0e76-4c5e-bb61-e6d70a5b43ef'
$ project_id='394a4ccb-22c6-4ef0-8c75-8b77efbefb51'
```

List all cloud zones.

```
curl -X GET -H 'Content-Type: application/json' -H "Authorization: Bearer $access_token"
"$url/iaas/api/zones?apiVersion=$api_version"  | jq "."
```

A snippet of the response shows the cloud account ID with the external region ID and cloud zone ID.

```
...
      "externalRegionId": "eu-west-1",
      "cloudAccountId": "14e6b70c-0e76-4c5e-bb61-e6d70a5b43ef",
      "name": "AWS / eu-west-1-changed",
      "description": "test description",
      "id": "f32a30fd-67ac-43e3-9512-60cf6ef7bee8"
...
```

Assign the external region ID variable.

```
$ external_region_id='eu-west-1'
```

Assign the cloud zone placement variable.

```
$ placement_name='/iaas/api/zone/f32a30fd-67ac-43e3-9512-60cf6ef7bee8'
```

To list the fabric images in your cloud account and zone, specify the cloud account ID and external region ID in the request.

```
curl -X GET -H 'Content-Type: application/json' -H "Authorization: Bearer $access_token" \
    "$url/iaas/api/fabric-images/?apiVersion=$api_version&"\
    '$filter'="(((externalRegionId eq '') or (externalRegionId eq '$external_region_id')) \
    and ((cloudAccountId ne '*') or cloudAccountId eq '$cloud_account_id')))" | jq "."
```

Examine the response to find the image ID that you want.

```
...
      "externalRegionId": "eu-west-1",
      "isPrivate": false,
      "externalId": "ami-9a9012e9",
...
```

To list the fabric flavors in your cloud account and zone, specify the cloud account ID and external region ID in the request.

```
curl -X GET -H 'Content-Type: application/json' -H "Authorization: Bearer $access_token" \
    "$url/iaas/api/fabric-flavors/?apiVersion=$api_version&"\
    '$filter'="(((externalRegionId eq '') or (externalRegionId eq '$external_region_id')) \
    and ((cloudAccountId ne '*') or cloudAccountId eq '$cloud_account_id')))" | jq "."
```

Examine the response to find the flavor ID that you want.

```
...          {
      "id": "x1e.xlarge",
      "name": "x1e.xlarge",
      "cpuCount": 4,
      "memoryInMB": 124928,
      "storageType": "ssd",
      "dataDiskSizeInMB": 122880,
      "dataDiskMaxCount": 1,
      "networkType": "Up to 10 Gigabit"
    },
...
```

Assign the resource type for the VM.

```
$ resource_type = 'Cloud.AWS.EC2.Instance'
```

To deploy the VM, assign variables for image and flavor.

```
$ image_name='ami-9a9012e9'
$ flavor_name='x1e.xlarge'
```

Create and deploy the VM.

```
curl -X POST \
  $url/deployment/api/resources?apiVersion=$api_version_deployment \
  -H "Authorization: Bearer $access_token" \
  -H "Content-Type: application/json" \
  -d '{
    "name":"cloud_machine_1",
    "projectId": "'$project_id'",
    "type":"'$resource_type'",
    "properties":{
        "imageRef": "'$image_name",
        "flavor": "'$flavor_name",
        "placement": "'$placement_name'"
```

```
  },
}' | jq "."
```

The response shows the deployment ID.

```
{
    "deploymentId": "fccd2081-fd44-44c8-878c-f962ef71969a",
    "projectId": "394a4ccb-22c6-4ef0-8c75-8b77efbefb51",
    "requestId": "1f8f2e4f-0b2e-448d-8439-f1a05b1e90c1",
    "resourceId": "9a510ccb-0543-47e8-a2f2-f1f65fcd0b0a"
}
```

Assign the deployment ID variable.

```
$ deployment_id='fccd2081-fd44-44c8-878c-f962ef71969a'
```

Get the status of the deployment.

```
$ curl -X GET \
  $url/deployment/api/deployments/$deployment_id?apiVersion=$api_version"
  -H "Authorization: Bearer $access_token" | jq "."
```

A snippet of the response shows the deployment status.

```
...
  ],
  "status": "CREATE_SUCCESSFUL"
}
```

Managing IP Addresses

As an Automation Assembler administrator, you can reserve IP addresses so that VMware Aria Automation does not use them for deployment. If you no longer need them, you can release them.

Before reserving IP addresses, you must verify that available IP addresses exist within a network IP range.

Query for IP Addresses

When you use Automation Assembler to create a deployment, VMware Aria Automation allocates IP addresses to manage resources. To identify the IP addresses that have been allocated, you query the network IP range. You can only reserve IP addresses that have not been allocated.

You create network IP ranges with a range of IP addresses. The following procedure shows how to determine the IP addresses in a range that are available.

Prerequisites

- Verify that all general prerequisites and prerequisites for the Automation Assembler Infrastructure as a Service (IaaS) service have been satisfied. See Chapter 3 Prerequisites for API Use Case Examples.

Procedure

1. To get the ID of the network range that you want to use, list the internal network IP ranges in your deployment.

   ```
   curl -X GET -H 'Content-Type: application/json' -H "Authorization: Bearer $access_token"
   "$url/iaas/api/network-ip-ranges?apiVersion=$api_version"  | jq "."
   ```

   Examine the response to find the ID of the network IP range that you want to use.

2. Assign a variable for the network IP range ID.

   ```
   ip_range_id='<your_network_ip_range_id>'
   ```

3. To query IP ranges associated with a particular network and get information such as IP availability, you can use one of the following commands:

   - If you have the network IP range ID, use:

     ```
     curl -X GET -H 'Content-Type: application/json' -H "Authorization: Bearer
     $access_token" "$url/iaas/api/network-ip-ranges/$ip_range_id?apiVersion=$api_version"
     | jq "."
     ```

   - If you have the network ID, use:

     ```
     curl -X GET -H 'Content-Type: application/json' -H "Authorization:
     Bearer $access_token" "$url/iaas/api/networks/<your_network_id>/network-ip-ranges?
     apiVersion=$api_version"  | jq "."
     ```

   - If you have the networking fabric ID, use:

     ```
     curl -X GET -H 'Content-Type: application/json' -H "Authorization:
     Bearer $access_token" "$url/iaas/api/fabric-networks/<your_fabric_network_id>/network-
     ip-ranges?apiVersion=$api_version"  | jq "."
     ```

   - If you have the networking fabric for vSphere ID, use:

     ```
     curl -X GET -H 'Content-Type: application/json' -H "Authorization: Bearer
     $access_token" "$url/iaas/api/fabric-networks-vsphere/<your_fabric_network_vsphere_id>/
     network-ip-ranges?apiVersion=$api_version"  | jq "."
     ```

   Examine the response to see the number of allocated and available IP addresses. If the number of available IP addresses is too low, check another network IP range until you find one with enough IP addresses to fit your needs.

4 Use the network IP range ID to query the status of IP addresses within the range.

```
curl -X GET -H 'Content-Type: application/json' -H "Authorization: Bearer $access_token"
"$url/iaas/api/network-ip-ranges/$ip_range_id/ip-addresses?apiVersion=$api_version"  | jq
"."
```

Examine the result. All IP addresses appear with `"ipAddressStatus": "ALLOCATED"` or `"ipAddressStatus": "RELEASED"` and are not available.

Example: Query for IP addresses within an IP range

The example query requests and responses show how to find allocated IP addresses.

Assign variables.

```
$ url='https://appliance.domain.com'
$ api_version='2021-07-15'
```

List the internal network IP ranges in your deployment.

```
$ curl -X GET -H 'Content-Type: application/json' -H "Authorization: Bearer $access_token"
"$url/iaas/api/network-ip-ranges?apiVersion=$api_version"  | jq "."
```

A snippet of the response shows the network IP ranges with their IDs.

```
...
      "startIPAddress": "fe45::2",
      "orgId": "8040fbde-5ff0-41f3-a8a1-9e25a3311be2",
      "endIPAddress": "fe45::fffe",
      "createdAt": "2023-02-28",
      "ipVersion": "IPv6",
      "name": "rangeIPv6",
      "id": "703a02df-3b6d-4ad7-b146-582b398055f2",
...
```

Assign the variable for the network IP range ID.

```
$ ip_range_id='703a02df-3b6d-4ad7-b146-582b398055f2'
```

Query the IP range associated with the network IP range ID.

```
$ curl -X GET -H 'Content-Type: application/json' -H "Authorization: Bearer $access_token"
"$url/iaas/api/network-ip-ranges/$ip_range_id?apiVersion=$api_version"  | jq "."
```

A snippet of the response shows the number of available and allocated IPv6 addresses. It also shows the IP version and the start and end of the IP address sequence.

```
{
    "numberOfAllocatedIPs": 2,
    "numberOfAvailableIPs": 65531,
    "numberOfReleasedIPs": 0,
    "totalNumberOfIPs": 65533,
    "numberOfUserAllocatedIPs": 0,
```

```
    "numberOfSystemAllocatedIPs": 2,
    "startIPAddress": "fe45::2",
    "endIPAddress": "fe45::fffe",
    "ipVersion": "IPv6",
    "name": "rangeIPv6",
    "id": "703a02df-3b6d-4ad7-b146-582b398055f2",
    "createdAt": "2023-02-28",
    "updatedAt": "2023-02-28",
    "organizationId": "8040fbde-5ff0-41f3-a8a1-9e25a3311be2",
    "orgId": "8040fbde-5ff0-41f3-a8a1-9e25a3311be2",
    "_links": {
        "self": {
            "href": "/iaas/api/network-ip-ranges/703a02df-3b6d-4ad7-b146-582b398055f2"
        }
    }
}
```


The following example shows how to query of IP ranges associated with fabric network ID **33c2bbb5-5b26-4a5a-87c6-9e96db72451d**.

```
curl -X GET -H 'Content-Type: application/json' -H "Authorization: Bearer
$access_token" "$url/iaas/api/fabric-networks/33c2bbb5-5b26-4a5a-87c6-9e96db72451d/network-ip-
ranges?apiVersion=$api_version"  | jq "."
```

The response shows two IP ranges with details about associated IPv4 addresses.


```
"content": [
        {
            "numberOfAllocatedIPs": 0,
            "numberOfAvailableIPs": 10,
            "numberOfReleasedIPs": 0,
            "totalNumberOfIPs": 10,
            "numberOfUserAllocatedIPs": 0,
            "numberOfSystemAllocatedIPs": 0,
            "startIPAddress": "10.10.10.1",
            "endIPAddress": "10.10.10.10",
            "ipVersion": "IPv4",
            "name": "range-1",
            "id": "578ec19a-2b52-497e-8777-029b57d685ab",
            "createdAt": "2023-03-15",
            "updatedAt": "2023-03-29",
            "orgId": "89db05f7-1b93-4bd8-b395-1772d50813a4",
            "_links": {
                "fabric-networks": {
                    "hrefs": [
                        "/iaas/api/fabric-networks/33c2bbb5-5b26-4a5a-87c6-9e96db72451d",
                        "/iaas/api/fabric-networks/51f1ed6c-3db3-4877-857d-2bcc84f81897"
                    ]
                },
                "self": {
                    "href": "/iaas/api/network-ip-ranges/578ec19a-2b52-497e-8777-029b57d685ab"
                }
            }
        },
```

```
        {
            "numberOfAllocatedIPs": 0,
            "numberOfAvailableIPs": 10,
            "numberOfReleasedIPs": 0,
            "totalNumberOfIPs": 10,
            "numberOfUserAllocatedIPs": 0,
            "numberOfSystemAllocatedIPs": 0,
            "startIPAddress": "10.10.10.11",
            "endIPAddress": "10.10.10.20",
            "ipVersion": "IPv4",
            "name": "range-2",
            "id": "95a0053c-b57f-4500-a59b-970042f4ce8c",
            "createdAt": "2023-03-29",
            "updatedAt": "2023-03-29",
            "orgId": "89db05f7-1b93-4bd8-b395-1772d50813a4",
            "_links": {
                "fabric-networks": {
                    "hrefs": [
                        "/iaas/api/fabric-networks/33c2bbb5-5b26-4a5a-87c6-9e96db72451d"
                    ]
                },
                "self": {
                    "href": "/iaas/api/network-ip-ranges/95a0053c-b57f-4500-a59b-970042f4ce8c"
                },
                "fabric-network": {
                    "href": "/iaas/api/fabric-networks/33c2bbb5-5b26-4a5a-87c6-9e96db72451d"
                }
            }
        }
    ],
    "totalElements": 2,
    "numberOfElements": 2
}
```


If the IP range has a sufficient number of available IP addresses, get more information about the IP addresses in that range.

```
 $ curl -X GET -H 'Content-Type: application/json' -H "Authorization: Bearer $access_token"
"$url/iaas/api/network-ip-ranges/$ip_range_id/ip_addresses?apiVersion=$api_version"  | jq "."
```

The response provides details about the umavailable IP addresses including their IP version and how they were allocated.

- `"ipAllocationType": "SYSTEM"` indicates that an IP address was automatically allocated for use in a deployment.
- `"ipAllocationType": "USER"` indicates that an IP address was manually allocated for use by a user.


```
{
    "content": [
        {
            "ipAddress": "fe4500:10:118:136:fcd8:d68d:9701:4440",
            "ipAddressDecimalValue": 168561938,
```

```
            "ipVersion": "IPv6",
            "ipAddressStatus": "ALLOCATED",
            "ipAllocationType": "SYSTEM",
            "id": "e83d9a43-ea2d-428c-ae75-da396c1bb205",
            "createdAt": "2023-02-21",
            "updatedAt": "2023-02-21",
            "orgId": "8040fbde-5ff0-41f3-a8a1-9e25a3311be2",
            "_links": {
                "network-ip-range": {
                    "href": "/iaas/api/network-ip-ranges/4e6b700c-5d07-4247-b0c7-ae13d1d7188a"
                },
                "self": {
                    "href": "/iaas/api/network-ip-ranges/4e6b700c-5d07-4247-b0c7-ae13d1d7188a/
ip-addresses/e83d9a43-ea2d-428c-ae75-da396c1bb205"
                },
                "connected-resource": {
                    "href": "/iaas/api/machines/83ede26c-656a-4a08-8716-bea29c21d3f4/network-
interfaces/051d92e6-5729-495d-ad66-9e443e5747c8"
                }
            }
        },
        {
            "ipAddress": "fe45:10:118:136:fcd8:d68d:9701:4450",
            "ipAddressDecimalValue": 168561941,
            "ipVersion": "IPv6",
            "ipAddressStatus": "ALLOCATED",
            "ipAllocationType": "SYSTEM",
            "id": "439f9c9b-2b2f-484d-8867-2d7b541ddeec",
            "createdAt": "2023-02-22",
            "updatedAt": "2023-02-22",
            "orgId": "8040fbde-5ff0-41f3-a8a1-9e25a3311be2",
            "_links": {
                "network-ip-range": {
                    "href": "/iaas/api/network-ip-ranges/4e6b700c-5d07-4247-b0c7-ae13d1d7188a"
                },
                "self": {
                    "href": "/iaas/api/network-ip-ranges/4e6b700c-5d07-4247-b0c7-ae13d1d7188a/
ip-addresses/439f9c9b-2b2f-484d-8867-2d7b541ddeec"
                },
                "connected-resource": {
                    "href": "/iaas/api/machines/81633fba-e86b-4bfa-a06f-3c4f7a754568/network-
interfaces/1c26b194-2862-4d7e-be9c-9881d6cdb871"
                }
            }
        }
    ],
    "totalElements": 2,
    "numberOfElements": 2
}
```


You can allocate any of the IP addresses in the network range except `fe4500:10:118:136:fcd8:d68d:9701:4440` and `fe45:10:118:136:fcd8:d68d:9701:4450`.

What to do next

IP addresses with the status `"ipAddressStatus": "ALLOCATED"` or `"ipAddressStatus": "RELEASED"` are not available for allocation. You can allocate any of the remaining the IP addresses in the range. See Allocate IP Addresses.

Allocate IP Addresses

As a cloud administrator, you can use the network IP range API to allocate the IP addresses that you want to reserve and make them unavailable for deployment by VMware Aria Automation.

To allocate IP addresses, you submit a request with either:

- A list of specific IP addresses.
- The number of IP addresses.

You can also provide an optional description.

Note The following restrictions apply to the request:

- No more than 100 IP addresses can be allocated in a single call.
- If listing specific IP addresses, the addresses must all be of the same type. Mixing IPv4 and IPv6 addresses is not supported.

Prerequisites

- Verify that all general prerequisites and prerequisites for the Automation Assembler Infrastructure as a Service (IaaS) service have been satisfied. See Chapter 3 Prerequisites for API Use Case Examples.
- Verify that you have the ID of the network IP range that includes the IP addresses that you want to reserve. See Query for IP Addresses.
- If you plan to specify the IP addresses from the network IP range, verify that you know the IPv4 or IPv6 addresses. See Query for IP Addresses.
- If you are a cloud administrator and you plan to specify the number of IP addresses from the network IP range, verify that the number is less than or equal to the number of available IP addresses in the range.

Procedure

1 Assign the variable for the network IP range ID.

```
ip_range_id='<your_network_ip_range_id>'
```

2 Allocate the IP addresses by providing a list of addresses.

The following request example allocates two IP addresses.

- Specify either IPv4 or IPv6 addresses.

- To release additional IP addresses, provide additional addresses in the payload.

```
curl --location --request POST \
  $url/iaas/api/network-ip-ranges/$ip_range_id/ip-addresses/allocate?
apiVersion=$api_version \
  -H "Authorization: Bearer $access_token" \
  -H 'Content-Type: application/json' \
  -d '{
     "description":"<your_optional_description>",
     "ipAddresses":["<ipv_address_1>", "<ipv_address_2>"]
}'
```

Examine the response for the self link to track the request.

3 (Optional) If you do not list IP addresses, provide the number of IP addresses to allocate.

```
curl --location --request POST \
  $url/iaas/api/network-ip-ranges/$ip_range_id/ip-addresses/allocate?
apiVersion=$api_version \
  -H "Authorization: Bearer $access_token" \
  -H 'Content-Type: application/json' \
  -d '{
     "description":"<your_optional_description",
     "numberOfIps": "<number_of_IPs>"
}'
```

Examine the response for the self link to track the request.

4 Assign the selfLink variable.

```
selfLink_id='example-selfLink-alphanumeric-string'
```

5 Use the selfLink variable to track the request.

```
curl -X GET -H 'Content-Type: application/json' -H "Authorization: Bearer $access_token"
"$url/iaas/api/request-tracker/$selfLink_id?apiVersion=$api_version" | jq "."
```

The IPs are allocated when the response includes `"status": "FINISHED"` and provides a list of allocated resources.

Note If you specified the number of IP addresses to allocate, you query for the details of each resource in the response to get the allocated IP address associated with the resource.

```
curl -X GET -H 'Content-Type: application/json' -H "Authorization: Bearer
$access_token" "$url/iaas/api/network-ip-ranges/$ip_range_id/ip-addresses/<ip_address_id>?
apiVersion=$api_version"  | jq "."
```

Example: Allocate IP addresses within an IP range

Using the network IP range with the IP addresses that you want to reserve, specify the IP addresses to allocate.

- Two IPv4 IP addresses used in this example are: `["196.192.2.19", "196.192.2.20"]`

- Two IPv6 IP addresses used in this example are:
  `["fe45:10:118:136:fcd8:d68d:9701:8975","fe45:10:118:136:fcd8:d68d:9701:8985"]`

Assign variables.

```
$ url='https://appliance.domain.com'
$ api_version='2021-07-15'
$ ip_range_id='703a02df-3b6d-4ad7-b146-582b398055f2'
```

Allocate IPv4 addresses by providing a list.

```
curl --location --request POST \
  $url/iaas/api/network-ip-ranges/$ip_range_id/ip-addresses/allocate?apiVersion=$api_version \
  -H "Authorization: Bearer $access_token" \
  -H 'Content-Type: application/json' \
  -d '{
     "description":"Allocate ipv4 addresses for QA test machines",
     "ipAddresses":["196.192.2.19", "196.192.2.20"]
}'
```

Or allocate IPv6 addresses by providing a list.

```
curl --location --request POST \
  $url/iaas/api/network-ip-ranges/$ip_range_id/ip-addresses/allocate?apiVersion=$api_version \
  -H "Authorization: Bearer $access_token" \
  -H 'Content-Type: application/json' \
  -d '{
     "description":"Allocate ipv6 addresses for QA test machines",
     "ipAddresses":
["fe45:10:118:136:fcd8:d68d:9701:8975","fe45:10:118:136:fcd8:d68d:9701:8985"]
}'
```

Or specify the number of IP addresses to allocate, such as four.

```
curl --location --request POST \
  $url/iaas/api/network-ip-ranges/$ip_range_id/ip-addresses/allocate?apiVersion=$api_version \
  -H "Authorization: Bearer $access_token" \
  -H 'Content-Type: application/json' \
  -d '{
     "description":"Allocate four IP addresses for QA test machines",
     "numberOfIps": "4"
}'
```

The response provides the selfLink variable for tracking.

```
{
    "progress": 1,
    "status": "INPROGRESS",
    "name": "IP Address Allocation Task",
    "id": "ab726e31-ed8bfdfb-83d9-4964-b719-8b395c87d2c9",
    "selfLink": "/iaas/api/request-tracker/ab726e31-ed8bfdfb-83d9-4964-b719-8b395c87d2c9"
}
```

Assign the selfLink variable.

```
selfLink_id= 'ab726e31-e909-4bcc-87ef-5139294eb26a'
```

Track the request

```
curl -X GET -H 'Content-Type: application/json' -H "Authorization: Bearer $access_token"
"$url/iaas/api/request-tracker/$selfLink_id?apiVersion=$api_version" | jq "."
```

When the request is complete, the response includes `"status": "FINISHED"`.

- This is an example of the response when two IPv4 or IPv6 addresses are specified and allocated. You know the IP addresses because you provided them in the request.

```
{
    "progress": 100,
    "status": "FINISHED",
    "resources": [
        "/iaas/api/network-ip-ranges/703a02df-3b6d-4ad7-b146-582b398055f2/ip-addresses/
6212ac8b-a885-4416-bf0d-ebb88628bef1",
        "/iaas/api/network-ip-ranges/703a02df-3b6d-4ad7-b146-582b398055f2/ip-addresses/
9b65104d-debd-4830-8296-3cb87509d293"
    ],
    "name": "IP Address Allocation Task",
    "id": "ab726e31-e909-4bcc-87ef-5139294eb26a",
    "selfLink": "/iaas/api/request-tracker/ab726e31-e909-4bcc-87ef-5139294eb26a"
}
```

- This is an example of the response when four IP addresses are requested and allocated.

```
{
    "progress": 100,
    "status": "FINISHED",
    "resources": [
        "/iaas/api/network-ip-ranges/703a02df-3b6d-4ad7-b146-582b398055f2/ip-addresses/
a4cceea7-3a94-439b-99fa-420774e61eba",
        "/iaas/api/network-ip-ranges/703a02df-3b6d-4ad7-b146-582b398055f2/ip-addresses/
4c492669-5ec2-4465-992b-c9c252edf052",
        "/iaas/api/network-ip-ranges/703a02df-3b6d-4ad7-b146-582b398055f2/ip-addresses/
0ce9ee20-1a40-460d-80df-fb1791fbd42a",
        "/iaas/api/network-ip-ranges/703a02df-3b6d-4ad7-b146-582b398055f2/ip-addresses/
d7d876b6-92b1-4530-a305-d586c8474f11"
    ],
    "name": "IP Address Allocation Task",
    "id": "ab726e31-e909-4bcc-87ef-5139294eb26a",
    "selfLink": "/iaas/api/request-tracker/ab726e31-e909-4bcc-87ef-5139294eb26a"
}
```

To get an IP address, query for the details of an allocated resource. The following example queries for the details of the resource with IP address ID **a4cceea7-3a94-439b-99fa-420774e61eba**.

```
curl -X GET -H 'Content-Type: application/json' -H "Authorization:
Bearer $access_token" "$url/iaas/api/network-ip-ranges/$ip_range_id/ip-addresses/
a4cceea7-3a94-439b-99fa-420774e61eba?apiVersion=$api_version"  | jq "."
```

The response provides details about the resource including the allocated IP address and IP version. `" ipAllocationType": "USER"` indicates that an Automation Assembler administrator manually reserved the IP address.

```
{
            "ipAddress": "196.192.2.20",
            "ipAddressDecimalValue": 168561941,
            "ipVersion": "IPv4",
            "ipAddressStatus": "ALLOCATED",
            "ipAllocationType": "USER",
            "id": "439f9c9b-2b2f-484d-8867-2d7b541ddeec",
            "createdAt": "2023-02-22",
            "updatedAt": "2023-02-22",
            "orgId": "8040fbde-5ff0-41f3-a8a1-9e25a3311be2",
            "_links": {
                "network-ip-range": {
                    "href": "/iaas/api/network-ip-ranges/703a02df-3b6d-4ad7-
b146-582b398055f2"
                },
                "self": {
                    "href": "/iaas/api/network-ip-ranges/703a02df-3b6d-4ad7-
b146-582b398055f2/ip-addresses/439f9c9b-2b2f-484d-8867-2d7b541ddeec"
                },
                "connected-resource": {
                    "href": "/iaas/api/machines/81633fba-e86b-4bfa-a06f-3c4f7a754568/
network-interfaces/1c26b194-2862-4d7e-be9c-9881d6cdb871"
                }
            }
        }
    ]
}
```

What to do next

If you no longer need allocated IP addresses, you can release them. See Release IP Addresses.

Release IP Addresses

If you no longer need reserved IP addresses, you can release them so that VMware Aria Automation can use them for deployment.

This procedure shows how to request specific IP addresses to release.

Prerequisites

- Verify that all general prerequisites and prerequisites for the Automation Assembler Infrastructure as a Service (IaaS) service have been satisfied. See Chapter 3 Prerequisites for API Use Case Examples.
- Verify that you have the ID of the network IP range that includes the IP addresses that you want to release. See Query for IP Addresses.
- Verify that you know the IPv4 or IPv6 addresses that you want to release. The IP addresses must all be the same type. Releasing a mix of IPv4 and IPv6 addresses in the same request is not supported. See Allocate IP Addresses.

Procedure

1 Assign the variable for the network IP range ID.

```
ip_range_id='<your_network_ip_range_id>'
```

2 Release the IP addresses.

The following request releases two IP addresses.

- Specify either IPv4 or IPv6 addresses.
- To release more IP addresses, provide the IP addresses in the payload.

Note No more than 100 IP addresses can be released in a single API call.

```
curl --location --request POST \
  $url/iaas/api/network-ip-ranges/$ip_range_id/ip-addresses/release?
apiVersion=$api_version \
  -H "Authorization: Bearer $access_token" \
  -H 'Content-Type: application/json' \
  -d '{
     "ipAddresses":["<ipv_address_1>", "<ipv_address_2>"]
}'
```

Examine the response for the self link to track the request.

3 Assign the self link variable.

```
selfLink_id='example-selfLink-alphanumeric-string'
```

4 Use the selfLink variable to track the request.

```
curl -X GET -H 'Content-Type: application/json' -H "Authorization: Bearer $access_token"
"$url/iaas/api/request-tracker/$selfLink_id?apiVersion=$api_version" | jq "."
```

The IP addresses are released, when the response includes `"status": "FINISHED"`.

Example: Release IP addresses within an IP range

To release IP addresses, make a request with the network IP range and specify the IP addresses to release.

Assign variables.

```
$ url='https://appliance.domain.com'
$ api_version='2021-07-15'
$ ip_range_id='4e6b700c-5d07-4247-b0c7-ae13d1d7188a'
```

Release two IPv4 addresses.

```
curl --location --request POST \
  $url/iaas/api/network-ip-ranges/$ip_range_id/ip-addresses/release?apiVersion=$api_version \
  -H "Authorization: Bearer $access_token" \
  -H 'Content-Type: application/json' \
  -d '{
     "ipAddresses":["196.192.2.19", "196.192.2.20"]
}'
```

The response provides a selfLink for tracking.

```
{
    "progress": 1,
    "status": "INPROGRESS",
    "name": "IP Address Release Task",
    "id": "452bb595-ca0f-4685-8fcc-8bd6321bf7c6",
    "selfLink": "/iaas/api/request-tracker/452bb595-ca0f-4685-8fcc-8bd6321bf7c6"
}
```

Assign the selfLink variable.

```
selfLink_id= '452bb595-ca0f-4685-8fcc-8bd6321bf7c6'
```

Track the request.

```
curl -X GET -H 'Content-Type: application/json' -H "Authorization: Bearer $access_token"
"$url/iaas/api/request-tracker/$selfLink_id?apiVersion=$api_version" | jq "."
```

The IP addresses are released, when the response includes `"status": "FINISHED"`.

```
{
    "progress": 100,
    "status": "FINISHED",
    "resources": [
        "/iaas/api/network-ip-ranges/4e6b700c-5d07-4247-b0c7-ae13d1d7188a/ip-addresses/
3f08cabe-fedf-4e9b-96f6-d3bf69016cc4",
        "/iaas/api/network-ip-ranges/4e6b700c-5d07-4247-b0c7-ae13d1d7188a/ip-addresses/
ca287794-c06b-401c-9626-0b8f5f75d1c2"
    ],
    "name": "IP Address Release Task",
    "id": "452bb595-ca0f-4685-8fcc-8bd6321bf7c6",
```

```
    "selfLink": "/iaas/api/request-tracker/452bb595-ca0f-4685-8fcc-8bd6321bf7c6"
}
```

Creating and Using a First Class Disk

Using a vSphere Storage Profile that supports First Class Disk (FCD) storage, you can create block device-based storage that is independent of a VM.

First Class Disks offer many advantages. For example, you can attach an FCD to a VM and take multiple snapshots of the disk over time. If you find that you do not need certain snapshots, you can delete them. You can also revert an FCD to an earlier snapshot.

Note If you attach an FCD to a VM and then delete the VM, both the FCD and its snapshots are deleted. And if the FCD is detached from a VM, you cannot delete the FCD without deleting its snapshots first.

Create a First Class Disk

To create a First Class Disk (FCD), you make a POST request using the block device specification. The request body includes a project ID, disk capacity, persistence setting, and constraints from the vSphere Storage Profile for an FCD creation.

Prerequisites

- Verify that all general prerequisites and prerequisites for the Automation Assembler Infrastructure as a Service (IaaS) service have been satisfied. See Chapter 3 Prerequisites for API Use Case Examples.
- Verify that you have added a project and you have the project ID. See Create a Project to use in Automation Assembler.
- Know the capacity of the disk and the persistence of the disk that you are creating.
- Verify that you have created a storage profile for an FCD and that you have the `defaultItem` and the tags from the response. See Create a vSphere Storage Profile for a First Class Disk.

Procedure

1 Assign the project ID variable.

```
project_id='<your_project_id>'
```

2 Set the capacity and persistence for the disk.

```
capacity_in_gb=<integer>
persistence=<true|false>
```

3 Deploy the FCD.

- With `mandatory` set to true, all tags in the `expression` must match a storage profile, otherwise provisioning fails.

- The `expression` is the key:value tag pair used to create the storage profile. See Create a vSphere Storage Profile for a First Class Disk.

```
curl -X POST \
  $url/iaas/api/block-devices?apiVersion=$api_version \
  -H 'Content-Type: application/json' \
  -H "Authorization: Bearer $access_token" \
  -d '{
    "projectId": "'$project_id'",
    "name": "FCD-example",
    "capacityInGB": "'$capacity_in_gb'",
    "persistent" : "'$persistent'",
    "constraints": [
        {
            "mandatory": "true",
            "expression": "type:fcd"
        }
    ]
}' | jq "."
```

The response includes a selfLink value.

```
{
  "progress": 0,
  "status": "INPROGRESS",
  "name": "Provisioning",
  "id": "example-selfLink-alphanumeric-string",
  "selfLink": "/iaas/api/request-tracker/example-selfLink-alphanumeric-string"
}
```

4 Assign the selfLink variable.

```
selfLink_id='example-selfLink-alphanumeric-string'
```

5 Use the selfLink variable to track the progress of the FCD creation.

```
curl -X GET -H 'Content-Type: application/json' -H "Authorization: Bearer $access_token"
$url/iaas/api/request-tracker/$selfLink_id?apiVersion=$api_version | jq "."
```

In the list of resources, the response includes block devices with the block device ID in the path.

```
{
  "progress": 100,
  "message": "success",
  "status": "FINISHED",
  "resources": [
    "/iaas/api/block-devices/example-blockdevice-alphanumeric-string"
  ],
  ...
}
```

6 Assign the block device ID variable.

```
block_device_id='example-blockdevice-alphanumeric-string'
```

7 (Optional) Retrieve the created block device object.

```
curl -X GET -H 'Content-Type: application/json' -H "Authorization: Bearer $access_token"
$url/iaas/api/block-devices/$block_device_id?apiVersion=$api_version | jq "."
```

8 (Optional) Retrieve all the FCD block device types.

```
curl -X GET -H 'Content-Type: application/json'
-H "Authorization: Bearer $access_token" "$url/iaas/api/block-devices?
%24filter=customProperties.diskType%20eq%20firstClass&apiVersion=$api_version" | jq "."
```

9 (Optional) Delete the FCD block device.

```
curl -X DELETE -H 'Content-Type: application/json' -H "Authorization: Bearer
$access_token" $url/iaas/api/block-devices/$block_device_id?apiVersion=$api_version | jq
"."
```

Example: Create a First Class Disk

With constraints from a vSphere Storage Profile for FCD storage, use the block device specification to deploy a First Class Disk for a project ID with a two GB capacity and persistence set to false.

```
$ url='https://appliance.domain.com'
$ api_version='2021-07-15'
$ project_id='f5357a28-df59-47e0-b983-8a562910d0be'
$ capacity_in_gb=2
$ persistent=false
```

Deploy the FCD.

```
$ curl -X POST \
  $url/iaas/api/block-devices?apiVersion=$api_version \
  -H 'Content-Type: application/json' \
  -H "Authorization: Bearer $access_token" \
  -d '{
    "projectId": "'$project_id'",
    "name": "FCD-example",
    "capacityInGB": "'$capacity_in_gb'",
    "persistent" : "'$persistent'",
    "constraints": [
        {
            "mandatory": "true",
            "expression": "type:fcd"
        }
    ]
}' | jq "."
```

The response provides a selfLink to the request.

```
{
  "progress": 0,
  "status": "INPROGRESS",
  "name": "Provisioning",
  "id": "86707da6-d5d6-4ebc-94a2-0a22f3fcb794",
  "selfLink": "/iaas/api/request-tracker/86707da6-d5d6-4ebc-94a2-0a22f3fcb794"
}
```

Assign the selfLink ID variable.

```
$ selfLink_id='86707da6-d5d6-4ebc-94a2-0a22f3fcb794'
```

Track the progress of the request.

```
$ curl -X GET -H 'Content-Type: application/json' -H "Authorization: Bearer $access_token"
$url/iaas/api/request-tracker/$selfLink_id?apiVersion=$api_version | jq "."
```

After the request completes successfully, the response provides the block device ID.

```
{
  "progress": 100,
  "message": "success",
  "status": "FINISHED",
  "resources": [
    "/iaas/api/block-devices/e1cbc8e1-76bb-4bef-8e51-a582437266c2"
  ],
  "name": "Provisioning",
  "id": "86707da6-d5d6-4ebc-94a2-0a22f3fcb794",
  "selfLink": "/iaas/api/request-tracker/86707da6-d5d6-4ebc-94a2-0a22f3fcb794"
}
```

Assign the block device ID variable.

```
$ block_device_id='e1cbc8e1-76bb-4bef-8e51-a582437266c2'
```

What to do next

Use the block device ID to attach your FCD to a VM and manage your FCD snapshots. See Attach a First Class Disk and Manage First Class Disk Snapshots.

Attach a First Class Disk

To attach a First Class Disk (FCD) to a VM, you make a POST request with the machine ID of the VM. The request body includes the block device ID that you obtained from creating the FCD.

Prerequisites

- Verify that all general prerequisites and prerequisites for the Automation Assembler Infrastructure as a Service (IaaS) service have been satisfied. See Chapter 3 Prerequisites for API Use Case Examples.
- Verify that you have created an FCD and you have a block device ID. See Create a First Class Disk.
- Verify that the hardware version of the machine to which you plan to attach the FCD is vmx-13 or later.

Procedure

1 Assign the block device ID variable.

```
block_device_id='<your_block_device_id>'
```

2 Get a list of machines.

```
curl -X GET -H 'Content-Type: application/json' -H "Authorization: Bearer $access_token"
$url/iaas/api/machines?apiVersion=$api_version | jq "."
```

3 Examine the response to find the machine that you want to attach the FCD to.

4 Assign a machine ID.

```
machine_id='<your_machine_id>'
```

5 Attach the FCD to the machine.

```
curl -X POST \
  $url/iaas/api/machines/$machine_id/disks?apiVersion=$api_version \
  -H 'Content-Type: application/json' \
  -H "Authorization: Bearer $access_token" \
  -d '{
    "blockDeviceId": "'$block_device_id'"
}' | jq "."
```

The response includes a selfLink value.

```
{
  "progress": 0,
  "status": "INPROGRESS",
  "name": "Provisioning",
  "id": "example-selfLink-alphanumeric-string",
  "selfLink": "/iaas/api/request-tracker/example-selfLink-alphanumeric-string"
}
```

6 Assign the selfLink variable.

```
selfLink_id='example-selfLink-alphanumeric-string'
```

7 Use the selfLink to track the progress of the FCD attachment.

```
curl -X GET -H 'Content-Type: application/json' -H "Authorization: Bearer $access_token"
$url/iaas/api/request-tracker/$selfLink_id?apiVersion=$api_version | jq "."
```

Once complete, the response includes a list of resources with a machine that has your machine ID in the path.

```
{
  "progress": 100,
  "message": "success",
  "status": "FINISHED",
  "resources": [
    "/iaas/api/machines/your-machine-id"
  ],
  ...
}
```

8 (Optional) Detach the FCD.

```
curl -X DELETE -H 'Content-Type: application/json' -H
"Authorization: Bearer $access_token" $url/iaas/api/machines/$machine_id/disks/
$block_device_id?apiVersion=$api_version | jq "."
```

Example: Attach a First Class Disk

With the block device ID from the FCD, attach the FCD to a VM.

```
$ url='https://appliance.domain.com'
$ api_version='2021-07-15'
$ block_device_id='e1cbc8e1-76bb-4bef-8e51-a582437266c2'
```

Get a list of machines.

```
$ curl -X GET -H 'Content-Type: application/json' -H "Authorization: Bearer $access_token"
$url/iaas/api/machines?apiVersion=$api_version | jq "."
```

Examine the response. Identify the machine that you want to attach the FCD to.

```
...
    {
      "powerState": "ON",
      "externalRegionId": "Datacenter:datacenter-3",
      "cloudAccountIds": [
        "683c647b-413d-4673-a236-08b3694cd652"
      ],
      "provisioningStatus": "READY",
      "customProperties": {
        "osType": "LINUX",
        "vcUuid": "8d6dabbb-46b4-41b2-b76e-7745330f8f7d",
        "memoryGB": "0",
        "datacenter": "Datacenter:datacenter-3",
        "instanceUUID": "502a55ea-580c-9ad0-4275-82f96d3a4683",
```

```
        "softwareName": "Red Hat Enterprise Linux 7 (64-bit)",
        "cpuCount": "1",
        "memoryInMB": "256"
      },
      "externalId": "502a55ea-580c-9ad0-4275-82f96d3a4683",
      "name": "Cloud_vSphere_Machine_1-mcm100156-139639218287",
      "id": "fcaad107-48c3-320f-989f-31b0c8d4a6a0",
      "createdAt": "2022-04-02",
      "updatedAt": "2022-04-02",
...
```

Assign the machine ID variable.

```
$ machine_id='fcaad107-48c3-320f-989f-31b0c8d4a6a0'
```

Attach the FCD to the machine.

```
$ curl -X POST \
  $url/iaas/api/machines/$machine_id/disks?apiVersion=$api_version \
  -H 'Content-Type: application/json' \
  -H "Authorization: Bearer $access_token" \
  -d '{
    "blockDeviceId": "'$block_device_id'"
}' | jq "."
```

The response provides a selfLink to the request.

```
{
  "progress": 0,
  "status": "INPROGRESS",
  "name": "Provisioning",
  "id": "18050d7d-e3b2-4dd0-b0a0-5883ec766999",
  "selfLink": "/iaas/api/request-tracker/18050d7d-e3b2-4dd0-b0a0-5883ec766999"
}
```

Assign the selfLink ID variable.

```
$ selfLink_id='18050d7d-e3b2-4dd0-b0a0-5883ec766999'
```

Track the progress of the request.

```
$ curl -X GET -H 'Content-Type: application/json' -H "Authorization: Bearer $access_token"
$url/iaas/api/request-tracker/$selfLink_id?apiVersion=$api_version | jq "."
```

After the request completes successfully, the response includes your machine ID.

```
{
  "progress": 100,
  "message": "success",
  "status": "FINISHED",
  "resources": [
    "/iaas/api/machines/fcaad107-48c3-320f-989f-31b0c8d4a6a0"
```

```
  ],
  ...
}
```

Manage First Class Disk Snapshots

To create a snapshot of a First Class Disk (FCD), you make a POST request with the block device ID of the FCD. Using the snapshot ID created, you can revert an FCD to a snapshot or delete a snapshot of an FCD.

Prerequisites

- Verify that all general prerequisites and prerequisites for the Automation Assembler Infrastructure as a Service (IaaS) service have been satisfied. See Chapter 3 Prerequisites for API Use Case Examples.
- Verify that you have created an FCD and you have a block device ID. See Create a First Class Disk.

Procedure

1 Assign the block device ID variable.

```
block_device_id='<your_block_device_id>'
```

2 Create a snapshot of the FCD.

```
curl -X POST \
  $url/iaas/api/block-devices/$block_device_id/operations/snapshots?
apiVersion=$api_version \
  -H 'Content-Type: application/json' \
  -H "Authorization: Bearer $access_token" \
  -d '{
    "description": "example description"
  }' | jq "."
```

The response includes a selfLink value.

```
{
  "progress": 0,
  "status": "INPROGRESS",
  "name": "Provisioning",
  "id": "example-selfLink-alphanumeric-string",
  "selfLink": "/iaas/api/request-tracker/example-selfLink-alphanumeric-string"
}
```

3 Assign the selfLink variable.

```
selfLink_id='example-selfLink-alphanumeric-string'
```

4 Use the selfLink to track the progress of the FCD snapshot creation.

```
curl -X GET -H 'Content-Type: application/json' -H "Authorization: Bearer $access_token"
$url/iaas/api/request-tracker/$selfLink_id?apiVersion=$api_version | jq "."
```

The response indicates if the snapshot is successful.

```
{
  "progress": 100,
  "message": "success",
  "status": "FINISHED",
  "resources": [
    "/iaas/api/block-devices/your-block-device-id"
  ],
  ...
}
```

5 (Optional) To create additional FCD snapshots, repeat Step 2 to Step 4.

6 To get a snapshot ID, list all FCD snapshots.

```
curl -X GET -H 'Content-Type: application/json' -H "Authorization: Bearer $access_token"
$url/iaas/api/block-devices/$block_device_id/snapshots?apiVersion=$api_version | jq "."
```

If you created multiple snapshots, the response lists multiple snapshot IDs.

7 Examine the response and select a snapshot ID to assign as a variable.

```
snapshot_id=<your_snapshot_id_1>
```

8 (Optional) You can list an individual snapshot.

```
curl -X GET -H 'Content-Type: application/json' -H "Authorization:
Bearer $access_token" $url/iaas/api/block-devices/$block_device_id/snapshots/$snapshot_id?
apiVersion=$api_version | jq "."
```

9 (Optional) You can revert an FCD to a snapshot.

```
curl -X POST \
  $url/iaas/api/block-devices/$block_device_id/operations/revert?
id=$snapshot_id\&apiVersion=$api_version \
 -H 'Content-Type: application/json' \
 -H "Authorization: Bearer $access_token" \
 -d '' | jq "."
```

To track the progress of the reversion, perform Step 3 and Step 4.

10 (Optional) You can delete a snapshot.

```
curl -X DELETE -H 'Content-Type: application/json' -H "Authorization:
Bearer $access_token" $url/iaas/api/block-devices/$block_device_id/snapshots/$snapshot_id?
apiVersion=$api_version | jq "."
```

Example: Create snapshots of a First Class Disk and revert an FCD to a snapshot

With the block device ID from the created FCD, create multiple snapshots of an FCD.

```
$ url='https://appliance.domain.com'
$ api_version='2021-07-15'
$ block_device_id='e1cbc8e1-76bb-4bef-8e51-a582437266c2'
```

Create a snapshot of the FCD.

```
$ curl -X POST \
  $url/iaas/api/block-devices/$block_device_id/operations/snapshots?apiVersion=$api_version \
  -H 'Content-Type: application/json' \
  -H "Authorization: Bearer $access_token" \
  -d '{
    "description": "Example description 1"
  }' | jq "."
```

The response provides a selfLink to the request.

```
{
  "progress": 0,
  "status": "INPROGRESS",
  "name": "Provisioning",
  "id": "d08bb46c-cf7e-40b6-bdf8-893390ba4d51",
  "selfLink": "/iaas/api/request-tracker/d08bb46c-cf7e-40b6-bdf8-893390ba4d51"
}
```

Assign the selfLink ID variable.

```
$ selfLink_id='d08bb46c-cf7e-40b6-bdf8-893390ba4d51'
```

Track the progress of the request.

```
$ curl -X GET -H 'Content-Type: application/json' -H "Authorization: Bearer $access_token"
$url/iaas/api/request-tracker/$selfLink_id?apiVersion=$api_version | jq "."
```

Create a second snapshot of the FCD.

```
$ curl -X POST \
  $url/iaas/api/block-devices/$block_device_id/operations/snapshots?apiVersion=$api_version \
  -H 'Content-Type: application/json' \
  -H "Authorization: Bearer $access_token" \
  -d '{
    "description": "Example description 2"
  }' | jq "."
```

List all the snapshots of the FCD.

```
curl -X GET -H 'Content-Type: application/json' -H "Authorization: Bearer $access_token" $url/
iaas/api/block-devices/$block_device_id/snapshots?apiVersion=$api_version | jq "."
```

Examine the response to see all snapshot IDs.


```
[
  {
    "name": "357ed3e5-8b2e-4533-b6fe-3ea6e15b8de5",
    "desc": "Example description 1",
    "isCurrent": false,
    "id": "16cfdbb8-559c-49ff-8162-0a4c57079c81",
    "createdAt": "2022-04-02",
    "updatedAt": "2022-04-02",
    "owner": "user@mycompany.com",
    "organizationId": "b373cda4-ae0f-4d5a-9eca-f307bd30c9cd",
    "orgId": "b373cda4-ae0f-4d5a-9eca-f307bd30c9cd",
    "_links": {
      "self": {
        "href": "/iaas/api/block-devices/e1cbc8e1-76bb-4bef-8e51-a582437266c2/snapshots/
16cfdbb8-559c-49ff-8162-0a4c57079c81"
      }
    }
  },
  {
    "name": "b04f7513-c695-4662-b5e8-a023a7b1bfe7",
    "desc": "Example description 2",
    "isCurrent": true,
    "id": "ed1b09ff-1175-4cdd-b07e-7bb906a9ddc4",
    "createdAt": "2022-04-02",
    "updatedAt": "2022-04-02",
    "owner": "user@mycompany.com",
    "organizationId": "b373cda4-ae0f-4d5a-9eca-f307bd30c9cd",
    "orgId": "b373cda4-ae0f-4d5a-9eca-f307bd30c9cd",
    "_links": {
      "self": {
        "href": "/iaas/api/block-devices/e1cbc8e1-76bb-4bef-8e51-a582437266c2/snapshots/
ed1b09ff-1175-4cdd-b07e-7bb906a9ddc4"
      }
    }
  }
]
```


Assign a snapshot ID variable.

```
snapshot_id='16cfdbb8-559c-49ff-8162-0a4c57079c81'
```

List information about the snapshot.

```
$ curl -X GET -H 'Content-Type: application/json' -H "Authorization: Bearer $access_token"
$url/iaas/api/block-devices/$block_device_id/snapshots/$snapshot_id?apiVersion=$api_version |
jq "."
```

The response shows information about the single snapshot.


```
{
    "name": "357ed3e5-8b2e-4533-b6fe-3ea6e15b8de5",
    "desc": "Example description 1",
```

```
    "isCurrent": false,
    "id": "16cfdbb8-559c-49ff-8162-0a4c57079c81",
    "createdAt": "2022-04-02",
    "updatedAt": "2022-04-02",
    "owner": "user@mycompany.com",
    "organizationId": "b373cda4-ae0f-4d5a-9eca-f307bd30c9cd",
    "orgId": "b373cda4-ae0f-4d5a-9eca-f307bd30c9cd",
    "_links": {
      "self": {
        "href": "/iaas/api/block-devices/e1cbc8e1-76bb-4bef-8e51-a582437266c2/snapshots/
16cfdbb8-559c-49ff-8162-0a4c57079c81"
      }
    }
```

Revert the FCD to the snapshot.

```
$ curl -X GET -H 'Content-Type: application/json' -H "Authorization: Bearer $access_token"
$url/iaas/api/block-devices/$block_device_id/snapshots/$snapshot_id?apiVersion=$api_version |
jq "."
```

To validate the reversion, list information about the snapshot again.

```
$ curl -X GET -H 'Content-Type: application/json' -H "Authorization: Bearer $access_token"
$url/iaas/api/block-devices/$block_device_id/snapshots/$snapshot_id?apiVersion=$api_version |
jq "."
```

In the response, `"isCurrent":true` shows that the FCD has reverted to the snapshot.

```
{
    "name": "357ed3e5-8b2e-4533-b6fe-3ea6e15b8de5",
    "desc": "Example description 1",
    "isCurrent": true,
    "id": "16cfdbb8-559c-49ff-8162-0a4c57079c81",
    "createdAt": "2022-04-02",
    "updatedAt": "2022-04-02",
    "owner": "user@mycompany.com",
    "organizationId": "b373cda4-ae0f-4d5a-9eca-f307bd30c9cd",
    "orgId": "b373cda4-ae0f-4d5a-9eca-f307bd30c9cd",
    "_links": {
      "self": {
        "href": "/iaas/api/block-devices/e1cbc8e1-76bb-4bef-8e51-a582437266c2/snapshots/
16cfdbb8-559c-49ff-8162-0a4c57079c81"
      }
    }
```

Working with Azure Disk Snapshots

You can use the Automation Assembler IaaS API to create or delete snapshots of Azure managed disks. The snapshot provides a backup of your block device.

Note If you create a snapshot of an independent disk and then delete the disk, the snapshot is left behind. So before deleting a disk from the Azure cloud account, check for snapshots. If snapshots exist, delete them first before deleting the disk.

Create a Block Device

If you do not already have an Azure managed disk, create a block device to use for your snapshot. To create a block device, you make a POST request using the block device specification. The request body includes a project ID, disk capacity, persistence setting, and constraints from the Azure Storage Profile for a managed disk.

Prerequisites

- Verify that all general prerequisites and prerequisites for the Automation Assembler Infrastructure as a Service (IaaS) service have been satisfied. See Chapter 3 Prerequisites for API Use Case Examples.
- Verify that you have added a project and you have the project ID. See Create a Project to use in Automation Assembler.
- Know the capacity of the disk and the persistence of the disk that you are creating.
- Verify that you have created a storage profile for a managed disk and that you have the `defaultItem` and the tags from the response. See Create a Microsoft Azure Storage Profile for a Managed Disk.

Procedure

1 Assign the project ID variable.

```
project_id='<your_project_id>'
```

2 Set the capacity and persistence for the disk.

```
capacity_in_gb=<integer>
persistence=<true|false>
```

3 Deploy the block device.

   - With `mandatory` set to false, tags in the `expression` are not required to match tags in an existing storage profile for the deployment to succeed. However if tags are provided, Automation will try to match them when deploying the block device.

- The `expression` is the key:value tag pair used to create the storage profile. See Create a vSphere Storage Profile for a First Class Disk.

```
curl -X POST \
  $url/iaas/api/block-devices?apiVersion=$api_version \
  -H 'Content-Type: application/json' \
  -H "Authorization: Bearer $access_token" \
  -d '{
    "projectId": "'$project_id'",
    "name": "block-device-example",
    "capacityInGB": "'$capacity_in_gb'",
    "persistent" : "'$persistent'",
    "constraints": [
        {
            "mandatory": "false",
            "expression": "type:managed"
        }
    ]
}' | jq "."
```

The response includes a selfLink value.

```
{
  "progress": 0,
  "status": "INPROGRESS",
  "name": "Provisioning",
  "id": "example-selfLink-alphanumeric-string",
  "selfLink": "/iaas/api/request-tracker/example-selfLink-alphanumeric-string"
}
```

4 Assign the selfLink variable.

```
selfLink_id='example-selfLink-alphanumeric-string'
```

5 Use the selfLink variable to track the progress of the block device creation.

```
curl -X GET -H 'Content-Type: application/json' -H "Authorization: Bearer $access_token"
$url/iaas/api/request-tracker/$selfLink_id?apiVersion=$api_version | jq "."
```

In the list of resources, the response includes block devices with the block device ID in the path.

```
{
  "progress": 100,
  "message": "success",
  "status": "FINISHED",
  "resources": [
    "/iaas/api/block-devices/example-blockdevice-alphanumeric-string"
  ],
  ...
}
```

6 (Optional) If you want to retrieve the ID of an existing block device, use an `OData` filter with the block device name in the request.

```
block_device_name='<your_block_device_name>'
curl -X GET -H 'Content-Type: application/json'
-H "Authorization: Bearer $access_token" "$url/iaas/api/block-devices?
apiVersion=$api_version&"'$filter'"=name%20eq%20$block_device_name" | jq "."
```

7 Assign the block device ID variable.

```
block_device_id='example-blockdevice-alphanumeric-string'
```

Example: Create a Block Device

With constraints from an Azure Storage Profile for a managed disk, use the block device specification to deploy a managed disk for a project ID with a two GB capacity and persistence set to false.

```
$ url='https://appliance.domain.com'
$ api_version='2021-07-15'
$ project_id='f5357a28-df59-47e0-b983-8a562910d0be'
$ capacity_in_gb=2
$ persistent=false
```

Deploy the block device.

```
$ curl -X POST \
  $url/iaas/api/block-devices?apiVersion=$api_version \
  -H 'Content-Type: application/json' \
  -H "Authorization: Bearer $access_token" \
  -d '{
    "projectId": "'$project_id'",
    "name": "block-device-example",
    "capacityInGB": "'$capacity_in_gb'",
    "persistent" : "'$persistent'",
    "constraints": [
        {
            "mandatory": "false",
            "expression": "type:managed"
        }
    ]
}' | jq "."
```

The response provides a selfLink to the request.

```
{
  "progress": 0,
  "status": "INPROGRESS",
  "name": "Provisioning",
  "id": "22bdaf20-ce48-4a9f-8c1f-f4e74263645f",
  "selfLink": "/iaas/api/request-tracker/22bdaf20-ce48-4a9f-8c1f-f4e74263645f",
  "deploymentId":"cf33c90e-6f6d-48ed-82dd-a6a9f0e6f700"
}
```

Assign the selfLink ID variable.

```
$ selfLink_id='22bdaf20-ce48-4a9f-8c1f-f4e74263645f'
```

Track the progress of the request.

```
$ curl -X GET -H 'Content-Type: application/json' -H "Authorization: Bearer $access_token"
$url/iaas/api/request-tracker/$selfLink_id?apiVersion=$api_version | jq "."
```

After the request completes successfully, the response provides the block device ID.

```
{
  "progress": 100,
  "message": "success",
  "status": "FINISHED",
  "resources": [
    "/iaas/api/block-devices/41d600c2-429e-4c90-98d4-638e77724101"
  ],
  "name": "Provisioning",
  "id": "22bdaf20-ce48-4a9f-8c1f-f4e74263645f",
  "selfLink": "/iaas/api/request-tracker/22bdaf20-ce48-4a9f-8c1f-f4e74263645f",
  "deploymentId":"cf33c90e-6f6d-48ed-82dd-a6a9f0e6f700"
}
```

Assign the block device ID variable.

```
$ block_device_id='41d600c2-429e-4c90-98d4-638e77724101'
```

What to do next

Use the block device ID to create a snapshot. See Create and Manage Azure Disk Snapshots.

Create and Manage Azure Disk Snapshots

To create a snapshot of a Azure managed disk, you make a POST request with the block device ID of the managed disk. Using the snapshot ID created, you can list a snapshot or delete a snapshot of a managed disk.

Prerequisites

- Verify that all general prerequisites and prerequisites for the Automation Assembler Infrastructure as a Service (IaaS) service have been satisfied. See Chapter 3 Prerequisites for API Use Case Examples.
- Verify that you have an Azure managed disk and a block device ID. See Create a Block Device.

Procedure

1 Assign the block device ID variable.

```
block_device_id='<your_block_device_id>'
```

2 Create a snapshot of the managed disk.

```
curl -X POST \
  $url/iaas/api/block-devices/$block_device_id/operations/snapshots?
apiVersion=$api_version \
  -H 'Content-Type: application/json' \
  -H "Authorization: Bearer $access_token" \
  -d '{
    "name": "example_name"
  }' | jq "."
```

In addition to the required snapshot name, you can include optional snapshot properties that can be used independently or combined as in the following example.

```
curl -X POST \
  $url/iaas/api/block-devices/$block_device_id/operations/snapshots?
apiVersion=$api_version \
  -H 'Content-Type: application/json' \
  -H "Authorization: Bearer $access_token" \
  -d '{
    "name": "example_name"
    "snapshotProperties": {
       "incremental": "true",
       "resourceGroupName": "newRG",
       "encryptionSetId" : "'"$encryption_key"'"
 },
  }' | jq "."
```

- To create a snapshot that only consists of changes since the last snapshot, set `incremental` to true.
- To specify a target resource group for the snapshot, include the `resourceGroupName` property. You can look up existing resource group names in the Azure portal, or specify a name for a new resource group, If not specified, Automation creates the snapshot in the same resource group as the block device by default.
- To encrypt the snapshot with a customer-managed key, include the `encryptionSetId` property.

  To get the `encryptionSetId` property for a particular region ID, use the following API request:

  ```
  curl -X GET -H 'Content-Type: application/json' -H
  "Authorization: Bearer $access_token" $url/iaas/api/fabric-azure-disk-encryption-sets?
  regionId=$region_id&apiVersion=$api_version | jq "."
  ```

Examine the response for the `id` as in the following example.

```
    {
      "name": "test-encryption-1.1",
      "id": "/subscriptions/b8ef63a7-a5e3-44fa-8745-1ead33fa1f25/resourceGroups/
DISKRG67970/providers/Microsoft.Compute/diskEncryptionSets/test-encryption-1.1",
      "regionId": "eastus",
      "key": "key1234",
      "vault": "KeyVault1234"
    },
```

Assign the encryption key variable.

```
$encryption_key='/subscriptions/b8ef63a7-a5e3-44fa-8745-1ead33fa1f25/resourceGroups/
DISKRG67970/providers/Microsoft.Compute/diskEncryptionSets/test-encryption-1.1'
```

3 Examine the response for the selfLink value.

```
{
  "progress": 0,
  "status": "INPROGRESS",
  "name": "Provisioning",
  "id": "example-selfLink-alphanumeric-string",
  "selfLink": "/iaas/api/request-tracker/example-selfLink-alphanumeric-string"
}
```

4 Assign the selfLink variable.

```
selfLink_id='example-selfLink-alphanumeric-string'
```

5 Use the selfLink to track the progress of the snapshot creation.

```
curl -X GET -H 'Content-Type: application/json' -H "Authorization: Bearer $access_token"
$url/iaas/api/request-tracker/$selfLink_id?apiVersion=$api_version | jq "."
```

The response indicates if the snapshot is successful.

```
{
  "progress": 100,
  "message": "success",
  "status": "FINISHED",
  "resources": [
    "/iaas/api/block-devices/your-block-device-id"
  ],
  ...
}
```

6 To get a snapshot ID, list all snapshots.

```
curl -X GET -H 'Content-Type: application/json' -H "Authorization: Bearer $access_token"
$url/iaas/api/block-devices/$block_device_id/snapshots?apiVersion=$api_version | jq "."
```

If you created multiple snapshots, the response lists multiple snapshot IDs.

7 Examine the response and select a snapshot ID to assign as a variable.

```
snapshot_id=<your_snapshot_id_1>
```

8 (Optional) You can list an individual snapshot.

```
curl -X GET -H 'Content-Type: application/json' -H "Authorization:
Bearer $access_token" $url/iaas/api/block-devices/$block_device_id/snapshots/$snapshot_id?
apiVersion=$api_version | jq "."
```

9 (Optional) You can delete a snapshot.

```
curl -X DELETE -H 'Content-Type: application/json' -H "Authorization:
Bearer $access_token" $url/iaas/api/block-devices/$block_device_id/snapshots/$snapshot_id?
apiVersion=$api_version | jq "."
```

Example: Create snapshots of a managed disk

With the block device ID, create multiple snapshots of a managed disk.

```
$ url='https://appliance.domain.com'
$ api_version='2021-07-15'
$ block_device_id='41d600c2-429e-4c90-98d4-638e77724101'
```

Create a snapshot of the managed disk.

```
$ curl -X POST \
  $url/iaas/api/block-devices/$block_device_id/operations/snapshots?apiVersion=$api_version \
  -H 'Content-Type: application/json' \
  -H "Authorization: Bearer $access_token" \
  -d '{
    "name": "demo-snapshot-1"
  }' | jq "."
```

The response provides a selfLink to the request.

```
{
  "progress": 0,
  "status": "INPROGRESS",
  "name": "Provisioning",
  "id": "66123d15-8e5a-42b0-a0b4-e9ed8e21180a",
  "selfLink": "/iaas/api/request-tracker/66123d15-8e5a-42b0-a0b4-e9ed8e21180a"
}
```

Assign the selfLink ID variable.

```
$ selfLink_id='66123d15-8e5a-42b0-a0b4-e9ed8e21180a'
```

Track the progress of the request.

```
$ curl -X GET -H 'Content-Type: application/json' -H "Authorization: Bearer $access_token"
$url/iaas/api/request-tracker/$selfLink_id?apiVersion=$api_version | jq "."
```

Create a second snapshot of the managed disk.

```
$ curl -X POST \
  $url/iaas/api/block-devices/$block_device_id/operations/snapshots?apiVersion=$api_version \
  -H 'Content-Type: application/json' \
  -H "Authorization: Bearer $access_token" \
  -d '{
    "name": "demo-snapshot-2"
  }' | jq "."
```

List all the snapshots of the managed disk.

```
$ curl -X GET -H 'Content-Type: application/json' -H "Authorization: Bearer $access_token"
$url/iaas/api/block-devices/$block_device_id/snapshots?apiVersion=$api_version | jq "."
```

Examine the response to see all snapshot IDs.

```
[
  {
    "name": "demo-snapshot-1",
    "snapshotProperties": {
      "incremental": "false"
    },
    "id": "38348991-00a1-48be-80d6-00d62afcd280",
    "createdAt": "2022-04-02",
    "updatedAt": "2022-04-02",
    "owner": "user@mycompany.com",
    "organizationId": "1b6fd77b-f5d9-466b-88d3-97c0d9eb70c9",
    "orgId": "1b6fd77b-f5d9-466b-88d3-97c0d9eb70c9",
    "_links": {
      "self": {
        "href": "/iaas/api/block-devices/41d600c2-429e-4c90-98d4-638e77724101/snapshots/
38348991-00a1-48be-80d6-00d62afcd280"
      }
    }
  },
  {
    "name": "demo-snapshot-2",
    "snapshotProperties": {
      "incremental": "false"
    },
    "id": "80407f78-7f90-4d9f-83f4-10d3a1e982ac",
    "createdAt": "2022-04-02",
    "updatedAt": "2022-04-02",
    "owner": "user@mycompany.com",
    "organizationId": "1b6fd77b-f5d9-466b-88d3-97c0d9eb70c9",
    "orgId": "1b6fd77b-f5d9-466b-88d3-97c0d9eb70c9",
    "_links": {
```

```
      "self": {
        "href": "/iaas/api/block-devices/41d600c2-429e-4c90-98d4-638e77724101/snapshots/
80407f78-7f90-4d9f-83f4-10d3a1e982ac"
      }
    }
  }
]
```

Assign a snapshot ID variable.

```
$ snapshot_id='80407f78-7f90-4d9f-83f4-10d3a1e982ac'
```

List information about the snapshot.

```
$ curl -X GET -H 'Content-Type: application/json' -H "Authorization: Bearer $access_token"
$url/iaas/api/block-devices/$block_device_id/snapshots/$snapshot_id?apiVersion=$api_version |
jq "."
```

The response shows information about the single snapshot.

```
{
  "name": "demo-snapshot-2",
  "snapshotProperties": {
    "incremental": "false"
  },
  "id": "80407f78-7f90-4d9f-83f4-10d3a1e982ac",
  "createdAt": "2022-04-02",
  "updatedAt": "2022-04-02",
  "owner": "user@mycompany.com",
  "organizationId": "1b6fd77b-f5d9-466b-88d3-97c0d9eb70c9",
  "orgId": "1b6fd77b-f5d9-466b-88d3-97c0d9eb70c9",
  "_links": {
    "self": {
      "href": "/iaas/api/block-devices/41d600c2-429e-4c90-98d4-638e77724101/snapshots/
80407f78-7f90-4d9f-83f4-10d3a1e982ac"
    }
  }
}
```

Update the Custom Properties of a Machine

After deploying a machine, you can use the IaaS APIs to update the machine with custom properties. Custom properties provide you with the flexibility to add any information about the machine that you want.

For example, machine IDs are typically autogenerated. By updating the custom properties, you can identify the machine owner and include their contact email or phone information.

Prerequisites

- Verify that all general prerequisites and prerequisites for the Automation Assembler Infrastructure as a Service (IaaS) service have been satisfied. See Chapter 3 Prerequisites for API Use Case Examples.

- Verify that you have the ID of the virtual machine in your deployment. See Get Deployment Resource IDs.

Procedure

1 Assign your virtual machine ID variable.

Assigning this variable is useful if you plan to update the machine again.

```
virtual_machine_id='<your_virtual_machine_id>'
```

2 Update the machine with custom property names and values that you choose.

```
curl -X PATCH \
  $url/iaas/api/machines/$virtual_machine_id?apiVersion=$api_version \
  -H "Authorization: Bearer $access_token" \
  -H 'Content-Type: application/json' \
  -d '{
  "customProperties": {
    "additionalPropName1": "<custom_prop_value_1>",
    "additionalPropName2": "<custom_prop_value_2>",
    "additionalPropName3": "<custom_prop_value_3>"
  },
  "description": "string",
  "tags": "[ { \"key\" : \"ownedBy\", \"value\": \"Rainpole\" } ]"
}' | jq "."
```

3 A snippet of the response lists the added custom properties.

Example: Add a Custom Properties to Your Virtual Machine

Update the virtual machine with resource ID **42f49781-1490-4a08-ae21-8baf383a72ac** by adding custom properties.

Assign variables.

```
$ url='https://appliance.domain.com'
$ api_version='2021-07-15'
```

Assign the virtual machine ID.

```
$ virtual_machine_id='42f49781-1490-4a08-ae21-8baf383a72ac'
```

Update the machine with custom properties.

```
$ curl -X PATCH \
  $url/iaas/api/machines/$virtual_machine_id?apiVersion=$api_version \
  -H "Authorization: Bearer $access_token" \
  -H 'Content-Type: application/json' \
  -d '{
  "customProperties": {
    "ownerName": "VMuser_Example",
    "ownerEmail": "VMuser_Example@mycompany.com",
    "ownerCell": "123.456.7890"
```

```
  },
  "description": "string",
  "tags": "[ { \"key\" : \"my.enumeration.type\", \"value\": \"ec2_instance\" } ]"
}' | jq "."
```

A snippet of the response shows that the request was successful.

```
...
  "customProperties": {
       "ownerName": "VMuser_Example",
       "ownerEmail": "VMuser_Example@mycompany.com",
       "ownerCell": "123.456.7890"
       "image": "ubuntu",
       "OStype": "LINUX",
       "imageId": "ami-b1234cc5",
  ...
  },
...
```

Provision a VLAN Private Network

After creating a network profile with a VLAN transport zone, you can provision a VLAN private network.

With VLAN IDs in your private network definition, Automation Assembler allocates the cloud template to a network profile with a VLAN transport zone specified. If no VLAN ID is specified, the cloud template is allocated to a network profile with an overlay transport zone.

Prerequisites

- Verify that all general prerequisites and prerequisites for the Automation Assembler Infrastructure as a Service (IaaS) service have been satisfied. See Chapter 3 Prerequisites for API Use Case Examples.
- Verify that you have the project ID. See Create a Project to use in Automation Assembler.
- Verify that you know the VLAN IDs that you want to use. Comma separated values can be 0 - 4094.

Procedure

1 Assign your project ID variable.

```
project_id='<your_project_id>'
```

2 Provision the network with VLAN IDs that you choose.

```
curl -X POST \
  $url/iaas/api/networks?apiVersion=$api_version \
  -H "Authorization: Bearer $access_token" \
  -H 'Content-Type: application/json' \
  -d '{
    "name": "<your_network_name>",
```

```
    "description": "<your_description>",
    "projectId": "'$project_id'",
    "customProperties": {
      "networkType": "PRIVATE",
      "vlanIds": "<integer_values_0_to_4094>"
  }
}' | jq "."
```

The response includes a selfLink value.

```
{
  "progress": 0,
  "status": "INPROGRESS",
  "name": "Provisioning",
  "id": "example-selfLink-alphanumeric-string",
  "selfLink": "/iaas/api/request-tracker/example-selfLink-alphanumeric-string"
}
```

3 Assign the selfLink variable.

```
selfLink_id='example-selfLink-alphanumeric-string'
```

4 Use the selfLink variable to track the request.

```
curl -X GET -H 'Content-Type: application/json' -H "Authorization: Bearer $access_token"
"$url/iaas/api/request-tracker/$selfLink_id | jq "."
```

Example: Provision a VLAN Private Network

Provision a VLAN private network for a project ID

Assign variables.

```
$ url='https://appliance.domain.com'
$ api_version='2021-07-15'
```

Assign the project ID.

```
$ project_id='5944aacb-91de-4541-bb9e-ef2a5403f81b'
```

Provision the VLAN private network.

```
curl -X POST \
  $url/iaas/api/networks?apiVersion=$api_version \
  -H "Authorization: Bearer $access_token" \
  -H 'Content-Type: application/json' \
  -d '{
    "name": "example-vlan-network",
    "description": "Example VLAN Network created using API",
    "projectId": "'$project_id'",
    "customProperties": {
      "networkType": "PRIVATE",
      "vlanIds": "1004, 1005"
```

```
  }
}' | jq "."
```

The response includes a selfLink.

```
{
  "progress": 0,
  "status": "INPROGRESS",
  "name": "Provisioning",
  "id": "dab1fe2f-4104-4fce-b534-e7ab5c172788",
  "selfLink": "/iaas/api/request-tracker/dab1fe2f-4104-4fce-b534-e7ab5c172788",
  "deploymentId": "8964a3f6-e829-40ad-b07c-0177abb7b4f4"
}
```

Assign the selfLink variable.

```
selfLink_id='dab1fe2f-4104-4fce-b534-e7ab5c172788'
```

Use the selfLink variable to track the request.

```
curl -X GET -H 'Content-Type: application/json' -H "Authorization: Bearer $access_token"
"$url/iaas/api/request-tracker/$selfLink_id | jq "."
```

When the request completes successfully, the response shows the network has been created.

```
{
    "progress": 100,
    "message": "success",
    "status": "FINISHED",
    "resources": [
        "/iaas/api/networks/3f3b4f87-3e0d-48e8-a484-a8f2890977df"
    ],
    "name": "example-vlan-network Allocation",
    "id": "1cca9d9c-1e93-40d6-8bab-fa92dd89566a",
    "selfLink": "/iaas/api/request-tracker/1cca9d9c-1e93-40d6-8bab-fa92dd89566a",
    "deploymentId": "8b8eef00-e159-4ba9-9e78-09089a3f5786"
}
```

How do I use a placement policy to spread VMs by memory

If you want to balance your deployed resources across multiple cloud zones, you can use the IaaS API to define the placement policy in a project and its cloud zones. When you deploy a cloud template that uses the project, VMware Aria Automation allocates new VMs to zones and clusters with the most free memory, effectively spreading them for better memory usage.

Prerequisites for defining a placement policy

- Verify that all general prerequisites and prerequisites for the Automation Assembler Infrastructure as a Service (IaaS) service have been satisfied. See Chapter 3 Prerequisites for API Use Case Examples.

How to specify spread by memory in your project

This example shows how to find the project with cloud zones where you want to place your VMs. Then using the project ID, you update the project with `"placementPolicy": "SPREAD_MEMORY"` to spread VM memory use across the cloud zones.

For information about project-level placement policies, see How do project-level placement policies affect resource allocation in VMware Aria Automation .

Get all projects.

```
curl -X GET -H 'Accept: application/json' -H "Authorization: Bearer $access_token" "$url/
iaas/api/projects?apiVersion=$api_version" | jq "."
```

A snippet of the response shows a project with its assigned cloud zones. The project ID `6c2f2d0d-ecee-42e3-90be-7bb66d6da2f9` has two zones with zone IDs:

- `3c2bbe36-bf8e-4484-9c31-ce552422aaf1`
- `8992bdf0-136f-401c-822a-e22dae67259b`

```
{
    "administrators": [],
    "members": [],
    "viewers": [],
    "zones": [
        {
            "zoneId": "3c2bbe36-bf8e-4484-9c31-ce552422aaf1",
            "priority": 0,
            "maxNumberInstances": 0,
            "allocatedInstancesCount": 0,
            "memoryLimitMB": 0,
            "allocatedMemoryMB": 0,
            "cpuLimit": 0,
            "allocatedCpu": 0,
            "storageLimitGB": 0,
            "allocatedStorageGB": 0.0
        },
        {
            "zoneId": "8992bdf0-136f-401c-822a-e22dae67259b",
            "priority": 0,
            "maxNumberInstances": 0,
            "allocatedInstancesCount": 0,
            "memoryLimitMB": 0,
            "allocatedMemoryMB": 0,
            "cpuLimit": 0,
            "allocatedCpu": 0,
            "storageLimitGB": 0,
            "allocatedStorageGB": 0.0
        }
    ],
    "constraints": {},
    "operationTimeout": 0,
    "sharedResources": true,
```

```
    "placementPolicy": "DEFAULT",
    "customProperties": {},
    "name": "project1",
    "description": "",
    "id": "6c2f2d0d-ecee-42e3-90be-7bb66d6da2f9",
    "orgId": "f098d692-e980-41a5-b349-83084fce1ea0",
    "_links": {
        "self": {
            "href": "/iaas/api/projects/6c2f2d0d-ecee-42e3-90be-7bb66d6da2f9"
        }
    }
}
```

Use the project ID to update the placement policy and spread memory over the two cloud zones.

```
curl -X PATCH \
  "$url/iaas/api/projects/6c2f2d0d-ecee-42e3-90be-7bb66d6da2f9?apiVersion=$api_version"
  -H 'Content-Type: application/json'
  -H "Authorization: Bearer $access_token"
  -d '{
    "placementPolicy": "SPREAD_MEMORY"
  }' | jq "."
```

How to specify spread by memory in cloud zones

This example shows how to use the cloud zone IDs in the project ID `6c2f2d0d-ecee-42e3-90be-7bb66d6da2f9` to check free memory in compute resources. Provided that free memory is available, you update the cloud zones with `"placementPolicy": "SPREAD_MEMORY"`.

For information about placement policies in Automation Assembler cloud zones, see Learn more about Automation Assembler cloud zones.

To check available memory in zone ID `3c2bbe36-bf8e-4484-9c31-ce552422aaf1`, list compute resources.

```
curl -X GET -H 'Content-Type: application/json' -H "Authorization: Bearer $access_token"
"$url/iaas/api/zones/3c2bbe36-bf8e-4484-9c31-ce552422aaf1/computes?apiVersion=$api_version"
| jq "."
```

The response shows two clusters with available memory:

- `ESO_PKS_VC01_Cluster03` has 747 Gbytes available memory.
- `ESO_PKS_VC01_Cluster04` has 397 Gbytes available memory.

```
{
  "content": [
    {
      "externalRegionId": "Datacenter:datacenter-3",
      "tags": [],
      "type": "Cluster",
      "lifecycleState": "READY",
      "powerState": "ON",
```

```
      "customProperties": {
        "vcUuid": "74620317-856c-4f55-a862-dd2b43f07373",
        "hostCount": "2",
        "datacenter": "Datacenter:datacenter-3",
        "cpuPkgCount": "4",
        "cpuCoreCount": "56",
        "vsanConfigId": "52a60c7a-ef2e-af08-7cb2-36b06f686ebb",
        "isVsanEnabled": "true",
        "MaxVCPUperInstance": "56",
        "MemoryAvailableBytes": "747407147008"
      },
      "externalId": "domain-c21",
      "name": "ESO_PKS_VC01_Cluster03",
      "id": "e03f62e1-9a48-4d2c-8aa7-7bdb97293749",
      "createdAt": "2022-07-25",
      "updatedAt": "2022-07-26",
      "orgId": "f098d692-e980-41a5-b349-83084fce1ea0"
    },
    {
      "externalRegionId": "Datacenter:datacenter-3",
      "tags": [],
      "type": "Cluster",
      "lifecycleState": "READY",
      "powerState": "ON",
      "customProperties": {
        "vcUuid": "74620317-856c-4f55-a862-dd2b43f07373",
        "hostCount": "1",
        "datacenter": "Datacenter:datacenter-3",
        "cpuPkgCount": "2",
        "cpuCoreCount": "28",
        "vsanConfigId": "",
        "isVsanEnabled": "false",
        "MaxVCPUperInstance": "56",
        "MemoryAvailableBytes": "397091536896"
      },
      "externalId": "domain-c24",
      "name": "ESO_PKS_VC01_Cluster04",
      "id": "d2d42957-b6df-4e45-879a-93dfbec9a528",
      "createdAt": "2022-07-25",
      "updatedAt": "2022-07-26",
      "orgId": "f098d692-e980-41a5-b349-83084fce1ea0"
    }
  ],
  "totalElements": 2,
  "numberOfElements": 2
}
```


Use the zone ID `3c2bbe36-bf8e-4484-9c31-ce552422aaf1` to spread memory over the clusters.

```
curl -X PATCH \
  "$url/iaas/api/zones/3c2bbe36-bf8e-4484-9c31-ce552422aaf1?apiVersion=$api_version"
  -H 'Content-Type: application/json'
  -H "Authorization: Bearer $access_token"
  -d '{
```

```
    "placementPolicy": "SPREAD_MEMORY"
  }' | jq "."
```

To check available memory in zone ID `8992bdf0-136f-401c-822a-e22dae67259b`, list compute resources.

```
curl -X GET -H 'Content-Type: application/json' -H "Authorization: Bearer $access_token"
"$url/iaas/api/zones/8992bdf0-136f-401c-822a-e22dae67259b/computes?apiVersion=$api_version"
| jq "."
```

The response shows two clusters with available memory:

- `ESO_PKS_VC01_Cluster01` has 830 Gbytes available memory.
- `ESO_PKS_VC01_Cluster05` has 397 Gbytes available memory.

```
{
  "content": [
    {
      "externalRegionId": "Datacenter:datacenter-3",
      "tags": [],
      "type": "Cluster",
      "lifecycleState": "READY",
      "powerState": "ON",
      "customProperties": {
        "vcUuid": "74620317-856c-4f55-a862-dd2b43f07373",
        "hostCount": "5",
        "datacenter": "Datacenter:datacenter-3",
        "cpuPkgCount": "10",
        "cpuCoreCount": "140",
        "vsanConfigId": "52de077e-2f21-b24c-536b-79ef9a412968",
        "isVsanEnabled": "true",
        "MaxVCPUperInstance": "56",
        "MemoryAvailableBytes": "830677712896"
      },
      "externalId": "domain-c18",
      "name": "ESO_PKS_VC01_Cluster01",
      "id": "98691ba2-2f84-4d31-a9f1-690d254c5305",
      "createdAt": "2022-07-25",
      "updatedAt": "2022-07-26",
      "orgId": "f098d692-e980-41a5-b349-83084fce1ea0"
    },
    {
      "externalRegionId": "Datacenter:datacenter-3",
      "tags": [],
      "type": "Cluster",
      "lifecycleState": "READY",
      "powerState": "ON",
      "customProperties": {
        "vcUuid": "74620317-856c-4f55-a862-dd2b43f07373",
        "hostCount": "1",
        "datacenter": "Datacenter:datacenter-3",
        "cpuPkgCount": "2",
        "cpuCoreCount": "28",
        "vsanConfigId": "",
```

```
        "isVsanEnabled": "false",
        "MaxVCPUperInstance": "56",
        "MemoryAvailableBytes": "397045399552"
      },
      "externalId": "domain-c26",
      "name": "ESO_PKS_VC01_Cluster05",
      "id": "f576f346-ed3b-4180-acdc-3c217e9fa0fd",
      "createdAt": "2022-07-25",
      "updatedAt": "2022-07-26",
      "orgId": "f098d692-e980-41a5-b349-83084fce1ea0"
    }
  ],
  "totalElements": 2,
  "numberOfElements": 2
}
```

Use the zone ID `8992bdf0-136f-401c-822a-e22dae67259b` to spread memory over the clusters.

```
curl -X PATCH \
  "$url/iaas/api/zones/8992bdf0-136f-401c-822a-e22dae67259b?apiVersion=$api_version"
  -H 'Content-Type: application/json'
  -H "Authorization: Bearer $access_token"
  -d '{
    "placementPolicy": "SPREAD_MEMORY"
  }' | jq "."
```

Protecting Sensitive Data

By using the Automation Assembler IaaS API to mark certain data as sensitive in a request body, you can store the data in encrypted form, and ensure that only the encrypted form of data is visible in the response. Automation decrypts the data only when the actual value is needed, for example before sending a request to the cloud.

Data encryption works for certain types of data and is limited to the following use cases:

- When provisioning resources such as machines, load balancers, disks, or networks, the following types of data support encryption:
  - Custom property values for all types of resources.
  - Remote access passwords for machines.
  - Sensitive parts of the cloud config for machines.
- When creating or updating projects, custom properties support encryption.
- When updating a deployed machine, custom properties support encryption.

  Note Data encryption is only supported for deployed machines. It is not supported for discovered machines.

- When creating or updating image profiles, cloud config supports encryption. This means that you can mark parts of the cloud config script as sensitive. For example if the script includes passwords, you can mark the passwords as sensitive.

How to provision a machine with sensitive data

To mark data as sensitive, you add sensitive values with a prefix and suffix. The following example shows how to provision a new machine with sensitive values such as custom properties and a remote access password. This machine is also provisioned with a project that includes an encrypted custom property, so that the custom property is added to the machine.

1 In Automation Assembler, create a cloud account. Add a cloud zone to the cloud account and add a flavor mapping and image mapping to the cloud zone.

2 In your browser or HTTP client application, verify that all general prerequisites and prerequisites for the Automation Assembler Infrastructure as a Service (IaaS) service have been satisfied. See Chapter 3 Prerequisites for API Use Case Examples.

3 Create a project with the cloud zone that you created using the Automation Assembler UI. Include a sensitive custom property for the Active Directory (AD) password. In this way, when users related to the project provision resources with the project, they have the same AD password.

   The following example shows the AD password enclosed with the `((sensitive:` prefix and the `))` suffix to mark it as sensitive.

```
curl -X POST \
  "$url/iaas/api/projects?apiVersion=$api_version"
  -H 'Content-Type: application/json'
  -H "Authorization: Bearer $access_token"
  -d '{
      "name" : "example-project",
      "customProperties": {
           "activeDirectoryPassword":((sensitive:My-password123!))"
      }
  }' | jq "."
```

   A snippet of the response lists the project ID.

```
...
  "name": "example-project",
  "description": "This is an example project",
  "id": "5944aacb-91de-4541-bb9e-ef2a5403f81b",
  "organizationId": "8327d53f-91ea-420a-8613-ba8f3149db95",
...
```

4 Provision a virtual machine with sensitive data.

The following example includes the custom property **costCenterPassword** and a password for remote access, with values that are both marked as sensitive using the `((sensitive:` prefix and the `))` suffix. The request body also includes the ID of the project with the encrypted AD password.

```
curl -X POST \
  "$url/iaas/api/machines?apiVersion=$api_version"
  -H 'Content-Type: application/json'
  -H "Authorization: Bearer $access_token"
  -d '{
      "name" : "example-vm",
      "image" : "ubuntu",
      "flavor" : "small",
      "projectId" : "5944aacb-91de-4541-bb9e-ef2a5403f81b",
      "customProperties": {
           "costCenterPassword":"((sensitive:Pass4costCtr$$$))"
      "remoteAccess": {
           "authentication":"usernamePassword",
           "username":"example-user",
           "password":"((sensitive:example-sensitive-pass!123))"
    }
  }' | jq "."
```

Note The password for remote access is marked sensitive as an example. If left unmarked, the remote access password is encrypted because it is sensitive by default.

5 After successfully provisioning the machine, issue a **GET /iaas/api/machines** request to obtain information about the machine.

In a snippet of the response, values for the custom property **costCenterPassword** and remote access password are encrypted and appear in their encrypted form with the `((secret:v1:` prefix as in the following example.

```
..."customProperties": {
    ...
    "costCenterPassword": "((secret:v1:AAHeSZhRynh8+NSdswAdsfdsgSDffhbfh))",
    ...
},
...
"bootConfig": {
   "content": "#cloud-config\nusers:\n- default\n- name: example-user\n ...\n passwd:
((secret:v1:AAFPdqFQBiJbGKdklseiHSN28ckjSghjngj))\n..."
}
...
```

Automation converts the remote access information in the request into a cloud config script in the response. The encrypted password appears as a content value in the `bootConfig` .

Verify that the remote access password works

Even though the password is encrypted in the Automation database, you can use the user name and plain text password from the request to log in to the machine because the password is decrypted before it is sent to the cloud.

Note You can choose to verify that your remote access password works only if the cloud provider allows remote access. For example, Azure might allow remote access while GCP or AWS might not.

To test your password, use the IP address of the newly provisioned machine such as `192.168.12.1234` and the user name such as `example-user`. Log in to the remote machine with:

```
$ ssh example-user@192.168.12.1234
```

When prompted for the password, copy and paste the plain text password from the request or `example-sensitive-pass!123`. A successful login verifies that the machine was provisioned with the remote access password provided in the request.

Properties that Support Encryption

Image profiles, projects and all types of provisioned entities can include sensitive information. The table below lists all endpoints that support encryption and the parts of the request body that can contain sensitive data.

Endpoint	You can apply encryption to:	Example Input
Create/update/list machines **Note** Data encryption is only supported for deployed machines. It is not supported for discovered machines.	■ customProperties: value ■ nics: customProperties: value ■ Sensitive parts of bootConfig: content ■ remoteAccess: password	`{ "name": "machine1", "customProperties": { "username": "guest", "password": "((sensitive:mypass))" } }`
Create/update/list image profiles	Sensitive parts of imageMapping: value: cloudConfig	`{ "imageMapping" : { "ubuntu": { "id": "{{awsUbuntuId}}", "cloudConfig": "#cloud-config\nchpasswd:\n list:|\nuser1: ((sensitive:password1))" } }, "name" : "aws-image-profile", "regionId": "{{awsRegionId}}" }`

Endpoint	You can apply encryption to:	Example Input

Create/update/list projects	customProperties: value	<pre>{
		  "name": "project1",
		  "customProperties": {
		  "vidm-password":
		"((sensitive:mypass))"
		  }
		}</pre>

Create/list load balancers	■ customProperties: value
	■ nics: customProperties: value
		<pre>{
		  "name": "load-balancer1",
		  "nics": {
		  "customProperties": {
		  "dhcp-server-
		password":
		"((sensitive:mypass))"
		  }
		}</pre>

Create/list networks	customProperties: value	<pre>{
		  "name": "network1",
		  "customProperties": {
		  "dhcp-server-password":
		"((sensitive:mypass))"
		  }
		}</pre>

Create/list security groups	customProperties: value	<pre>{
		  "name": "security-group1",
		  "customProperties": {
		  "some-password":
		"((sensitive:mypass))"
		  }
		}</pre>

Create/list block device	customProperties: value	<pre>{
		  "name": "device1",
		  "customProperties": {
		  "some-password":
		"((sensitive:mypass))"
		  }
		}</pre>

Create/list compute gateway	customProperties: value	<pre>{
		  "name": "gateway1",
		  "customProperties": {
		  "some-password":
		"((sensitive:mypass))"
		  }
		}</pre>

Querying with the Automation APIs

By adding query options to an API request, you control the amount of output returned by the server and make the response easier to interpret. The API service uses the options specified to transform the data by filtering or paginating before returning the results.

You can use the following query options in your API requests. The options do not apply to all endpoints.

`$top`	Number of records to get. For more information, see Using Pagination and Count.
`$skip`	Number of records to skip. For more information, see Using Pagination and Count.
`$count`	If set to **`true`**, shows the total number of records. If used with a filter, shows the number of records matching the filter. For more information, see Using Pagination and Count.
`$select`	Names the subset of properties to list in the response.
`$filter`	Filters results by a predicate expression with operators such as, **`eq`**, **`ne`**, **`and`**, and **`or`**. For specialized filtering examples, see: ■ Filtering Resources by Region ID ■ Filtering Operations for Projects ■ Filtering for Machine Status

Endpoints that support all query options

To query for any of the following endpoints, you can use all options. Examples show how to construct a request using the **`$filter`** option with a logical **`or`** operation.

Note Automation APIs do not support filtering for nested properties with a "." in the property name.

For example, you can filter for a property with the name **`createdByEmail`** as in the following example:

```
$filter=customProperties.createdByEmail%20eq%20'user@mycompany.com'
```

However, API filtering does not support a property with the name **`my.createdByEmail`** as in the following example:

```
$filter=customProperties.my.createdByEmail%20eq%20'user@mycompany.com'
```

Endpoint	Example
Machine	`$url/iaas/api/machines?$filter=name%20ne%20'example-name'%20or%20customProperties.osType%20eq%20'example-os'`
Cloud Account	`$url/iaas/api/cloud-accounts?$filter=name%20ne%20'example-cloud-account'%20or%20customProperties.isExternal%20eq%20'false'`
Fabric Azure Storage Account	`$url/iaas/api/fabric-azure-storage-accounts/?$filter=name%20ne%20'example-name'%20or%20type%20eq%20'example-type`
Fabric Compute	`$url/iaas/api/fabric-computes?$filter=name%20ne%20'example-name'%20or%20customProperties.isExternal%20eq%20'false'`
Fabric Image	`$url/iaas/api/fabric-images?$filter=name%20ne%20'example-name'%20or%20osFamily%20eq%20'example-os'`
Fabric Network	`$url/iaas/api/fabric-networks?$filter=name%20ne%20'example-name'%20or%20externalId%20eq%20'example-id'`
Fabric Network (vSphere)	`$url/iaas/api/fabric-networks-vsphere?$filter=name%20ne%20'example-name'%20or%20externalId%20eq%20'example-id'`
Fabric vSphere Datastores	`$url/iaas/api/fabric-vsphere-datastores?$filter=name%20ne%20'example-name'%20or%20externalId%20eq%20'example-id'`
Fabric vSphere Storage Policies	`$url/iaas/api/fabric-vsphere-storage-policies?$filter=name%20ne%20'example-name'%20or%20externalId%20eq%20'example-id'`

Querying for endpoints with a specified ID

To query for an endpoint with specified ID, you can only use the **`$select`** option. Examples show how to construct a request.

Endpoint	Example
Cloud Account by ID	`$url/iaas/api/cloud-account/{id}?$select=name`
Machine by ID	`$url/iaas/api/machines/{id}?$select=name`
Fabric Azure Storage Account by ID	`$url/iaas/api/fabric-azure-storage-accounts/{id}?$select=name`
Fabric Image by ID	`$url/iaas/api/fabric-images/{id}?$select=name`
Fabric Network by ID	`$url/iaas/api/fabric-networks/{id}?$select=name`
Fabric Network (vSphere) by ID	`$url/iaas/api/fabric-networks-vsphere/{id}?$select=name`
Fabric vSphere Datastores by ID	`$url/iaas/api/fabric-vsphere-datastores/{id}?$select=name`
Fabric vSphere Storage Policies by ID	`$url/iaas/api/fabric-vsphere-storage-policies/{id}?$select=name`

Querying for a partial match

To query for the partial match of a name that starts with, ends with, or is contained within another name, the **`$filter`** options are the same for most IaaS endpoints but are different for `iaas/api/projects` and `iaas/api/deployments` endpoints. Examples show how to construct the partial match filters for the different endpoint types.

Filter Operation	Query with **`iaas/api/projects`** or **`iaas/api/deployments`**	Query with most IaaS endpoints
Name starts with **`foo`**	`?$filterstartswith(name, 'foo')`	`$filter=name%20eq%20'foo*'`
Name ends with **`foo`**	`?$filter=endswith(name, 'foo')`	`$filter=name%20eq%20'*foo'`
`foo` contained within the name	`?$filter=substringof('foo', name)`	`$filter=name%20eq%20'*foo*'`

Querying for deployments

To query for deployments, you can use all options except `$select`. The following example shows how to use the **`$filter`** option to list deployments that are not named `example-name` or have `projectId='example-id'`.

```
GET $url/iaas/api/deployments?$filter=name%20ne%20'example-
name'%20or%20projectId%20eq%20'example-id'
```

Using Pagination and Count

Pagination controls the number of elements or the range of elements returned in an API response. When the count flag is specifed, the response shows the total number of records. Use parameters in combination to traverse all elements in a result.

To paginate a response, you use the following parameters.

`$top=N`	Selects only the first N elements of the set. N must be a positive integer. Specifying N limits the maximum number of elements that the server returns in the response. The default IaaS API page size is 100. If `$top` is left unspecified, pagination selects the first 100 elements. If you want the output to include elements beyond the first 100 elements, specify a top value greater than 100.
`$skip=N`	Skips N elements and selects only the remaining elements starting with element N+1.

Pagination examples

The following examples show how to combine parameter values to control the elements returned in a vSphere deployment with 45 cloud accounts.

If you want to...	Use this request
List the first 20 cloud accounts, or 1–20	`curl -X GET -H 'Content-Type: application/json' -H "Authorization: Bearer $access_token" "$url/iaas/api/cloud-accounts?apiVersion=$api_version&$filter=cloudAccountType%20eq%20%27vsphere%27&$top=20&$skip=0" | jq "."`
List the second set of 20 cloud accounts, or 21–40	`curl -X GET -H 'Content-Type: application/json' -H "Authorization: Bearer $access_token" "$url/iaas/api/cloud-accounts?apiVersion=$api_version&$filter=cloudAccountType%20eq%20%27vsphere%27&$top=20&$skip=20" | jq "."`
List the third set of 20 cloud accounts, or 41–45	`curl -X GET -H 'Content-Type: application/json' -H "Authorization: Bearer $access_token" "$url/iaas/api/cloud-accounts?apiVersion=$api_version&$filter=cloudAccountType%20eq%20%27vsphere%27&$top=20&$skip=40" | jq "."`

Count example

The following example shows how to count the AWS cloud accounts.

```
curl -X GET -H 'Content-Type: application/json' -H "Authorization: Bearer $access_token"
"$url/iaas/api/cloud-accounts?$filter=cloudAccountType%20eq%20%27aws%27&$count=true" | jq "."
```

Filtering Resources by Region ID

You can use a filter in the IaaS APIs to identify resources provisioned in a cloud account of a particular region. By specifying the resources for which you want information such as network and security, compute, storage, and tags content, you limit the API response to provide only the information that you want.

You can also use filtering to identify and automatically update resources. For example, if you have automated builds on an vSphere private cloud account, you can use filtering to identify the image profiles associated with a region of the cloud account and automatically update those profiles in Automation.

Obtaining the externalRegionID and cloudAccountId

To get information about a resource that is in a region of a cloud account, you filter by its externalRegionId and its cloudAccountId. The following example shows how to obtain the IDs for a vSphere cloud account and use them to construct a generic query. It assumes that all general prerequisites and prerequisites for the Automation Assembler Infrastructure as a Service (IaaS) service have been satisfied. See Chapter 3 Prerequisites for API Use Case Examples.

1 List vSphere cloud accounts.

```
curl -X GET "$url/iaas/api/cloud-accounts?
apiVersion=$api_version&"'$filter='"cloudAccountType%20eq%20'vsphere'" -H 'Content-Type:
application/json' -H "Authorization: Bearer $access_token"  | jq "."
```

2 Examine the response to obtain the cloud account ID and region ID for the vSphere account that you want.

```
...
        },
        "name": "vc60",
        "description": "Created by User",
        "id":
"e0f23c91d5ecca75-7f703c5265a63d87-7e3d8d60a55d1306cc791422547ead9153c3bdf1c802400819ad45a3
41cba1f3-c39814fe67b8247557cab2652647d",
        "updatedAt": "2022-04-02",
        "organizationId": "f670fdfc-66d6-4689-9793-d524e7066d1e",
        "orgId": "f670fdfc-66d6-4689-9793-d524e7066d1e",
        "_links": {
          "regions": {
            "hrefs": [
```

```
              "/iaas/api/regions/277e3cd9fe87527557cab268cae5a"
            ]
          },
...
```

3 Assign the cloud account ID and region ID variables.

```
cloud_account_id='e0f23c91d5ecca75-7f703c5265a63d87-7e3d8d60a55d1306cc791422547ead9153c3bdf
1c802400819ad45a341cba1f3-c39814fe67b8247557cab2652647d'
region_id='277e3cd9fe87527557cab268cae5a'
```

4 List regions with the region ID and associated with the cloud account ID.

```
curl -X GET -H 'Content-Type: application/json' -H "Authorization: Bearer $access_token"
"$url/iaas/api/regions/?
apiVersion=$api_version&"'$filter'"=id%20eq%20'"$region_id"'%20and%20cloudAccountId%20eq%20
'"$cloud_account_id"'"  | jq "."
```

5 Examine the response to obtain the external region ID.

```
...
          {
            "externalRegionId": "Datacenter:datacenter-2",
            "name": "VC60-Datacenter",
            "cloudAccountId":
"e0f23c91d5ecca75-7f703c5265a63d87-7e3d8d60a55d1306cc791422547ead9153c3bdf1c802400819ad45a3
41cba1f3-c39814fe67b8247557cab2652647d",
            "id": "277e3cd9fe87527557cab268cae5a",
            "updatedAt": "2022-04-02",
...
```

6 Assign the external region ID variable.

```
external_region_id='Datacenter:datacenter-2'
```

Using the externalRegionId and cloudAccountId, you can construct a generic filter to use in any IaaS API that queries for resources.

```
$filter="externalRegionId%20eq%20'$external_region_id'%20and%20cloudAccountId%20eq%20'$cloud_a
ccount_id'"
```

Constructing a query for a VMC cloud account

The VMC cloud account consists of an AWS cloud account facade and associated internal vSphere and NSX cloud accounts. To construct the query for a resource in the VMC cloud account, you use the cloudAccountId of the associated vSphere cloud account and not the cloudAccountId of the VMC cloud account.

The following example shows how to obtain the cloudAccountId and externalRegionId and use them to construct a generic query. It assumes that you know the name of the VMC cloud account and that all general prerequisites and prerequisites for the Automation Assembler Infrastructure as a Service (IaaS) service have been satisfied. See Chapter 3 Prerequisites for API Use Case Examples.

1 Assign the cloud account name variable.

```
cloud_account_name='<my_vmc_cloud_account>'
```

2 List VMC cloud accounts with your VMC cloud account name.

```
curl -X GET "$url/iaas/api/cloud-accounts?
apiVersion=$api_version&"'$filter='"cloudAccountType%20eq%20'vmc'%20and%20name%20eq%20'$clo
ud_account_name'"  -H 'Content-Type: application/json' -H "Authorization: Bearer
$access_token"  | jq "."
```

3 Examine the response to obtain the cloud account ID for the associated vSphere cloud account.

Under `_links`, two associated cloud accounts are listed. The first cloud account is the vSphere cloud account. The second is the NSX cloud account.

```
...
       "_links": {
                "regions": {
                    "hrefs": [
                        "/iaas/api/regions/5a8da7fe-63d9-4f82-84e8-f47bfdfef43c"
                    ]
                },
                "associated-cloud-accounts": {
                    "hrefs": [
                        "/iaas/api/cloud-accounts/df59d7cd-d3ee-4bc3-bf4a-8a4027cefb05",
                        "/iaas/api/cloud-accounts/6993943b-82bc-4ec7-bad6-65f2f595da7e"
                    ]
                },
                "self": {
                    "href": "/iaas/api/cloud-accounts/95be1cc2-c70a-4311-9f18-6218786ac51b"
                }
            }

...
```

4 To assign the cloud account ID variable, use the associated vSphere cloud account.

```
assoc_cloud_account_id='df59d7cd-d3ee-4bc3-bf4a-8a4027cefb05'
```

5 List regions in the associated vSphere cloud account.

```
curl -X GET -H 'Content-Type: application/json'
-H "Authorization: Bearer $access_token" "$url/iaas/api/regions/?
apiVersion=$api_version&"'$filter'"=cloudAccountId%20eq%20'"$assoc_cloud_account_id"'"  |
jq "."
```

6 Examine the response to obtain the external region ID.

```
...
         {
            "externalRegionId": "Datacenter:datacenter-3",
            "cloudAccountId": "df59d7cd-d3ee-4bc3-bf4a-8a4027cefb05",
            "id": "e572c96b-9084-4c37-b74d-bf61f0b08e79",
            "updatedAt": "2022-04-02",
            "organizationId": "33a92056-18f0-469c-a088-9c8eae4e4888",
            "orgId": "33a92056-18f0-469c-a088-9c8eae4e4888",
            "_links": {
                "self": {
                    "href": "/iaas/api/regions/e572c96b-9084-4c37-b74d-bf61f0b08e79"
                },
                "cloud-account": {
                    "href": "/iaas/api/cloud-accounts/df59d7cd-d3ee-4bc3-bf4a-8a4027cefb05"
                }
            }
        }

...
```

7 Assign the external region ID variable.

```
external_region_id='Datacenter:datacenter-3'
```

Using the externalRegionId and cloudAccountId for the associated vSphere cloud account, you can construct a generic filter to use in any IaaS API that queries for resources in the VMC cloud account.

```
$filter="externalRegionId%20eq%20'$external_region_id'%20and%20cloudAccountId%20eq%20'$assoc_c
loud_account_id'"
```

Resource Query Examples

The following examples show how to incorporate the filter into resource queries.

Resource	Query
Security Group	`$url/iaas/api/security-groups?$filter="externalRegionId%20eq%20'$external_region_id'%20and%20cloudAccountId%20eq%20'$cloud_account_id'"`
Security Group (NSX only)	`$url/iaas/api/security-groups?$filter="externalRegionId%20eq%20'global'%20and%20cloudAccountId%20eq%20'$cloud_account_id'"`
Region	`$url/iaas/api/regions?$filter="externalRegionId%20eq%20'$external_region_id'%20and%20cloudAccountId%20eq%20'$cloud_account_id'"`
Network	`$url/iaas/api/networks?$filter="externalRegionId%20eq%20'$region_id'%20and%20cloudAccountId%20eq%20'$cloud_account_id'"`
Network Domains	`$url/iaas/api/network-domains?$filter="externalRegionId%20eq%20'$external_region_id'%20and%20cloudAccountId%20eq%20'$cloud_account_id'"`
Machine	`$url/iaas/api/machines?$filter="externalRegionId%20eq%20'$external_region_id'%20and%20cloudAccountId%20eq%20'$cloud_account_id'"`
Load Balancer	`$url/iaas/api/load-balancers?$filter="externalRegionId%20eq%20'$external_region_id'%20and%20cloudAccountId%20eq%20'$cloud_account_id'"`
Fabric vSphere Datastore	`$url/iaas/api/fabric-vsphere-datastores?$filter="externalRegionId%20eq%20'$external_region_id'%20and%20cloudAccountId%20eq%20'$cloud_account_id'"`
Fabric Network vSphere	`$url/iaas/api/fabric-networks-vsphere?$filter="externalRegionId%20eq%20'$external_region_id'%20and%20cloudAccountId%20eq%20'$cloud_account_id'"`
Fabric Network	`$url/iaas/api/fabric-networks?$filter="externalRegionId%20eq%20'$external_region_id'%20and%20cloudAccountId%20eq%20'$cloud_account_id'"`

Resource	Query
Fabric Compute	`$url/iaas/api/fabric-computes?$filter="externalRegionId%20eq%20'$external_region_id'%20and%20cloudAccountId%20eq%20'$cloud_account_id'"`
Block Device	`$url/iaas/api/block-devices?$filter="externalRegionId%20eq%20'$external_region_id'%20and%20cloudAccountId%20eq%20'$cloud_account_id'"`
Network Profile	`$url/iaas/api/network-profiles?$filter="externalRegionId%20eq%20'$external_region_id'%20and%20cloudAccountId%20eq%20'$cloud_account_id'"`
Flavor Profile	`$url/iaas/api/flavor-profiles?$filter="externalRegionId%20eq%20'$external_region_id'%20and%20cloudAccountId%20eq%20'$cloud_account_id'"`
Image Profiles	`$url/iaas/api/image-profiles?$filter="externalRegionId%20eq%20'$external_region_id'%20and%20cloudAccountId%20eq%20'$cloud_account_id'"`
Storage Profiles	`$url/iaas/api/storage-profiles?$filter="externalRegionId%20eq%20'$external_region_id'%20and%20cloudAccountId%20eq%20'$cloud_account_id'"`
Storage Profiles Azure	`$url/iaas/api/storage-profiles-azure?$filter="externalRegionId%20eq%20'$external_region_id'%20and%20cloudAccountId%20eq%20'$cloud_account_id'"`
Storage Profiles AWS	`$url/iaas/api/storage-profiles-aws?$filter="externalRegionId%20eq%20'$external_region_id'%20and%20cloudAccountId%20eq%20'$cloud_account_id'"`
Storage Profiles vSphere	`$url/iaas/api/storage-profiles-vsphere?$filter="externalRegionId%20eq%20'$external_region_id'%20and%20cloudAccountId%20eq%20'$cloud_account_id'"`

Filtering for Machine Status

To filter for machines with deployed or discovered status, you can use the IaaS APIs. A machine with the discovered status is in the cloud and has not yet been onboarded. A machine with the deployed status has been onboarded or provisioned from Automation.

If a machine has a discovered status, you can bring it into VMware Aria Automation management using the onboarding tool. See Onboard selected machines as a single deployment.

After onboarding, the machine has a deployed status and is associated with the deployment. Then you can manage it in the same way as any other provisioned machine.

To query for deployed or discovered status, you use the following parameters.

`$filter=deploymentId eq '*'`	Returns all machines that are deployed and under Automation management. Deployed machines have a deployment ID.
`$filter=deploymentId ne '*'`	Returns all machines that are discovered in the cloud. Machines without the deployment ID property are not deployed.

Machine filter examples

The following examples show how to list deployed or discovered machines.

If you want to...	Use this request
List all deployed machines	`curl -X GET "$url/iaas/api/machines?apiVersion=$api_version&"'$filter='"deploymentId%20eq%20'*' -H 'Content-Type: application/json' -H "Authorization: Bearer $access_token"  | jq "."`
List all discovered machines	`curl -X GET "$url/iaas/api/cloud-accounts?apiVersion=$api_version&"'$filter='"deploymentId%20ne%20'*' -H 'Content-Type: application/json' -H "Authorization: Bearer $access_token"  | jq "."`

Filtering Operations for Projects

To restrict the output from a project endpoint, you include a filter in the `GET /project-service/api/projects` request. The API service uses the operation to filter for a collection of projects to paginate, order, or count your results.

Filter operation examples

The following examples show how to incorporate filter operations into project queries.

Operation	Query
equal	`$url/project-service/api/projects?$filter=name%20eq%20'00-TestProject'`
not equal	`$url/project-service/api/projects?$filter=name%20ne%20'00-TestProject'`

Operation	Query
logical and	`$url/project-service/api/projects?$filter=name%20eq%20'00-TestProject'%20and%20sharedResources%20eq%20true`
logical or	`$url/project-service/api/projects?$filter=name%20eq%20'00-TestProject'%20or%20sharedResources%20eq%20false`
logical negation	`$url/project-service/api/projects?$filter=not%20startswith(name,'00')`
starts with	`$url/project-service/api/projects?$filter=startswith(name, '00')`
substring of	`$url/project-service/api/projects?$filter=substringof('00', name)`
ends with	`$url/project-service/api/projects?$filter=endswith(name, '00')`
length	`$url/project-service/api/projects?$filter=length(name)%20eq%205`
index of	`$url/project-service/api/projects?$filter=indexof(name,'00')%20eq%200`
substring from	`$url/project-service/api/projects?$filter=substring(name, 1)%20eq%20'0-Te'`
substring from to	`$url/project-service/api/projects?$filter=substring(name, 1, 2)%20eq%20'0-'`
to lower	`$url/project-service/api/projects?$filter=tolower(name)%20eq%20'00-testproject'`
to upper	`$url/project-service/api/projects?$filter=toupper(name)%20eq%20'00-TESTPROJECT'`
concatenate	`$url/project-service/api/projects?$filter=concat(concat(name,','),description)%20eq%20'test project,test project description'`
trim	`$url/project-service/api/projects?$filter=trim(name)%20eq%20'00-TestProject'`

Related query examples

The following examples show how to use related query options to paginate, order, or count your results.

If you want to...	Use this request
List the first twenty items	`curl -X GET $url/project-service/api/projects?$top=20 -H 'Content-Type: application/json' -H "Authorization: Bearer $access_token"  | jq "."`
Skip the first ten items in the collection	`curl -X GET $url/project-service/api/projects?$skip=10 -H 'Content-Type: application/json' -H "Authorization: Bearer $access_token"  | jq "."`

List the items in ascending order with **asc** or descending order with **desc**. If the type of order is not specified, the list defaults to ascending order.	For ascending order:
	`curl -X GET $url/project-service/api/projects?$skip=0$orderBy=name%20asc -H 'Content-Type: application/json' -H "Authorization: Bearer $access_token"  | jq "."`
	For descending order:
	`curl -X GET $url/project-service/api/projects?$skip=0$orderBy=name%20desc -H 'Content-Type: application/json' -H "Authorization: Bearer $access_token"  | jq "."`

List the total number of items in the collection.	`curl -X GET $url/project-service/api/projects?$count=true -H 'Content-Type: application/json' -H "Authorization: Bearer $access_token"  | jq "."`

Setting up Automation Assembler using APIs

5

As a Automation Assembler administrator, you can use the Infrastructure as a Service (IaaS) APIs to set up connections with your cloud account vendor and integration applications in VMware Aria Automation.

Read the following topics next:

- Adding Cloud Accounts
- Integrating with other applications

Adding Cloud Accounts

You use the IaaS APIs to create cloud accounts for various cloud platforms. Automation Assembler uses permissions configured in the cloud accounts to collect data from regions or data centers, and to deploy cloud templates to those regions.

After adding a cloud account, you use the cloud account ID to create a cloud zone, flavor mappings, image mappings, network, profiles, or storage profiles.

Add an Amazon Web Services Cloud Account

To create an Amazon Web Services cloud account with or without cloud zones, you make a POST request. The request body includes the parameters specific to Amazon Web Services that are required to create the cloud account.

Prerequisites

- Verify that all general prerequisites and prerequisites for the Automation Assembler Infrastructure as a Service (IaaS) service have been satisfied. See Chapter 3 Prerequisites for API Use Case Examples.
- Verify that you have the Amazon Web Services access key ID and the Amazon Web Services secret access key for the new cloud account.

Procedure

1 Assign the Amazon Web Services account variables.

```
aws_access_key_id='<your_aws_access_key_id>'
aws_secret_access_key='<your_aws_secret_access_key>'
```

2 Submit a request to create an Amazon Web Services cloud account without default cloud zones.

When the parameter **createDefaultZones** is left unspecified, the cloud account is created without default cloud zones by default.

```
curl -X POST \
  "$url/iaas/api/cloud-accounts?apiVersion=$api_version" \
  -H 'Content-Type: application/json' \
  -H "Authorization: Bearer $access_token" \
  -d '{
    "privateKeyId": "'$aws_access_key_id'",
    "cloudAccountType": "aws",
    "name": "<your_aws_cloud_account>",
    "description": "This is a demo AWS cloud account without cloud zones",
    "regionIds": [ "<your_region_id1>", "<your_region_id2>" ],
    "privateKey": "'$aws_secret_access_key'"
  }' | jq "."
```

3 (Optional) Submit a request to create an Amazon Web Services cloud account with default cloud zones.

```
curl -X POST \
  "$url/iaas/api/cloud-accounts?apiVersion=$api_version" \
  -H 'Content-Type: application/json' \
  -H "Authorization: Bearer $access_token" \
  -d '{
    "privateKeyId": "'$aws_access_key_id'",
    "cloudAccountType": "aws",
    "name": "<your_aws_cloud_account>",
    "description": "This is a demo AWS cloud account with default cloud zones",
    "regionIds": [ "<your_region_id1>", "<your_region_id2>" ],
    "createDefaultZones": true,
    "privateKey": "'$aws_secret_access_key'"
  }' | jq "."
```

4 To obtain your cloud account ID, examine the response.

5 Assign the cloud account ID variable to your cloud account ID.

```
cloud_account_id='<your_cloud_account_id>'
```

6 List all cloud accounts.

```
curl -X GET $url/iaas/api/cloud-accounts?apiVersion=$api_version -H 'Content-Type:
application/json' -H "Authorization: Bearer $access_token"  | jq "."
```

7 (Optional) To look up the cloud account you created, list the cloud account with your cloud account ID.

```
curl -X GET $url/iaas/api/cloud-accounts/$cloud_account_id?apiVersion=$api_version -H
'Content-Type: application/json' -H "Authorization: Bearer $access_token" | jq "."
```

The response shows the cloud account name and ID for the Amazon Web Services cloud account you created.

Example: Create an Amazon Web Services Cloud Account

Assign the required variables.

```
$ url='https://appliance.domain.com'
$ api_version='2021-07-15'
$ aws_access_key_id='FEDCBA5JJ2F43TUVW7XA'
$ aws_secret_access_key='XYZfGo0/Vb5XPezeC58QRSvLMNOwsHhuB2IbEJxL'
```

Create a cloud account named **`demo-aws-account`** without default cloud zones.

```
$ curl -X POST \
  "$url/iaas/api/cloud-accounts?apiVersion=$api_version" \
  -H 'Content-Type: application/json' \
  -H "Authorization: Bearer $access_token" \
  -d '{
    "privateKeyId": "'$aws_access_key_id'",
    "cloudAccountType": "aws",
    "name": "demo-aws-account",
    "description": "This is a demo AWS cloud account without cloud zones",
    "regionIds": [ "us-east-1", "us-west-1" ],
    "privateKey": "'$aws_secret_access_key'"
  }' | jq "."
```

A snippet of the response from your request shows the account ID.

```
...
"tags": [],
"name": "demo-aws-account",
"id": "c8c3c9bfdb449475-7f703c5265a63d87-
f8e705d89b2569e1aac66c6d00bf4fc7ef4b1c44100f0e944af31eb8ba3d2a5a-
f4226a20b65c4675574bc5fbff6c0",
"updatedAt": "2022-04-02",
"organizationId": "8327d53f-91ea-420a-8613-ba8f3149db95",
...
```

Add a vSphere Cloud Account

To create a vSphere cloud account, you make a POST request. The request body includes the parameters specific to vSphere that are required to create the cloud account.

The following procedure shows how to create a vSphere cloud account that supports a trusted certificate. To obtain a trusted certificate, you submit a request to validate asynchronously with the vSphere cloud account specification. When the validation request completes successfully, you use the certificate ID from the response to obtain the trusted certificate that you submit when you create the vSphere cloud account.

Prerequisites

- Verify that all general prerequisites and prerequisites for the Automation Assembler Infrastructure as a Service (IaaS) service have been satisfied. See Chapter 3 Prerequisites for API Use Case Examples.
- Verify that you have the following parameters for the new cloud account:
  - vSphere host name
  - vSphere user name
  - vSphere password
- Verify that you have an existing vSphere, NSX-T, NSX-V, or VMC cloud account that you want to associate with the new cloud account and obtain the cloud account ID.

  a List all cloud accounts.

  ```
  curl -X GET $url/iaas/api/cloud-accounts?apiVersion=$api_version -H 'Content-Type:
  application/json' -H "Authorization: Bearer $access_token"  | jq "."
  ```

  b Examine the response to obtain the cloud account ID such as the `id` value in this example.

  ```
  ...
        "name": "vsphere-account-example",
        "id": "b9fa1b42c767de7558ceff3b78004",
        "updatedAt": "2022-04-02",
        "orgId": "f670fdfc-66d6-4689-9793-d524e7066d1e",
  ...
  ```

Procedure

1 List all cloud proxies.

```
curl -X GET -H 'Content-Type: application/json' -H "Authorization: Bearer $access_token"
"$url/iaas/api/data-collectors?apiVersion=$api_version" | jq "."
```

2 To obtain the data collector ID, examine the response.

3 Assign the data collector ID variable.

```
data_collector_id='<your_datacollector_id>'
```

4 Assign the vSphere account variables.

```
vsphere_host_name='<your_vsphere_host_name>'
vsphere_user='<your_vsphere_user_name>'
vsphere_password='<your_vsphere_password>'
```

5 List external region IDs from a vSphere cloud account.

```
curl -X POST \
  "$url/iaas/api/cloud-accounts-vsphere/region-enumeration?apiVersion=$api_version" \
  -H 'Content-Type: application/json' \
```

```
  -H "Authorization: Bearer $access_token" \
  -d '{
  "cloudAccountType": "vsphere",
  "username": "'$vsphere_user'",
  "password": "'$vsphere_password'",
  "hostName": "'$vsphere_host_name'",
  "dcid": "'$data_collector_id'",
  "acceptSelfSignedCertificate": "false"
}' | jq "."
```

6 To obtain the external region ID, examine the response and assign the region ID variable.

```
vsphere_region_id='<your_vsphere_region_id>'
```

7 Submit a request to validate asynchronously with the vSphere cloud account specification.

```
curl -X POST -H 'Content-Type: application/json' -H "Authorization: Bearer $access_token"
"$url/iaas/api/cloud-accounts-vsphere?apiVersion=$api_version&validateOnly" | jq "."
```

The response includes a selfLink.

```
{
  "progress": 0,
  "status": "INPROGRESS",
  "name": "Cloud account specification validation",
  "id": "example-selfLink-alphanumeric-string",
  "selfLink": "/iaas/api/request-tracker/example-selfLink-alphanumeric-string"
}
```

8 Assign the selfLink variable.

```
selfLink_id='example-selfLink-alphanumeric-string'
```

9 Use the selfLink variable to track the request.

```
curl -X GET -H 'Content-Type: application/json' -H "Authorization: Bearer $access_token"
"$url/iaas/api/request-tracker/$selfLink_id?apiVersion=$api_version" | jq "."
```

When the validation request completes successfully, the response includes a resource with the certificate ID.

```
{
  "progress": 0,
  "message": "valid certificate found",
  "status": "SUCCEEDED",
  "resources": [
      "/iaas/api/certificates/example-certificate-id-string"
  ],
  "name": "Cloud account specification validation",
  "id": "example-selfLink-alphanumeric-string",
  "selfLink": "/iaas/api/request-tracker/example-selfLink-alphanumeric-string"
}
```

10 Assign the certificate ID variable.

```
certificate_id='example-certificate-id-string'
```

11 Submit a request to get the certificate information.

```
PEM_for_X509Certificate='curl -X GET -H 'Content-Type: application/json'
-H "Authorization: Bearer $access_token" "$url/iaas/api/certificates/$certificate_id?
apiVersion=$api_version" | jq "."'
```

12 Assign the ID of the existing cloud account to associate with the new cloud account.

```
existing_cloud_account_ID='<your_existing_cloud_account_ID>'
```

13 Include the certificate in the request to create a vSphere cloud account with default cloud zones.

To create a vSphere cloud account without default cloud zones, use `"createDefaultZones":false`.

```
curl -X POST \
  "$url/iaas/api/cloud-accounts-vpshere?apiVersion=$api_version" \
  -H 'Content-Type: application/json' \
  -H "Authorization: Bearer $access_token" \
  -d '{
    "name": "demo-vsphere-account",
    "description": "This is a demo vSphere account with default cloud zones",
    "username": "'$vsphere_user'",
    "password": "'$vsphere_password'",
    "hostName": "'$vsphere_host_name'",
    "acceptSelfSignedCertificate":false,
    "associatedCloudAccountIds": "'$existing_cloud_account_ID'",
    "createDefaultZones":true,
    "dcId": "'$data_collector_id'",
    "regions":[
      {
         "name": "'$vsphere_region_id'",
         "ExternalRegionId":"'$vsphere_region_id'"
      }
    ],
    "tags": [
      {
        "key": "env",
        "value": "dev"
      }
    ],
```

```
    "certificateInfo":{
       "certificate": "'$PEM_for_X509Certificate'"
       }
}' | jq "."
```

Note The following example shows how to create a vSphere cloud account with multiple cloud account IDs. However, the payload can only include a single NSX-P-Cloud endpoint and a single VMC endpoint.

```
"associatedCloudAccountIds": "[\"'$existing_NSXT_cloud_account_ID'\",
\"'$existing_VMC_cloud_account_ID'\", \"'$existing_vSphere_cloud_account_ID'\"]",
```

14 List all cloud accounts.

```
curl -X GET $url/iaas/api/cloud-accounts?apiVersion=$api_version -H 'Content-Type:
application/json' -H "Authorization: Bearer $access_token" | jq "."
```

15 Examine the response and verify that the name and ID of the vSphere cloud account you created is listed.

Example: Create a vSphere Cloud Account

This example creates a cloud account with default cloud zones.

Assign the required variables.

```
$ url='https://appliance.domain.com'
$ api_version='2021-07-15'
```

List all cloud proxies.

```
$ curl -X GET -H 'Content-Type: application/json' -H "Authorization: Bearer $access_token"
"$url/iaas/api/data-collectors?apiVersion=$api_version" | jq "."
```

A snippet of the response from your request shows the data collector IDs.

```
...
      {
      "dcId": "60740040-f3cd-4694-96da-15e547242bf7",
      "ipAddress": "10.108.78.154",
      "name": "example-prod-corp-rdc",
      "hostName": "corp-v783-dhcp-79-85.eng.mycompany.com",
      "status": "ACTIVE"
    },
...
```

Assign the data collector ID variable.

```
$ data_collector_id='60740040-f3cd-4694-96da-15e547242bf7'
```

Assign the vSphere account variables.

```
$ vsphere_host_name='corp-v783-dhcp-79-85.eng.mycompany.com'
$ vsphere_user='admin@mycompany.com'
$ vsphere_password='my_vsphere_password'
```

List external region IDs from your vSphere cloud account.

```
$ curl -X POST \
  "$url/iaas/api/cloud-accounts-vsphere/region-enumeration?apiVersion=$api_version" \
  -H 'Content-Type: application/json' \
  -H "Authorization: Bearer $access_token" \
  -d '{
  "cloudAccountType": "vsphere",
  "username": "'$vsphere_user'",
  "password": "'$vsphere_password'",
  "hostName": "'$vsphere_host_name'",
  "dcid": "'$data_collector_id'",
  "acceptSelfSignedCertificate": "false"
}' | jq "."
```

A snippet of the response shows the region ID to use.

```
...
{
  "externalRegionIds": [
    "Datacenter:datacenter-2"
  ]
}
...
```

Assign the region ID variable.

```
$ vsphere_region_id='Datacenter:datacenter-2'
```

Submit request to validate asynchronously with the vSphere cloud account specification.

```
$ curl -X POST -H 'Content-Type: application/json' -H "Authorization: Bearer $access_token"
"$url/iaas/api/cloud-accounts-vsphere?apiVersion=$api_version&validateOnly" | jq "."
```

A snippet of the response shows the selfLink.

```
{
  "progress": 0,
  "status": "INPROGRESS",
  "name": "Cloud account specification validation",
  "id": "bcdee18-a77d-46f8-b068-4013e80e2b55",
  "selfLink": "/.../request-tracker/bbcdee18-a77d-46f8-b068-4013e80e2b55"
}
```

Assign the selfLink variable.

```
$ selfLink_id='bbcdee18-a77d-46f8-b068-4013e80e2b55'
```

Submit a request to track the request with the selfLink.

```
$ curl -X GET -H 'Content-Type: application/json' -H "Authorization: Bearer $access_token"
"$url/iaas/api/request-tracker/$selfLink_id?apiVersion=$api_version | jq "."
```

When the request succeeds, the response shows the resource with the certificate ID.

```
{
  "progress": 0,
  "message": "valid certification path to requested target found",
  "status": "SUCCEEDED",
  "resources": [
      "/iaas/api/certificates/7fe4c108-64ff-4347-92de-b0790bda1a3c?apiversion=2021-07-15"
  ],
  "name": "Cloud account specification validation",
  "id": "bbcdee18-a77d-46f8-b068-4013e80e2b55",
  "selfLink": "/iaas/api/request-tracker/bbcdee18-a77d-46f8-b068-4013e80e2b55"
}
```

Assign the certificate ID variable.

```
$ certificate_id='7fe4c108-64ff-4347-92de-b0790bda1a3c'
```

To get certficate information, submit a request with the certificate ID.

```
$ PEM_for_X509Certificate='curl -X GET -H 'Content-Type: application/json' -H "Authorization:
Bearer $access_token" "$url/iaas/api/certificates/$certificate_id?apiVersion=$api_version" |
jq "."'
```

Assign the ID of the existing cloud account to associate with the new cloud account.

```
$ existing_cloud_account_id ='b9fa1b42c767de7558ceff3b78004'
```

Create a cloud account named **`demo-vsphere-account`** with default cloud zones.

```
$ curl -X POST \
  "$url/iaas/api/cloud-accounts-vpshere?apiVersion=$api_version" \
  -H 'Content-Type: application/json' \
  -H "Authorization: Bearer $access_token" \
  -d '{
    "name": "demo-vsphere-account",
    "description": "This is a demo vSphere account with default cloud zones",
    "username": "'$vsphere_user'",
    "password": "'$vsphere_password'",
    "hostName": "'$vsphere_host_name'",
    "acceptSelfSignedCertificate":false,
    "associatedCloudAccountIds": "'$existing_cloud_account_id'",
    "createDefaultZones":true,
    "dcId": "'$data_collector_id'",
    "regions":[
      {
         "name": "'$vsphere_region_id'",
         "ExternalRegionId":"'$vsphere_region_id'"
```

```
      }
    ],
    "tags": [
      {
        "key": "env",
        "value": "dev"
      }
    ],
    "certificateInfo":{
       "certificate": "'$PEM_for_X509Certificate'"
       }
}' | jq "."
```

A snippet of the response from your request shows the account ID.

```
...
      "tags": [],
      "name": "demo-vsphere-account",
      "id": "515684ccebafde75-7f703c5265a63d87-
e78aab87e9c8d5cd4cd1da1a285403f0f4e77a5240720d093e147b830b172542-23b5c527d7083675572f5099a8da0
",
      "updatedAt": "2022-04-02",
      "organizationId": "8327d53f-91ea-420a-8613-ba8f3149db95",
      "orgId": "8327d53f-91ea-420a-8613-ba8f3149db95",
...
```

Add an NSX-T or NSX-V Cloud Account

To create an NSX-T or NSX-V cloud account, you make a POST request. The request body includes the NSX-specific parameters required to create the cloud account.

As an alternative to using the `cloud-accounts` API call, you can use a `cloud-accounts-nsx-t` or `cloud-accounts-nsx-v` API call to list NSX-T or NSX-V cloud accounts respectively.

Prerequisites

- Verify that all general prerequisites and prerequisites for the Automation Assembler Infrastructure as a Service (IaaS) service have been satisfied. See Chapter 3 Prerequisites for API Use Case Examples.
- Verify that you have the following parameters for the new cloud account:
  - NSX host name
  - NSX user name
  - NSX password

Procedure

1 List all cloud proxies.

```
curl -X GET -H 'Content-Type: application/json' -H "Authorization: Bearer $access_token"
"$url/iaas/api/data-collectors?api_version=$api_version" | jq "."
```

2 Examine the response and assign the data collector variable.

```
data_collector_id='<your_datacollector_id>'
```

3 Assign the NSX account variables.

```
nsx_host_name='<your_nsx_host_name>'
nsx_username='<your_nsx_user_name>'
nsx_password='<your_nsx_password>'
```

4 (Optional) If you have a vSphere cloud account that you want to associate with the NSX cloud account, assign the ID of the vSphere cloud account to the **vsphere_cloud_account_id** variable. See Add a vSphere Cloud Account .

```
vpshere_cloud_accout_id='<your_vsphere_cloud_account_id>'
```

5 Submit a request to create an NSX-V cloud account. To add an NSX-T cloud account, use **"cloudAccountType": "nsxt"**. This example includes the **vsphere_cloud_account_id** variable.

```
curl -X POST \
  "$url/iaas/api/cloud-accounts?apiVersion=$api_version" \
  -H 'Content-Type: application/json' \
  -H "Authorization: Bearer $access_token" \
  -d '{
  "cloudAccountType": "nsxv",
  "privateKeyId": "'$nsx_username'",
  "privateKey": "'$nsx_password'",
  "associatedCloudAccountIds": [
    "'$vsphere_cloud_account_id'"
  ],
  "cloudAccountProperties": {
    "hostName": "'$nsx_host_name'",
    "acceptSelfSignedCertificate": "true",
    "dcId": "'$data_collector_id'",
    "privateKeyId": "'$nsx_username'",
    "privateKey": "'$nsx_password'"
  },

  "tags": [
    {
      "key": "env",
      "value": "prod"
    }
  ],
  "name": "<your_nsx_cloud_account>",
  "description": "Example NSX cloud account description"
}' | jq "."
```

6 To obtain the NSX cloud account ID, examine the response.

7 Assign the NSX cloud account ID variable.

```
nsx_cloud_id='<your_nsx_cloud_id>'
```

8 List all cloud accounts.

```
curl -X GET -H 'Content-Type: application/json' -H "Authorization: Bearer $access_token"
"$url/iaas/api/cloud-accounts?apiVersion=$api_version"  | jq "."
```

9 (Optional) List all NSX-T cloud accounts using the `cloud-accounts-nsx-t` API call.

```
curl -X GET -H 'Content-Type: application/json' -H "Authorization: Bearer $access_token"
"$url/iaas/api/cloud-accounts-nsx-t?apiVersion=$api_version"  | jq "."
```

10 (Optional) List all NSX-V cloud accounts using the `cloud-accounts-nsx-v` API call.

```
curl -X GET -H 'Content-Type: application/json' -H "Authorization: Bearer $access_token"
"$url/iaas/api/cloud-accounts-nsx-v?apiVersion=$api_version"  | jq "."
```

11 Examine the response and verify that the name and ID of the NSX cloud account you created is listed.

Note The ID of the associated vSphere cloud account appears under `_links` and `hrefs` as in the following code snippet example.

```
...
      "name": "nsxv manager",
      "id": "515684ccebafde75-7f703c5265a63d87-
e78aab87e9c8d5cd4cd1da1a285403f0f4e77a5240720d093e147b830b172542-
d5a5e16bdc3eec75572358fd24ab6",
      "updatedAt": "2022-04-02",
      "organizationId": "f670fdfc-66d6-4689-9793-d524e7066d1e",
      "orgId": "f670fdfc-66d6-4689-9793-d524e7066d1e",
      "_links": {
        "associated-cloud-accounts": {
          "hrefs": [
            "/iaas/api/cloud-accounts/515684ccebafde75-7f703c5265a63d87-
e78aab87e9c8d5cd4cd1da1a285403f0f4e77a5240720d093e147b830b172542-23b5c527d7083675572f5099a8
da0"
          ]
        },
        "self": {
          "href": "/iaas/api/cloud-accounts/515684ccebafde75-7f703c5265a63d87-
e78aab87e9c8d5cd4cd1da1a285403f0f4e77a5240720d093e147b830b172542-
d5a5e16bdc3eec75572358fd24ab6"
...
```

Example: Create an NSX-V Cloud Account

This example creates an NSX-V cloud account that includes an existing vSphere cloud account.

Assign the required variables.

```
$ url='https://appliance.domain.com'
$ api_version='2021-07-15'
```

List all cloud proxies.

```
$ curl -X GET -H 'Content-Type: application/json' -H "Authorization: Bearer $access_token"
"$url/iaas/api/data-collectors?api_version=$api_version" | jq "."
```

A snippet of the response from your request shows the data collector IDs.

```
...
    {
      "dcId": "cd7d1eb4-573f-4150-8206-de3d536490ca",
      "ipAddress": "10.139.116.60",
      "name": "localhost.localdom",
      "hostName": "localhost.localdom",
      "status": "ACTIVE"
    },,
...
```

Assign the data collector ID variable.

```
$ data_collector_id='cd7d1eb4-573f-4150-8206-de3d536490ca'
```

Assign the NSX account variables.

```
$ nsx_host_name='nsx-manager.mycompany.local'
$ nsx_username='admin'
$ nsx_password='my_nsx_password'
```

Assign the account variables for your existing vSphere cloud account.

```
$ vsphere_cloud_account_id='515684ccebafde75-7f703c5265a63d87-
e78aab87e9c8d5cd4cd1da1a285403f0f4e77a5240720d093e147b830b172542-23b5c527d7083675572f5099a8da0
'
```

Create an NSX-V cloud account named **`demo-nsxv-account`**.

```
$ curl -X POST \
  "$url/iaas/api/cloud-accounts?apiVersion=$api_version" \
  -H 'Content-Type: application/json' \
  -H "Authorization: Bearer $access_token" \
  -d '{
  "cloudAccountType": "nsxv",
  "privateKeyId": "'$nsx_username'",
  "privateKey": "'$nsx_password'",
  "associatedCloudAccountIds": [
    "'$vsphere_cloud_account_id'"
  ],
  "cloudAccountProperties": {
    "hostName": "'$nsx_host_name'",
```

```
    "acceptSelfSignedCertificate": "true",
    "dcId": "'$data_collector_id'",
    "privateKeyId": "'$nsx_username'",
    "privateKey": "'$nsx_password'"
  },

  "tags": [
    {
      "key": "env",
      "value": "prod"
    }
  ],
  "name": "demo-nsxv-account",
  "description": "Example NSX cloud account description"
}' | jq "."
```

A snippet of the response from your request shows the account ID.

```
...
      "tags": [],
      "name": "demo-nsx-account",
      "id": "7b2c48362c94567559080d8f575a2",
      "updatedAt": "2022-04-02",
      "organizationId": "8327d53f-91ea-420a-8613-ba8f3149db95",
      "orgId": "8327d53f-91ea-420a-8613-ba8f3149db95",
...
```

Add a VMware Cloud on AWS Cloud Account with a Proxy

Adding a VMware Cloud on AWS cloud account with a proxy requires manual deployment of a cloud proxy VM. Then you make a POST request with a request body that includes the data collector ID along with parameters specific to VMware Cloud on AWS.

Prerequisites

- Verify that all general prerequisites and prerequisites for the Automation Assembler Infrastructure as a Service (IaaS) service have been satisfied. See Chapter 3 Prerequisites for API Use Case Examples.
- Verify that a cloud proxy VM has been manually deployed.
- Verify that you have the following parameters for the new cloud account:
  - VMC API token.
  - SDDC name.
  - vCenter private IP.
  - NSX Manager IP.
  - vCenter user name.
  - vCenter password.

- vCenter data center ID.

Procedure

1 List all cloud proxies.

```
curl -X GET -H 'Content-Type: application/json' -H "Authorization: Bearer $access_token"
"$url/iaas/api/data-collectors?api_version=$api_version" | jq "."
```

2 Examine the response and assign the data collector variable.

```
data_collector_id='<your_datacollector_id>'
```

3 Assign the account variables for the VMC cloud account.

```
vmc_api_token='<your_vmc_api_token>'
vmc_sddc_name='<your_vmc_sddc_name>'
vmc_vcenter_private_ip='<your_vcenter_private_ip>'
vmc_nsx_manager_ip='<your_nsx_manager_ip>'
vmc_vcenter_username='<your_vcenter_username>'
vmc_vcenter_password='<your_vcenter_password>'
vmc_vcenter_datacenter_id='<your_datacenter_id>'
```

4 Submit a request to create a VMC cloud account.

```
curl -X POST \
  "$url/iaas/api/cloud-accounts?apiVersion=$api_version" \
  -H "Authorization: Bearer $access_token" \
  -H 'Content-Type: application/json' \
  -d '{
    "name": "vmc-endpoint",
    "description": "VMC cloud account",
    "cloudAccountType": "vmc",
    "privateKeyId": "'"$vmc_vcenter_username"'",
    "privateKey": "'"$vmc_vcenter_password"'",
    "cloudAccountProperties": {
        "sddcId": "'"$vmc_sddc_name"'",
        "apiKey": "'"$vmc_api_token"'",
        "hostName": "'"$vmc_vcenter_private_ip"'",
        "nsxHostName": "'"$vmc_nsx_manager_ip"'",
        "dcId": "'"$vmc_data_collector_id"'",
        "acceptSelfSignedCertificate": "false"
    },
    "regionIds": [
        "'"$vmc_vcenter_datacenter_id"'"
    ]
}' | jq "."
```

The response includes a selfLink value.

5 Assign the selfLink variable.

```
selfLink_id='example-selfLink-alphanumeric-string'
```

6 Use the selfLink variable to track the request.

```
curl -X GET -H 'Content-Type: application/json' -H "Authorization: Bearer $access_token"
"$url/iaas/api/request-tracker/$selfLink_id | jq "."
```

The VMware Cloud on AWS cloud account is created when the response shows `"status": "FINISHED"`.

Example: Create a VMC Cloud Account with a Proxy

This example creates a VMware Cloud on AWS cloud account with a cloud proxy VM that has been manually deployed.

Assign the required variables.

```
$ url='https://appliance.domain.com'
$ api_version='2021-07-15'
```

List all cloud proxies.

```
$ curl -X GET -H 'Content-Type: application/json' -H "Authorization: Bearer $access_token"
"$url/iaas/api/data-collectors?api_version=$api_version" | jq "."
```

A snippet of the response from your request shows the data collector IDs.

```
...
    {
      "dcId": "cd7d1eb4-573f-4150-8206-de3d536490ca",
      "ipAddress": "10.139.116.60",
      "name": "localhost.localdom",
      "hostName": "localhost.localdom",
      "status": "ACTIVE"
    },,
...
```

Assign the data collector ID variable.

```
$ data_collector_id='cd7d1eb4-573f-4150-8206-de3d536490ca'
```

Assign the variables for the VMC cloud account.

```
$ vmc_data_collector_id=a1235a7f-d49f-4365-8ed9-2d7d0805e4bc
$ vmc_api_token=ab392fba-32a8-49a5-a084-d422fa32c5b8
$ vmc_sddc_name=MYCOM-PRD-NSXT-M7GA-052019
$ vmc_vcenter_private_ip=10.70.57.196
$ vmc_nsx_manager_ip=10.70.57.131
$ vmc_vcenter_username=cloudadmin@vmc.local
$ vmc_vcenter_password=aBcqCW+m4+XEQg7
$ vmc_vcenter_datacenter_id=Datacenter:datacenter-1
```

Create a VMC cloud account named **demo-vmc-account**.

```
$ curl -X POST \
  "$url/iaas/api/cloud-accounts?apiVersion=$api_version" \
  -H "Authorization: Bearer $access_token" \
  -H 'Content-Type: application/json' \
  -d '{
    "name": "demo-vmc-account",
    "description": "VMC cloud account",
    "cloudAccountType": "vmc",
    "privateKeyId": "'"$vmc_vcenter_username"'",
    "privateKey": "'"$vmc_vcenter_password"'",
    "cloudAccountProperties": {
        "sddcId": "'"$vmc_sddc_name"'",
        "apiKey": "'"$vmc_api_token"'",
        "hostName": "'"$vmc_vcenter_private_ip"'",
        "nsxHostName": "'"$vmc_nsx_manager_ip"'",
        "dcId": "'"$vmc_data_collector_id"'",
        "acceptSelfSignedCertificate": "false"
    },
    "regionIds": [
        "'"$vmc_vcenter_datacenter_id"'"
    ]
}' | jq "."
```

The response includes a selfLink variable.

```
{
    "progress": 0,
    "status": "INPROGRESS",
    "name": "Cloud account creation/update",
    "id": "0dc374ba-08ec-4422-8615-24f4f94ef5aa",
    "selfLink": "/iaas/api/request-tracker/0dc374ba-08ec-4422-8615-24f4f94ef5aa"
}
```

Assign the selfLink variable.

```
selfLink_id='0dc374ba-08ec-4422-8615-24f4f94ef5aa'
```

Use the selfLink variable to track the request.

```
curl -X GET -H 'Content-Type: application/json' -H "Authorization: Bearer $access_token"
"$url/iaas/api/request-tracker/$selfLink_id | jq "."
```

When the request completes successfully, the response shows the cloud account ID at the end of the `resources` path.

```
{
    "progress": 100,
    "status": "FINISHED",
    "resources": [
        "/iaas/api/cloud-accounts/e6af1aa6-dc7a-4847-8adc-c4c73727e5b3"
    ],
```

```
    "name": "Cloud account creation/update",
    "id": "a85d1476-1b11-45b0-8d14-91951385c95d",
    "selfLink": "/iaas/api/request-tracker/a85d1476-1b11-45b0-8d14-91951385c95d"
}
```

Add a Microsoft Azure Cloud Account

To create a Microsoft Azure cloud account, you make a POST request. The request body includes parameters specific to Microsoft Azure that are required to create the cloud account.

As an alternative to using the `cloud-accounts` API call, you can use a `cloud-accounts-azure` API call that creates a Microsoft Azure cloud account with fewer input parameters.

Prerequisites

- Verify that all general prerequisites and prerequisites for the Automation Assembler Infrastructure as a Service (IaaS) service have been satisfied. See Chapter 3 Prerequisites for API Use Case Examples.
- Verify that you have the following parameters for the new cloud account:
  - Microsoft Azure subscription ID
  - Microsoft Azure tenant ID
  - Microsoft Azure client application ID
  - Microsoft Azure client application secret key

Procedure

1 Assign the Microsoft Azure account variables.

```
azure_subscription_id='<your_azure_subscription_id>'
azure_tenant_id='<your_azure_tenant_id>'
azure_client_application_id='<your_azure_client_application_id>'
azure_client_application_secret_key='<your_azure_client_application_secret_key>'
```

2 Submit a request to create a Microsoft Azure cloud account with default cloud zones.

```
curl -X POST \
  "$url/iaas/api/cloud-accounts?apiVersion=$api_version" \
  -H 'Content-Type: application/json' \
  -H "Authorization: Bearer $access_token" \
  -d '{
    "cloudAccountType":"azure",
    "privateKeyId":"'$azure_client_application_id'",
    "privateKey":"'$azure_client_application_secret_key'",
    "cloudAccountProperties": {
      "userLink":"'$azure_subscription_id'",
      "azureTenantId":"'$azure_tenant_id'"
    },
    "regionIds":["<your_region_id>"],
```

```
    "createDefaultZones":true,
    "name":"<your_azure_cloud_account>",
    "description": "This is a demo Azure cloud account",
  }' | jq "."
```

3 (Optional) Submit a request to create a Microsoft Azure cloud account with the `cloud-accounts-azure` API call.

```
curl -X POST \
  "$url/iaas/api/cloud-accounts-azure?apiVersion=$api_version" \
  -H 'Content-Type: application/json' \
  -H "Authorization: Bearer $access_token" \
  -d '{
    "name":"<your_azure_cloud_account>",
    "description": "This is a demo Azure cloud account",
    "subscriptionId": "'$azure_subscription_id'",
    "tenantId": "'$azure_tenant_id'",
    "clientApplicationId": "'$azure_client_application_id'",
    "clientApplicationSecretKey": "'$azure_client_application_secret_key'",
    "regionIds": [ "<your_region_id1>", "<your_region_id2>" ],
    "createDefaultZones": true,
    "tags": [ { "key": "env", "value": "dev" } ]
  }' | jq "."
```

4 List all cloud accounts.

```
curl -X GET $url/iaas/api/cloud-accounts?apiVersion=$api_version -H 'Content-Type:
application/json' -H "Authorization: Bearer $access_token" | jq "."
```

5 Examine the response and verify that the name and ID of the Microsoft Azure cloud account you created is listed.

Example: Create a Microsoft Azure Cloud Account

Assign the required variables.

```
$ url='https://appliance.domain.com'
$ api_version='2021-07-15'
$ azure_subscription_id='r1e31415-4a08-4072-be4a-19de37d12345'
$ azure_tenant_id='s39138ca-3abc-4b4a-a4d6-cd92d9dd62f0'
$ azure_client_application_id='te21wxyz-b183-42ac-cd84-3c4a2459b9a9'
$ azure_client_application_secret_key='udv6lY8MwpP5ABCDfSztP3ABCDEaLMNOPQRmDEUeiI0='
```

Create a cloud account named **`demo-azure-account`**.

```
$ curl -X POST \
  "$url/iaas/api/cloud-accounts?apiVersion=$api_version" \
  -H 'Content-Type: application/json' \
  -H "Authorization: Bearer $access_token" \
  -d '{
    "cloudAccountType":"azure",
    "privateKeyId":"'$azure_client_application_id'",
    "privateKey":"'$azure_client_application_secret_key'",
    "cloudAccountProperties": {
```

```
      "userLink":"'$azure_subscription_id'",
      "azureTenantId":"'$azure_tenant_id'"
    },
    "regionIds":["eastus"],
    "createDefaultZones":true,
    "name": "demo-azure-account",
    "description": "This is a demo Azure cloud account",
  }' | jq "."
```

A snippet of the response from your request shows the account ID.

```
...
"tags": [],
"name": "demo-azure-account",
"id": "c8c3c9bfdb449475-7f703c5265a63d87-
f8e705d89b2569e1aac66c6d00bf4fc7ef4b1c44100f0e944af31eb8ba3d2a5a-
f4226a20b65c4675574bc5fbff6c0",
"updatedAt": "2022-04-02",
"organizationId": "8327d53f-91ea-420a-8613-ba8f3149db95",
...
```

Add a Google Cloud Platform Cloud Account

To create a Google Cloud Platform (GCP) cloud account, you make a POST request. The request body includes parameters specific to Google Cloud Platform that are required to create the cloud account.

Prerequisites

- Verify that all general prerequisites and prerequisites for the Automation Assembler Infrastructure as a Service (IaaS) service have been satisfied. See Chapter 3 Prerequisites for API Use Case Examples.
- Verify that you have the following parameters for the new cloud account:
  - GCP project ID
  - GCP private key ID
  - GCP private key
  - GCP client email

Procedure

1 Assign the GCP account variables.

```
gcp_project_id='<your_gcp_projct_id>'
gcp_private_key_id='<your_gcp_private_key_id>'
gcp_private_key='<your_gcp_private_key>'
gcp_client_email='<your_gcp_client_email>'
```

2 Submit a request to create a GCP cloud account with default cloud zones.

```
curl -X POST \
  "$url/iaas/api/cloud-accounts?apiVersion=$api_version" \
  -H 'Content-Type: application/json' \
  -H "Authorization: Bearer $access_token" \
  -d '{
    "cloudAccountType":"gcp",
    "privateKeyId":"'$gcp_private_key_id'",
    "privateKey":"'$gcp_private_key'",
    "cloudAccountProperties":{
      "projectId":"'$gcp_project_id'",
      "clientEmail":"'$gcp_client_email'"
    },
    "regionIds":["<your_region_id>"],
    "createDefaultZones":true,
    "name":"<your_gcp_cloud_account>",
    "description": "This is a demo GCP cloud account",
  }' | jq "."
```

3 List all cloud accounts.

```
curl -X GET $url/iaas/api/cloud-accounts?apiVersion=$api_version -H 'Content-Type:
application/json' -H "Authorization: Bearer $access_token" | jq "."
```

4 Examine the response and verify that the name and ID of the GCP cloud account you created is listed.

Example: Create a Google Cloud Platform Cloud Account

Assign the required variables.

```
$ url='https://appliance.domain.com'
$ api_version='2021-07-15'
$ gcp_project_id='Example-e2e'
$ gcp_private_key_id='defg3c20c85abcde6a95b44222c4c1d68554b87e'
$ gcp_private_key='-----BEGIN PRIVATE KEY-----\\nMIIEvQIBADANBgkqhkiG9w0BAQEFAA
...
3izE4KDeebLh7SkWFbUt7lFW25UL20\\nKAY7FRTKpvbO+6Z/BnVePVI=\\n-----END PRIVATE KEY-----\\n'
$ gcp_client_email='123456789123-example@developer.gserviceaccount.com'
```

Create the cloud account.

```
$ curl -X POST \
  "$url/iaas/api/cloud-accounts?apiVersion=$api_version" \
  -H 'Content-Type: application/json' \
  -H "Authorization: Bearer $access_token" \
  -d '{
    "cloudAccountType":"gcp",
    "privateKeyId":"'$gcp_private_key_id'",
    "privateKey":"'$gcp_private_key'",
    "cloudAccountProperties":{
      "projectId":"'$gcp_project_id'",
      "clientEmail":"'$gcp_client_email'"
```

```
    },
    "regionIds":["us-west2"],
    "createDefaultZones":true,
    "name":"demo-gcp-account",
    "description": "This is a demo GCP cloud account",
  }' | jq "."
```

A snippet of the response from your request shows the account ID.

```
...
"tags": [],
"name": "demo-GCP-account",
"id": "c8c3c9bfdb449475-7f703c5265a63d87-
f8e705d89b2569e1aac66c6d00bf4fc7ef4b1c44100f0e944af31eb8ba3d2a5a-
f4226a20b65c4675574bc5fbff6c0",
"updatedAt": "2022-04-02",
"organizationId": "8327d53f-91ea-420a-8613-ba8f3149db95",
...
```

Integrating with other applications

To fully automate your organization setup, you can use the IaaS APIs to list, create, update, and delete VMware Aria Automation integrations with other applications programmatically.

Using different input variables, you can create integrations with external systems such as GitHub, Ansible, and external IPAM providers. The following table lists all supported integrations.

Integration	Type	What do I need to create an integration?	Additional Product Documentation
Active Directory	**activedirectory**	■ LDAP connection to Active Directory server. ■ Project configured with appropriate cloud zones, and image and flavor mappings to use with the Active Directory integration.	How do I create an Active Directory integration in Automation Assembler
Ansible	**ansible**	■ Ansible control machine running Ansible version 2.6.0 or later. ■ Read/write access to the directory where the Ansible inventory file is located. ■ Deactivated host key. ■ Vault password set.	Configure Ansible Open Source integration in Automation Assembler
Ansible Tower	**ansible.tower**	Credentials and templates in Ansible Tower configured to use with deployments.	Configure Ansible Tower Integration in Automation Assembler

Integration	Type	What do I need to create an integration?	Additional Product Documentation
Bitbucket	`org.bitbucket`	Personal access API token for Bitbucket.	Configure Bitbucket integration in Automation Assembler
GitHub	`com.github.saas`	Personal access API token for GitHub. See https://docs.gitlab.com/ee/user/profile/personal_access_tokens.html	Configure GitHub integration in Automation Assembler
GitLab	`com.gitlab.saas`	Personal access API token for GitLab.	Configure GitLab cloud template integration in Automation Assembler
IPAM	`ipam`	■ IPAM provider package. ■ Administrator credentials for the account with the external IPAM provider.	How to configure an external IPAM integration in VMware Aria Automation
My VMware	`my-vmware`	Account registered at https://my.vmware.com/	Configure MyVMware integration in Automation Assembler
Puppet	`puppet`	Puppet master name and hostname or IP address of the master.	Configure Puppet Enterprise integration inAutomation Assembler
Red Hat OpenShift	`cmx.openshift-endpoint`	Red Hat OpenShift server configured. **Note** If needed, you can create an OpenShift cluster with a cloud template that VMware provides. See https://flings.vmware.com/red-hat-openshift-container-platform-as-a-service-on-vrealize-automation-cloud.	Configure Red Hat OpenShift Integration in Automation Assembler
SaltStack	`saltstack`	■ Salt master used in the Automation Config integration that contains the Master Plugin. ■ Automation Config service administrator role in VMware Aria Automation. ■ Automation Assembler service administrator role in VMware Aria Automation.	Create an Automation Config integration in VMware Aria Automation

Integration	Type	What do I need to create an integration?	Additional Product Documentation
SDDC Manager	`sddc`	SDDC manager 4.1 or later installed.	Configure a VMware SDDC Manager integration
Terraform Runtime	`terraform.runtime`	Terraform runtime environment. See Preparing an Automation Assembler Terraform runtime environment.	Preparing for Terraform configurations in Automation Assembler
VMware Enterprise PKS	`cmx.pks-endpoint`	■ Pivotal Container Service (PKS) server configured with UAA authentication. ■ Cloud administrator credentials.	Configure VMware Tanzu Kubernetes Grid Integrated Edition Integration in vAutomation Assembler
VMware Aria Automation Orchestrator	`vro`	■ VMware Aria Automation Orchestrator configured. ■ VMware Aria Automation Orchestrator URL ■ Cloud Extensibility proxy deployed.	Configure a VMware Aria Automation Orchestrator integration in Automation Assembler
VMware Aria Operations	`vrops`	A local or non-local VMware Aria Operations login account with read-only privileges to the vCenter adapter instance of the vSphere endpoint. **Note** For non-local account login, username format is username@domain@authenticated-source, for example `jdoe@company.com@workspaceone`	Integrating with VMware Aria Operations

Create an Integration with Github

To create an integration with GitHub, you make a POST request. The request body includes properties specific to GitHub.

Prerequisites

- Verify that all general prerequisites and prerequisites for the Automation Assembler Infrastructure as a Service (IaaS) service have been satisfied. See Chapter 3 Prerequisites for API Use Case Examples.

- Verify that you have a personal access token for authentication to GitHub. See https://docs.gitlab.com/ee/user/profile/personal_access_tokens.html.

Procedure

1 Assign variables for GitHub.

```
private_key='<your_GitHub_personal_access_token>'
```

2 Submit a request to create a GitHub integration.

```
curl -X POST \
  "$url/iaas/api/integrations?apiVersion=$api_version" \
  -H 'Content-Type: application/json' \
  -H "Authorization: Bearer $access_token" \
  -d '{
    "integrationType":"com.github.saas",
    "name":"<your_github_integration>",
    "privateKey":"'$private_key'",
    "integrationProperties":{
      "url":"https://api.github.com"
    },
  }' | jq "."
```

The response includes a selfLink.

```
{
  "progress": 0,
  "status": "INPROGRESS",
  "name": "Integration creation/update",
  "id": "example-selfLink-alphanumeric-string",
  "selfLink": "/iaas/api/request-tracker/example-selfLink-alphanumeric-string"
}
```

3 Assign the selfLink variable.

```
selfLink_id='example-selfLink-alphanumeric-string'
```

4 Use the selfLink variable to track the request.

```
curl -X GET -H 'Content-Type: application/json' -H "Authorization: Bearer $access_token"
"$url/iaas/api/request-tracker/$selfLink_id?apiVersion=$api_version" | jq "."
```

When the request completes successfully, the response includes a resource with the integration ID.

```
{
  "progress": 100,
  "status": "FINISHED",
  "resources": [
      "/iaas/api/integrations/example-integration-id-string"
  ],
```

```
  "name": "Integration creation/update",
  "id": "example-selfLink-alphanumeric-string",
  "selfLink": "/iaas/api/request-tracker/example-selfLink-alphanumeric-string"
}
```

5 Assign the integration ID variable.

```
integration_id='example-integration-alphanumeric-string'
```

6 Use the integration ID variable to list the integration.

```
curl -X GET -H 'Content-Type: application/json' -H "Authorization: Bearer $access_token"
"$url/iaas/api/integrations/$integration_id?apiVersion=$api_version" | jq "."
```

7 Examine the response and verify that the name and ID of the integration that you created is listed.

Example: Create a GitHub integration

Assign the required variables.

```
$ url='https://appliance.domain.com'
$ api_version='2021-07-15'
$ private_key='8bc9401b5d28f4ec126929af0dc4e99dd0792b0f'
```

Create the integration.

```
$ curl -X POST \
  "$url/iaas/api/integrations?apiVersion=$api_version" \
  -H 'Content-Type: application/json' \
  -H "Authorization: Bearer $access_token" \
  -d '{
    "integrationType":"com.github.saas",
    "name":"Git integration example",
    "privateKey":"'$private_key'",
    "integrationProperties":{
      "url":"https://api.github.com"
    },
  }' | jq "."
```

The response includes a selfLink.

```
{
  "progress": 0,
  "status": "INPROGRESS",
  "name": "Integration creation/update",
  "id": "a0c5eb3a-9ffa-4bfb-b63b-c77510bcc597",
  "selfLink": "/iaas/api/request-tracker/a0c5eb3a-9ffa-4bfb-b63b-c77510bcc597"
}
```

Assign the selfLink variable

```
selfLink_id='a0c5eb3a-9ffa-4bfb-b63b-c77510bcc597'
```

Use the selfLink variable to track the request.

```
curl -X GET -H 'Content-Type: application/json' -H "Authorization: Bearer $access_token"
"$url/iaas/api/request-tracker/$selfLink_id?apiVersion=$api_version" | jq "."
```

When the request completes successfully, the response includes a resource with the integration ID.

```
{
  "progress": 100,
  "status": "FINISHED",
  "resources": [
      "/iaas/api/integrations/e5dda941-bb17-4f19bd15-7db0b8eab88c"
  ],
  "name": "Integration creation/update",
  "id": "a0c5eb3a-9ffa-4bfb-b63b-c77510bcc597",
  "selfLink": "/iaas/api/request-tracker/a0c5eb3a-9ffa-4bfb-b63b-c77510bcc597"
}
```

Assign the integration ID variable

```
integration_id='e5dda941-bb17-4f19bd15-7db0b8eab88c'
```

Use the integration ID variable to list the integration.

```
curl -X GET -H 'Content-Type: application/json' -H "Authorization: Bearer $access_token"
"$url/iaas/api/integrations/$integration_id?apiVersion=$api_version" | jq "."
```

When the request completes successfully, a snippet of the response shows integration details.

```
...
    {
       "integrationType": "com.github.saas",
       "integrationProperties": (
          "url": "https://api.github.com"
       },
       "name":"Git integration example",
       "id: "e5dda941-bb17-4f19bd15-7db0b8eab88c"
       "createdAt":"2022-04-02"
       "updatedAt": "2022-04-02"
       "orgId": "ce811934-ea1a-4f53-6bec-4656ca7d126",
       "_links": {
          "self": {
             "href": "/iaas/api/integrations/e5dda941-bb17-4f19bd15-7db0b8eab88c"
          }
       }
    }
...
```

Delete an Integration

To delete an integration, you make a DELETE request with the ID of the integration.

The following procedure shows how to delete an integration including an optional step to list the integration details before deleting the integration. It is a good practice to check the details of the integration so that you delete the correct integration.

Prerequisites

- Verify that all general prerequisites and prerequisites for the Automation Assembler Infrastructure as a Service (IaaS) service have been satisfied. See Chapter 3 Prerequisites for API Use Case Examples.
- Verify that you have the ID of the integration that you want to delete. See Create an Integration with Github.

Procedure

1 Assign your integration ID variable.

```
integration_id='<your_integration_id>'
```

2 (Optional) List the integration before deleting.

```
curl -X GET -H 'Content-Type: application/json' -H "Authorization: Bearer $access_token"
"$url/iaas/api/integrations/$integration_id?apiVersion=$api_version" | jq "."
```

3 (Optional) Examine the response to verify the integration details, such as integration name and integration type.

4 Submit a request to delete the integration.

```
curl -X DELETE -H 'Content-Type: application/json' -H "Authorization: Bearer
$access_token" "$url/iaas/api/integrations/$integration_id?apiVersion=$api_version" | jq
"."
```

The response includes a selfLink.

```
{
  "progress": 0,
  "status": "INPROGRESS",
  "name": "Integration deletion",
  "id": "example-selfLink-alphanumeric-string",
  "selfLink": "/iaas/api/request-tracker/example-selfLink-alphanumeric-string"
}
```

5 Assign the selfLink variable.

```
selfLink_id='example-selfLink-alphanumeric-string'
```

6 Use the selfLink variable to track the request.

```
curl -X GET -H 'Content-Type: application/json' -H "Authorization: Bearer $access_token"
"$url/iaas/api/request-tracker/$selfLink_id | jq "."
```

When the request completes successfully, the response shows the status as `FINISHED`.

```
{
  "progress": 100,
  "message": "Deleted",
  "status": "FINISHED",
  "name": "Integration deletion",
  "id": "example-selfLink-alphanumeric-string",
  "selfLink": "/iaas/api/request-tracker/example-selfLink-alphanumeric-string"
}
```

Example: Delete an integration

Assign the required variables.

```
$ url='https://appliance.domain.com'
$ api_version='2021-07-15'
$ integration_id='e5dda941-bb17-4f19bd15-7db0b8eab88c'
```

Delete the integration.

```
curl -X DELETE -H 'Content-Type: application/json' -H "Authorization: Bearer $access_token"
"$url/iaas/api/integrations/$integration_id?apiVersion=$api_version" | jq "."
```

The response includes a selfLink.

```
{
  "progress": 0,
  "status": "INPROGRESS",
  "name": "Integration deletion",
  "id": "fe472e75-f346-4de7-bbc4-5edddefd9dfa",
  "selfLink": "/iaas/api/request-tracker/fe472e75-f346-4de7-bbc4-5edddefd9dfa"
}
```

Assign the selfLink variable

```
selfLink_id='fe472e75-f346-4de7-bbc4-5edddefd9dfa'
```

Use the selfLink variable to track the request.

```
curl -X GET -H 'Content-Type: application/json' -H "Authorization: Bearer $access_token"
"$url/iaas/api/request-tracker/$selfLink_id | jq "."
```

When the request completes successfully, the response shows the status as `FINISHED`.

```
{
  "progress": 100,
  "message": "Deleted",
  "status": "FINISHED",
  "name": "Integration deletion",
  "id": "fe472e75-f346-4de7-bbc4-5edddefd9dfa",
  "selfLink": "/iaas/api/request-tracker/fe472e75-f346-4de7-bbc4-5edddefd9dfa"
}
```

How do I import an IPAM package

To import an IPAM package using the IaaS API, you reserve enough space for the package then you make multiple PATCH requests to import smaller pieces of the package. After importing all the pieces, you use a POST request to upload the file onto the server.

The IaaS API implements the TUS RFC protocol which provides a mechanism for resumable file uploads. By dividing a large package into smaller ZIP packages and importing the packages individually, you ensure that you will not exceed the server's file size limit with your upload.

For more information about the TUS RFC protocol, see https://github.com/tus/tus-resumable-upload-protocol/blob/main/protocol.md.

This procedure shows how to import your IPAM package using the integrations IPAM APIs. Then you complete the integration using the Automation Assembler UI or the integrations API.

Prerequisites

- Verify that all general prerequisites and prerequisites for the Automation Assembler IaaS API have been satisfied. See Chapter 3 Prerequisites for API Use Case Examples.
- Verify that you know the size of the package that you want to upload. If the package is large, divide it into smaller packages. The following example divides the large package into smaller packages that are each 9000000 bytes in size.

  ```
  split -b 9000000 <your_package>.zip <your_package>.zip_split
  ```

- If using the API to complete the IPAM integration with a cloud account, verify that you have the following parameters for your cloud account:
  - *privateKeyId* used to create your cloud account.
  - *privateKey* used to create your cloud account.
  - *hostName* used to create your cloud account.
  - *faasProviderEndpointID* is your cloud account ID.

Procedure

1 Reserve space for the package with `Upload-Length` equal to the size of the package in bytes.

```
curl -X POST \
  "$url/iaas/api/integrations-ipam/package-import?apiVersion=$api_version" \
  -H "Authorization: Bearer $access_token" \
  -H "Upload-Length: <package_size_in_bytes>" \
  -H "Tus-Resumable: 1.0.0" | jq "."
```

Examine the response header to get the location for the package.

Note You must complete the package import process before the location reservation expires in approximately two hours.

2 Assign a variable for the location ID.

```
location_id = '<package_location>'
```

3 Import the first small package with `Upload-Offset` equal to zero.

```
curl -X PATCH \
  "$url/iaas/api/integrations-ipam/package-import/$location_id/?apiVersion=$api_version" \
  -H "Authorization: Bearer $access_token" \
  -H "Tus-Resumable: 1.0.0" \
  -H "Content-Type: application/offset+octet-stream" \
  -H "Upload-Offset: 0" | jq "."
```

A successful response returns `204 No Content` and includes an upload offset number. Use that number in the PATCH request for the next package.

4 Import the next small package.

- You must import the small packages in a sequential order.
- `Upload-Offset` is the offset value from the response header of the previous PATCH request.

```
curl -X PATCH \
  "$url/iaas/api/integrations-ipam/package-import/$location_id/?apiVersion=$api_version" \
  -H "Authorization: Bearer $access_token" \
  -H "Tus-Resumable: 1.0.0" \
  -H "Content-Type: application/offset+octet-stream" \
  -H "Upload-Offset: <upload_offset_from_previous_response>" | jq "."
```

Verify that the successful response returns `204 No Content` and a larger upload offset number. Continue making PATCH requests with the upload offset numbers from the previous response until all the smaller packages have been imported.

Note If you do not know the current upload offset number, make a HEAD request with the location ID.

```
curl -X HEAD \
  "$url/iaas/api/integrations-ipam/package-import/$location_id/?apiVersion=$api_version" \
  -H "Authorization: Bearer $access_token" \
  -H "Tus-Resumable: 1.0.0" | jq "."
```

Use the upload offset number in the response for the next PATCH request.

5 After all the small packages have been imported, upload the file.

The input body includes:

- `bundleId` for the location ID.
- `"option": "OVERWRITE"` to overwrite any existing package.

```
curl -X POST \
  "$url/iaas/api/integrations-ipam/package-import?apiVersion=$api_version" \
```

```
  -H "Authorization: Bearer $access_token" \
  -H "Content-Type: application/json" \
  -H "Tus-Resumable: 1.0.0" \
  -d '{
    "bundleId":"$location_id",
    "option" : "OVERWRITE"
    }
| jq "."
```

A successful response includes the Provider ID, Provider name, and Provider version. To complete an IPAM integration, you can use the Automation Assembler UI or the integrations endpoint in the IaaS API.

6 To complete an integration using the Automation Assembler UI, select **Infrastructure > Integrations > Add integration > IPAM**. On the **New Integration** page that appears, select **Manage IPAM Providers** and choose the package with the Provider name and Provider version from the API response.

7 (Optional) To complete an integration using the API:

a Assign a variable for the provider ID.

```
provider_id = '<provider_id_from_package_upload>'
```

b Create a new IPAM integration.

```
curl -X POST \
  "$url/iaas/api/integrations?apiVersion=$api_version" \
  -H "Authorization: Bearer $access_token" \
  -H "Content-Type: application/json" \
  -d '{
    "integrationProperties": {
      "providerId" : "'$provider_id'",
      "faasProviderEndpointId": "<your_provider_endpoint_ID>",
      "privateKeyId": "<your_privateKeyId>",
      "privateKey": "<your_privateKey>",
      "hostName": "<your_hostName>",
      "dcId": "onprem"
    },
    "customProperties": {"isExternal": "true"},
    "integrationType": "ipam",
    "associatedCloudAccountIds": [],
    "associatedMobilityCloudAccountIds": {},
    "privateKey": "<your_privateKey>",
    "privateKeyId": "<your_privateKeyId>"
  } | jq "."
```

The response includes a self link to track the request.

c Assign the selfLink variable.

```
selfLink_id='<example_selfLink_alphanumeric_string>'
```

d Use the selfLink variable to track the request.

```
curl -X GET -H 'Content-Type: application/json' -H "Authorization: Bearer
$access_token" "$url/iaas/api/request-tracker/$selfLink_id?apiVersion=$api_version" |
jq "."
```

The integration is created with the response includes `"status": "FINISHED"`.

Example: Import an IPAM Package

This example shows how to import an Infoblox IPAM package with a total size of 73342782 bytes. To import incrementally using smaller packages, you divide the larger package into smaller packages. This example assumes that you have divded the package into multiple smaller packages each 9000000 bytes in size.

After importing the IPAM package, you create an integration with a cloud account using the the Automation Assembler UI or the integrations endpoint in the IaaS API. This example shows how to integrate the uploaded IPAM package with a cloud account that has the following parameters:

- faasProviderEndpointId: "f46337e9-e0c2-4ada-9244-766a8e54da48"
- hostName: "infoblox.sof-mbu.eng.mycompany.com"
- privateKey: "Password!23"
- privateKeyId: "administrator@mycompany.local"

Assign variables.

```
$ url='https://appliance.domain.com'
$ api_version='2021-07-15'
```

Reserve space for the package.

```
$ curl -X POST \
  "$url/iaas/api/integrations-ipam/package-import?apiVersion=$api_version" \
  -H "Authorization: Bearer $access_token" \
  -H "Upload-Length: 73342782" \
  -H "Tus-Resumable: 1.0.0" | jq "."
```

Examine the response to get the location ID.

```
status: 201
body: empty
headers: {location:/iaas/api/integrations-ipam/package-import/8a81db06-f27f-4ccb-
a7c5-960c35107ed6, ...}
```

Assign the location ID variable.

```
$ location_id='8a81db06-f27f-4ccb-a7c5-960c35107ed6'
```

Import the first package with a content length of 9000000 bytes and a zero upload offset value.

```
$ curl -X PATCH \
  "$url/iaas/api/integrations-ipam/package-import/$location_id/?apiVersion=$api_version" \
  -H "Authorization: Bearer $access_token" \
  -H "Content-Length: 9000000" \
  -H "Tus-Resumable: 1.0.0" \
  -H "Content-Type: application/offset+octet-stream" \
  -H "Upload-Offset: 0" | jq "."
```

Examine the header in the response to get the upload offset value.

```
status: 204
body: empty
headers: {upload-offset:9000000, ...}
```

Import the next package.

```
$ curl -X PATCH \
  "$url/iaas/api/integrations-ipam/package-import/$location_id/?apiVersion=$api_version" \
  -H "Authorization: Bearer $access_token" \
  -H "Content-Length: 9000000" \
  -H "Tus-Resumable: 1.0.0" \
  -H "Content-Type: "application/offset+octet-stream" \
  -H "Upload-Offset: 9000000" | jq "."
```

Examine the header in the response again to get the next upload offset value.

```
status: 204
body: empty
headers: {upload-offset:18000000, ...}
```

Import the next package.

```
$ curl -X PATCH \
  "$url/iaas/api/integrations-ipam/package-import/$location_id/?apiVersion=$api_version" \
  -H "Authorization: Bearer $access_token" \
  -H "Content-Length: 9000000" \
  -H "Tus-Resumable: 1.0.0" \
  -H "Content-Type: "application/offset+octet-stream" \
  -H "Upload-Offset: 18000000" | jq "."
```

Repeat the process to get the upload offset value from the header response and use that value to import the next package.

After importing the final small package, upload the complete package.

```
$ curl -X POST \
  "$url/iaas/api/integrations-ipam/package-import?apiVersion=$api_version" \
```

```
  -H "Authorization: Bearer $access_token" \
  -H "Tus-Resumable: 1.0.0" \
  -d '{
    "bundleId":"$location_id",
    "option" : "OVERWRITE"
    }
| jq "."
```

A successful response includes the provider ID, provider name, and the provider version

```
{
  "providerId": "86801580-9042-49b6-879d-5a7361d33519",
  "providerName": "Infoblox",
  "providerVersion": "1.6"
  "logoIcon":"iVBORw...",
...
```

To create an IPAM integration using the Automation Assembler UI, select **Infrastructure > Integrations > Add integration > IPAM**. On the **New Integration** page that appears, select **Manage IPAM Providers** and choose the package with the Provider name `Infoblox` and Provider version `1.6`.

To create an IPAM integration using the API, assign the provider ID variable.

```
$ provider_id = "86801580-9042-49b6-879d-5a7361d33519"
```

Create the integration.

```
$ curl -X POST \
  "$url/iaas/api/integrations?apiVersion=$api_version" \
  -H "Authorization: Bearer $access_token" \
  -H "Content-Type: application/json" \
  -d '{
    "integrationProperties": {
      "providerId" : "'$provider_id'",
      "faasProviderEndpointId": "f46337e9-e0c2-4ada-9244-766a8e54da48",
      "privateKeyId": "administrator@mycompany.local",
      "privateKey": "Password!23",
      "hostName": "infoblox.sof-mbu.eng.mycompany.com",
      "dcId": "onprem"
    },
    "customProperties": {"isExternal": "true"},
    "integrationType": "ipam",
    "associatedCloudAccountIds": [],
    "associatedMobilityCloudAccountIds": {},
    "privateKey": "Password!23",
    "privateKeyId": "administrator@mycompany.local"
  } | jq "."
```

Examine the response.

```
{
"progress": 0,
```

```
"status": "INPROGRESS",
"name": "Integration creation/update",
"id": "117ff057-9e26-4f0c-ae7b-eb9fcc1c15cc",
"selfLink": "/iaas/api/request-tracker/117ff057-9e26-4f0c-ae7b-eb9fcc1c15cc"
}
```

Assign the selfLink variable.

```
$ selfLink_id='117ff057-9e26-4f0c-ae7b-eb9fcc1c15cc'
```

Use the selfLink for tracking.

```
$ curl -X GET -H 'Content-Type: application/json' -H "Authorization: Bearer $access_token"
"$url/iaas/api/request-tracker/$selfLink_id?apiVersion=$api_version" | jq "."
```

The integration is complete when the response includes `"status": "FINISHED"`.

```
{
"progress": 100,
"status": "FINISHED",
"resources": [
     "/iaas/api/integrations/9df2b0a8-2ce6-4465-bf39-04290965da9e"     ],
     "name": "Integration creation/update",
     "id": "117ff057-9e26-4f0c-ae7b-eb9fcc1c15cc",
     "selfLink": "/iaas/api/request-tracker/117ff057-9e26-4f0c-ae7b-eb9fcc1c15cc"
}
```

Using Automation Assembler APIs to Build your Resource Infrastructure

6

You use the Automation Assembler APIs to add a project and set up an infrastructure for the team members in Automation so that new workloads can be provisioned.

Read the following topics next:

- Create a Cloud Zone
- Create a Project to use in Automation Assembler
- Add a Cloud Zone to Your Project
- Create Flavor Mappings
- Create Image Mappings
- Working with Networks
- Creating Storage Profiles

Create a Cloud Zone

To create a cloud zone, you first make GET request to obtain a region ID with a cloud account ID as input. Then you make a POST request with the region ID.

Prerequisites

- Verify that all general prerequisites and prerequisites for the Automation Assembler Infrastructure as a Service (IaaS) service have been satisfied. See Chapter 3 Prerequisites for API Use Case Examples.
- Verify that you have the cloud account ID for the cloud account you added. See Add an Amazon Web Services Cloud Account .

Procedure

1 Assign the cloud account ID variable.

```
cloud_account_id='<your_cloud_account_id>'
```

2 Look up the IDs for the region named **us-east-1** that is associated with the cloud account.

```
curl -X GET -H 'Content-Type: application/json' -H "Authorization: Bearer $access_token"
"$url/iaas/api/regions/?apiVersion=$api_version&"'$filter'"=externalRegionId%20eq%20'us-
east-1'%20and%20cloudAccountId%20eq%20'"$cloud_account_id"'"  | jq "."
```

3 To obtain a region ID, examine the response.

4 Assign the region ID variable.

```
region_id='<your_region_id>'
```

5 Create a cloud zone.

```
curl -X POST -H 'Content-Type: application/json' -H "Authorization: Bearer
$access_token" -d '{ "name": "Demo-zone-1", "description": "This zone is for
Demo", "regionId": "'$region_id'", "placementPolicy": "DEFAULT" }' "$url/iaas/api/zones?
apiVersion=$api_version"  | jq "."
```

6 To obtain a cloud zone ID, examine the response.

7 Assign the zone ID variable.

```
zone_id='<your_cloud_zone_id>'
```

8 List all cloud zones.

```
curl -X GET -H 'Content-Type: application/json' -H "Authorization: Bearer $access_token"
"$url/iaas/api/zones?apiVersion=$api_version"  | jq "."
```

9 (Optional) List the cloud zone with your cloud zone ID.

```
curl -X GET -H 'Content-Type: application/json' -H "Authorization: Bearer $access_token"
"$url/iaas/api/zones/$zone_id?apiVersion=$api_version"  | jq "."
```

10 Examine the output to verify that the cloud zone you created is listed.

Example: Create a Cloud Zone

Assign variables, specify a cloud account and look up IDs for the region named **us-east-1** associated with the cloud account.

```
$ url='https://appliance.domain.com'
$ api_version='2021-07-15'
$ cloud_account_id='c8c3c9bfdb449475-7f703c5265a63d87-
f8e705d89b2569e1aac66c6d00bf4fc7ef4b1c44100f0e944af31eb8ba3d2a5a-
f4226a20b65c4675574bc5fbff6c0'
$ curl -X GET -H 'Content-Type: application/json' -H "Authorization: Bearer $access_token"
"$url/iaas/api/regions/?apiVersion=$api_version&"'$filter'"=externalRegionId%20eq%20'us-
east-1'%20and%20cloudAccountId%20eq%20'"$cloud_account_id"'"  | jq "."
```

A snippet of the response from your request shows your cloud account ID with a region ID.

```
...
"externalRegionId": "us-east-1",
"cloudAccountId": "c8c3c9bfdb449475-7f703c5265a63d87-
f8e705d89b2569e1aac66c6d00bf4fc7ef4b1c44100f0e944af31eb8ba3d2a5a-
f4226a20b65c4675574bc5fbff6c0",
"id": "4965d34c3bfe0275574bc5fd858e8",
"updatedAt": "2022-04-02",
...
```

Specify a region ID and create a cloud zone.

```
$ region_id='4965d34c3bfe0275574bc5fd858e8'
$ curl -X POST -H 'Content-Type: application/json' -H "Authorization: Bearer
$access_token" -d '{ "name": "Demo-zone-1", "description": "This zone is for
Demo", "regionId": "'$region_id'", "placementPolicy": "DEFAULT" }' "$url/iaas/api/zones?
apiVersion=$api_version"  | jq "."
```

A snippet of the response from your request shows the zone ID.

```
...
"name": "Aws / us-east-1",
"id": "4965d34c3bfe0275574bc5fd8782a",
"updatedAt": "2022-04-02",
...
```

Create a Cloud Zone with a Folder

When creating a cloud zone, you can specify a folder with a path that relates to a datacenter, so that when you deploy a machine you deploy it to that folder.

To create a cloud zone with a folder, you first obtain an external region ID with a cloud account ID as input. Only the vSphere, VMC, and VCF cloud accounts support folders. Then you use the Folders API to get the external ID for the folder to specify.

The following procedure shows how to find a folder that you can use when creating a cloud zone.

Prerequisites

- Verify that all general prerequisites and prerequisites for the Automation Assembler Infrastructure as a Service (IaaS) service have been satisfied. See Chapter 3 Prerequisites for API Use Case Examples.

  Note Users with the following RBAC permissions can also use the folder API to create a cloud zone:

  - provisioning_cloud-zones_read
  - provisioning_cloud-zones_manage

- Verify that you have the cloud account ID for the cloud account with the folder that you want to use. See Add a vSphere Cloud Account .

Procedure

1 Assign the cloud account ID variable.

```
cloud_account_id='<your_cloud_account_id>'
```

2 List the info for the cloud account.

```
curl -X GET -H 'Content-Type: application/json' -H "Authorization: Bearer $access_token"
"$url/iaas/api/cloud-accounts/$cloud_account_id/?apiVersion=$api_version"  | jq "."
```

3 To find the region where you want to create a cloud zone, examine the enabledRegions section of the response and note the external region ID and its ID.

4 Assign the external region ID variable.

```
external_region_id='<your_external_region_id>'
```

```
region_id='<your_region_id>'
```

5 List all folders associated with the external region and the cloud account.

```
curl -X GET -H 'Content-Type: application/
json' -H "Authorization: Bearer $access_token" "$url/
iaas/api/folders?externalRegionid=$external_region_id&cloudaccountid=$cloud_account_id/?
apiVersion=$api_version"  | jq "."
```

Note Query parameters are optional:

- By providing the cloud account ID, you filter the folders by cloud account.
- By providing the external region ID, you filter the folders by datacenter.

6 Examine the response and note the external ID of the folder that you want to use.

Any of the folders listed in the response can be used in the cloud specification.

7 Assign the external ID variable.

```
external_id='<your_external_id>'
```

8 To create a cloud zone with a folder in the datacenter, specify the external ID as the value for the folder.

```
curl -X POST \
  "$url/iaas/api/zones?apiVersion=$api_version" \
  -H 'Content-Type: application/json' \
  -H "Authorization: Bearer $access_token" \
  -d '{
```

```
    "regionId": "'$region_id'",
    "name": "<cloud_zone_with_folder>",
    "folder": "'$external_id'"
  } | jq "."
```

The response shows the folder and cloud zone ID.

Results

Any resources that are deployed in the cloud zone are deployed to the folder.

Example: Create a Cloud Zone with a Folder

Assign variables and specify an ID for a vSphere cloud account.

```
$ url='https://appliance.domain.com'
$ api_version='2021-07-15'
$ cloud_account_id='2636b316-b514-47f3-93b8-78f18df51a29'
```

List the information for the vSphere cloud account.

```
$ curl -X GET -H 'Content-Type: application/json' -H "Authorization: Bearer $access_token"
"$url/iaas/api/cloud-accounts/$cloud_account_id/?apiVersion=$api_version"  | jq "."
```

Examine the response to find the region where you want to create a cloud zone. A snippet of the response shows the enabledRegions with an external region ID and your cloud account ID.

```
...
"enabledRegions": [
    {
        "externalRegionId": "Datacenter:datacenter-3",
        "name": "SDDC-Datacenter",
        "cloudAccountId": "2636b316-b514-47f3-93b8-78f18df51a29",
        "id": "8d633554-90ca-4f8a-8a0f-1cd3cd90a522",
        "updatedAt": "2022-04-02",
...
```

Assign the external region ID variable.

```
$ external_region_id= 'Datacenter:datacenter-3'
$ region_id= '8d633554-90ca-4f8a-8a0f-1cd3cd90a522'
```

List folders associated with the external region and cloud account.

```
$ curl -X GET -H 'Content-Type: application/json' -H "Authorization: Bearer $access_token"
"$url/iaas/api/folders?externalRegionid=$external_region_id&cloudaccountid=$cloud_account_id/?
apiVersion=$api_version"  | jq "."
```

A snippet of the response shows the folders associated with the external region and the cloud account. The external ID is the value that you assign to the folder when creating the cloud zone.

```
...
        "cloudAccountIds": [
```

```
            "2636b316-b514-47f3-93b8-78f18df51a29"
        ]
        "externalId": "dbuyukliiska",
        "name": "dbuyukliiska",
        "id": "3863f2cc-0340-483f-b24e-d351b7c53c8c",
        "createdAt": "2022-04-02",
        "updatedAt": "2022-04-02",
...
```

Assign the external ID.

```
$ external_id='dbuyukliiska'
```

Create a cloud zone with a specified folder.

```
curl -X POST \
  "$url/iaas/api/zones?apiVersion=$api_version" \
  -H 'Content-Type: application/json' \
  -H "Authorization: Bearer $access_token" \
  -d '{
    "regionId": "'$region_id'",
    "name": "Demo-zone-with-folder-1",
    "folder": "'$external_id'"
  } | jq "."
```

A snippet of the response from your request shows the folder and cloud zone ID.

```
...
    "folder": "dbuyukliiska",
    "externalRegionId": "Datacenter:datacenter-3",
    "cloudAccountId": "2636b316-b514-47f3-93b8-78f18df51a29",
    "name": "Demo-zone-with-folder-1",
    "id": "258bccb1-a337-4b63-9a8a-44a414761d93",
    "updatedAt": "2021-11-22",
...
```

Create a Project to use in Automation Assembler

As a Automation Assembler administrator, you make a POST request with a project name to create a project. Then you add members and cloud zones to the project so that project members can deploy cloud templates to the associated zones.

Prerequisites

- Verify that all general prerequisites and prerequisites for the Automation Assembler Infrastructure as a Service (IaaS) service have been satisfied. See Chapter 3 Prerequisites for API Use Case Examples.

- Verify that the project roles that you plan to assign have sufficient permissions to perform project-related tasks.

  Note A user with the project administrator or project member role can perform a limited number of project-related tasks. For a complete list of tasks and roles required, see Organization and service user roles in VMware Aria Automation.

- Prepare parameters including the project name, description, and email addresses for administrators, members, or viewers.

Procedure

1 Assign the project name variable.

```
project_name='<your_project_name>'
```

your_project_name is a name that you choose.

2 Create a project.

```
curl -X POST \
  "$url/iaas/api/projects?apiVersion=$api_version"
  -H 'Content-Type: application/json'
  -H "Authorization: Bearer $access_token"
  -d '{
    "name" : "'$project_name'",
    "description" : "your-description",
    "administrators" : [{ "email" : "<admin_email>", ["type" : <"user" | "group">]}],
    "members" : [{ "email" : "<member_email>", ["type" : <"user" | "group">]}],
    "viewers" : [{ "email" : "<viewer_email>", ["type" : <"user" | "group">]}],
  }' | jq "."
```

   - *admin_email*, *member_email*, and *viewer_email* are email addresses of an administrator, member, and viewer user or name of the group in the project.
   - The type parameter is optional. It assigns the administrator, member, or viewer to a user or group type. If unspecified, the value defaults to `user`.

3 Get a list of projects and filter for the project with *your_project_name*.

```
curl -X GET -H 'Accept: application/json' -H "Authorization: Bearer $access_token" "$url/
iaas/api/projects?apiVersion=$api_version&"'$filter'"=name%20eq%20'$project_name'" | jq "."
```

4 Examine the response and record the ID of your newly created project.

Example: Create a Project to use in Automation Assembler

Create a project named **`Example-Assembler-project`** with administrators, members, and viewers at **`mycompany.com`**. This example assumes that **`Example-project`** does not exist.

```
$ url='https://appliance.domain.com'
$ api_version='2021-07-15'
$ project_name='Example-Assembler-project'
```

Create a project for Automation Assembler.

```
$ curl -X POST \
  "$url/iaas/api/projects?apiVersion=$api_version" \
  -H 'Content-Type: application/json' \
  -H "Authorization: Bearer $access_token" \
  -d '{
    "name" : "'$project_name'",
    "description" : "This is an example project for Automation Assembler",
    "administrators" : [{"email" : "admin1@mycompany.com", "type" : "user"}],
    "members" : [{"email" : "member1@mycompany.com", "type" : "user"}],
    "viewers" : [{"email" : "viewer1@mycompany.com", "type" : "user"} ]
  }' | jq "."
```

The response shows the administrators, members, and viewers related to the project and the project ID.

```
{
  "administrators": [
    {
      "email": "admin1@mycompany.com",
      "type": "user"
    }
  ],
  "members": [
    {
      "email": "member1@mycompany.com",
      "type": "user"
    }
  ],
  "viewers": [
    {
      "email": "viewer1@mycompany.com",
      "type": "user"
    }
  ],
  "sharedResources": true,
  "name": "Example-Assembler-project",
  "description": "This is an example project for Automation Assembler",
  "id": "5944aacb-91de-4541-bb9e-ef2a5403f81b",
  "orgId": "8327d53f-91ea-420a-8613-ba8f3149db95"
}
```

What to do next

Add a cloud zone to your project. See Add a Cloud Zone to Your Project. If you want to add an administrator or user, see Add Users to Your Project.

Add Users to Your Project

As a Automation Assembler user with the project administrator role, you can use PATCH requests to add users and assign roles in your project.

Prerequisites

- Verify that all general prerequisites and prerequisites for the Automation Assembler Infrastructure as a Service (IaaS) service have been satisfied. See Chapter 3 Prerequisites for API Use Case Examples.
- Verify that you have the project administrator role in your project and you have the project ID. See Create a Project to use in Automation Assembler.
- Prepare parameters including additional email addresses for administrators, members, or viewers that you want to add to the project.

Procedure

1 Assign the project ID variable.

```
project_id='<your_project_id>'
```

your_project_id is the ID of the new project you created.

2 List the details of your project.

```
curl -X GET -H 'Accept: application/json' -H "Authorization: Bearer $access_token" "$url/
iaas/api/projects/$project_id?apiVersion=$api_version" | jq "."
```

3 Examine the response to see the administrators and users who are already in your project.

4 Submit a request to add a new administrator that includes the existing administrator for the project.

Note If the call does not include existing administrators for the project, the PATCH request removes those administrators from the project. Specifying the administrator type is optional.

```
curl -X PATCH \
  "$url/iaas/api/projects/$project_id?apiVersion=$api_version" \
  -H 'Content-Type: application/json' \
  -H "Authorization: Bearer $access_token" \
  -d '{
    "administrators" : [
      {"email" : "<your_new_administrator_email>", "type" : "user"},
      {"email" : "<existing_administrator>", "type" : "user"}
    ]
  }' | jq "."
```

5 Submit a request to add a new member that includes the existing users for the project.

Note If the call does not include existing members for the project, the PATCH request removes those members from the project. Specifying the member type is optional.

```
curl -X PATCH \
  "$url/iaas/api/projects/$project_id?apiVersion=$api_version" \
  -H 'Content-Type: application/json' \
  -H "Authorization: Bearer $access_token" \
  -d '{
    "members" : [
      {"email" : "<your_new_member_email>", "type" : "user"},
      {"email" : "<existing_member>", "type" : "user"}
    ]
  }' | jq "."
```

Example: Add Users to Your Automation Assembler Project

For the project **`Example-Assembler-project`**, add another administrator and member at **`mycompany.com`**.

```
$ url='https://appliance.domain.com'
$ api_version='2021-07-15'
$ project_id='5944aacb-91de-4541-bb9e-ef2a5403f81b'
```

List the details of your project.

```
$ curl -X GET -H 'Accept: application/json' -H "Authorization: Bearer $access_token" "$url/
iaas/api/projects/$project_id?apiVersion=$api_version" | jq "."
```

A snippet of the response shows existing administrators, members, and viewers.

```
...
"administrators": [
    {
      "email": "admin1@mycompany.com",
      "type": "user"
    }
  ],
  "members": [
    {
      "email": "member1@mycompany.com",
      "type": "user"
    }
  ],
  "viewers": [
    {
      "email": "viewer1@mycompany.com",
      "type": "user"
    }
  ],
...
```

Add the administrator. Include the existing administrator **admin1@mycompany.com** in the PATCH request.

```
$ curl -X PATCH \
  "$url/iaas/api/projects/$project_id?apiVersion=$api_version" \
  -H 'Content-Type: application/json' \
  -H "Authorization: Bearer $access_token" \
  -d '{
    "administrators" : [
      {"email" : "newadministrator@mycompany.com", "type" : "user"},
      {"email" : "admin1@mycompany.com", "type" : "user"}
    ]
  }' | jq "."
```

Add the member. Include the existing member **member1@mycompany.com** in the PATCH request.

```
$ curl -X PATCH \
  "$url/iaas/api/projects/$project_id?apiVersion=$api_version" \
  -H 'Content-Type: application/json' \
  -H "Authorization: Bearer $access_token" \
  -d '{
    "members" : [
      {"email" : "newmember@mycompany.com", "type" : "user"},
      {"email" : "member1@mycompany.com", "type" : "user"}
    ]
  }' | jq "."
```

The response shows the project with administrators, members, and viewers, including the new administrator and new member.

```
{
  "administrators": [
    {
      "email": "newadministrator@mycompany.com",
      "type": "user"
    },
    {
      "email": "admin1@mycompany.com",
      "type": "user"
    }
  ],
  "members": [
    {
      "email": "newmember@mycompany.com",
      "type": "user"
    },
    {
      "email": "member1@mycompany.com",
      "type": "user"
    }
  ],
  "viewers": [
    {
      "email": "viewer1@mycompany.com",
```

```
      "type": "user"
    }
  ],
  "sharedResources": true,
  "name": "Example-Assembler-project",
  "description": "This is an example project for Automation Assembler",
  "id": "5944aacb-91de-4541-bb9e-ef2a5403f81b",
  "orgId": "8327d53f-91ea-420a-8613-ba8f3149db95"
  }
}
```

Add a Cloud Zone to Your Project

As a Automation Assembler administrator, you use the `PATCH iaas/api/projects` request to attach a cloud zone to a project.

If the project already has cloud zones attached, review them to ensure that all zones needed for the project are included in the PATCH request to add a new cloud zone.

Prerequisites

- Verify that all general prerequisites and prerequisites for the Automation Assembler Infrastructure as a Service (IaaS) service have been satisfied. See Chapter 3 Prerequisites for API Use Case Examples.
- Verify that you have added a project and you have the project ID. See Create a Project to use in Automation Assembler.

Procedure

1 Assign the project ID variable.

```
project_id='<your_project_id>'
```

your_project_id is the ID of the new project you created.

2 List all cloud zones.

```
curl -X GET -H 'Content-Type: application/json' -H "Authorization: Bearer $access_token"
"$url/iaas/api/zones?apiVersion=$api_version"  | jq "."
```

3 To obtain a cloud zone ID, examine the response and find the ID of the zone that you want to attach to your project. For a snippet of the response, see Create a Cloud Zone.

4 Assign the cloud zone variable.

```
zone_id='<your_zone_id>'
```

5 List the details of your project.

```
curl -X GET -H 'Accept: application/json' -H "Authorization: Bearer $access_token" "$url/
iaas/api/projects/$project_id?apiVersion=$api_version" | jq "."
```

6 Examine the response to see the zones already attached to your project.

7 Submit a request to attach and configure a new cloud zone that includes the ID of existing cloud zones for the project.

Note If the call does not include existing cloud zones that are already attached to the project, the PATCH request removes those cloud zones from the project.

```
curl -X PATCH \
  "$url/iaas/api/projects/$project_id?apiVersion=$api_version" \
  -H 'Content-Type: application/json' \
  -H "Authorization: Bearer $access_token" \
  -d '{
    "zoneAssignmentConfigurations" : [
      {
        "zoneId" : "'$zone_id'",
        "priority": 1,
        "maxNumberInstances": 50
      },
      {
        "zoneId" : "<existing_cloud_zone_id>",
        "priority": 2,
        "maxNumberInstances": 100
      }
    ]
  }' | jq "."
```

Example: Attach a Cloud Zone to Your Project

For the new project **`Example-project`**, add a cloud zone.

```
$ url='https://appliance.domain.com'
$ api_version='2021-07-15'
$ project_id='5944aacb-91de-4541-bb9e-ef2a5403f81b'
```

List all cloud zones.

```
$ curl -X GET -H 'Content-Type: application/json' -H "Authorization: Bearer $access_token"
"$url/iaas/api/zones?apiVersion=$api_version"  | jq "."
```

Examine the response to find the cloud zone you want and assign the zone ID variable.

```
$ zone_id='4965d34c3bfe0275574bc5fd8782a'
```

List the details of your project.

```
$ curl -X GET -H 'Accept: application/json' -H "Authorization: Bearer $access_token" "$url/
iaas/api/projects/$project_id?apiVersion=$api_version" | jq "."
```

A snippet of the response shows an existing cloud zone.

```
...
"zones": [
    {
      "zoneId": "3cc2ecb989eee87557b0d532d4bb0",
      "priority": 0,
      "maxNumberInstances": 0
    }
...
```

Add the new cloud zone. Include the existing cloud zone **3cc2ecb989eee87557b0d532d4bb0** in the PATCH request.

```
$ curl -X PATCH \
  "$url/iaas/api/projects/$project_id?apiVersion=$api_version" \
  -H 'Content-Type: application/json' \
  -H "Authorization: Bearer $access_token" \
  -d '{
    "zoneAssignmentConfigurations" : [
      {
        "zoneId" : "'$zone_id'",
        "priority": 1,
        "maxNumberInstances": 50
      },
      {
        "zoneId" : "3cc2ecb989eee87557b0d532d4bb0",
        "priority": 2,
        "maxNumberInstances": 100
      }
    ]
  }' | jq "."
```

The response after adding a cloud zone lists the project with its administrators, members, and zone.

```
{
  "administrators": [
    {
      "email": "admin1@mycompany.com"
    }
  ],
  "members": [
    {
      "email": "user1@mycompany.com"
    }
  ],
  "zones": [
    {
      "zoneId": "4965d34c3bfe0275574bc5fd8782a",
      "priority": 1,
      "maxNumberInstances": 50
    },
    {
```

```
      "zoneId": "3cc2ecb989eee87557b0d532d4bb0",
      "priority": 2,
      "maxNumberInstances": 100
    }
  ],
  "sharedResources": true,
  "name": "Example-project",
  "description": "This is an example project",
  "id": "5944aacb-91de-4541-bb9e-ef2a5403f81b",
  "organizationId": "8327d53f-91ea-420a-8613-ba8f3149db95",
  "orgId": "8327d53f-91ea-420a-8613-ba8f3149db95",
  "_links": {
    "self": {
      "href": "/iaas/api/projects/edfd6f26-5d82-428f-96b0-b10ac5e4aca9"
    }
  }
}
```

Create Flavor Mappings

To create a flavor mapping, you make a POST request with a region ID associated with a cloud account. The cloud account can be an AWS, vSphere, Azure, or GCP cloud account.

Cloud vendors use flavors, or instance types, to express standard deployment sizings such as small (1 CPU, 2 GB RAM) or large (2 CPU, 8 GB RAM) for compute resources. When you build a cloud template, you pick a flavor that fits your needs and map a flavor name to a value for each account or region.

The same API calls create a flavor profile for an AWS, vSphere, Azure, or GCP cloud account. However, the flavor mapping used to create the flavor profile varies for each type of cloud account. This procedure provides the steps to create a flavor profile for an AWS cloud account. Additional examples show how to create flavor profiles for vSphere and Azure cloud accounts.

Prerequisites

- Verify that all general prerequisites and prerequisites for the Automation Assembler Infrastructure as a Service (IaaS) service have been satisfied. See Chapter 3 Prerequisites for API Use Case Examples.
- Verify that you have the cloud account ID for the new cloud account that you added. See Adding Cloud Accounts.

Procedure

1 Assign the cloud account ID variable.

```
cloud_account_id='<your_cloud_account_id>'
```

2 Look up region IDs associated with the cloud account and in the external region ID **us-east-1**.

```
curl -X GET -H 'Content-Type: application/json' -H "Authorization: Bearer $access_token"
"$url/iaas/api/regions/?apiVersion=$api_version&"'$filter'"=externalRegionId%20eq%20'us-
east-1'%20and%20cloudAccountId%20eq%20'"$cloud_account_id"'"  | jq "."
```

3 Examine the response to find the ID for the region that you want.

4 Assign the region ID variable.

```
region_id='<your_region_id>'
```

5 List all fabric flavors.

```
curl -X GET -H 'Content-Type: application/json' -H "Authorization: Bearer $access_token"
"$url/iaas/api/fabric-flavors/?apiVersion=$api_version" | jq "."
```

6 To select fabric flavor names with resources that fit your needs, examine the response.

7 Create a flavor profile for an AWS account that uses the fabric flavor names in your flavor mapping.

```
curl -X POST \
  $url/iaas/api/flavor-profiles?apiVersion=$api_version \
  -H 'Content-Type: application/json' \
  -H "Authorization: Bearer $access_token" \
  -d '{
    "name":"<your_flavor_profile>",
    "description":"Example AWS Compute flavors",
    "flavorMapping": {
      "small": {
        "name":"<flavor_name1_from_response>"
      },
      "medium": {
        "name":"<flavor_name2_from_response>"
      },
      "large":{
        "name":"<flavor_name3_from_response>"
      }
    },
    "regionId":"'$region_id'"
  }' | jq "."
```

8 To obtain the flavor profile ID, examine the response.

9 Assign the flavor profile ID variable.

```
flavor_profile_id='<your_flavor_profile_id>'
```

10 (Optional) Look up the flavor profile you created with your flavor profile ID.

```
curl -X GET -H 'Content-Type: application/json' -H "Authorization: Bearer $access_token"
$url/iaas/api/flavor-profiles/$flavor_profile_id?apiVersion=$api_version | jq "."
```

The response shows the name and ID for the flavor profile you created.

Note Using the external region ID and the cloud account ID, you can also filter for the flavor profile with a query that does not require the flavor profile ID. See Filtering Resources by Region ID.

Example: Create flavor mappings for different cloud accounts

Create a flavor mapping for an AWS cloud account.

1 Assign the required variables including the cloud account ID for an AWS cloud account.

```
$ url='https://appliance.domain.com'
$ api_version='2021-07-15'
$ cloud_account_id='c8c3c9bfdb449475-7f703c5265a63d87-5fa34c478df36b060e1ca3551254c403301
3bf3283908e4661cd1c6fb2f8b9ae-ce5aad01092b47558644f6b6615d'
```

2 Look up region IDs associated with the cloud account and in the external region ID **`us-east-1`**.

```
$ curl -X GET -H 'Content-Type: application/json' -H "Authorization: Bearer $access_token"
"$url/iaas/api/regions/?apiVersion=$api_version&"'$filter'"=externalRegionId%20eq%20'us-
east-1'%20and%20cloudAccountId%20eq%20'"$cloud_account_id"'"  | jq "."
```

A snippet of the response shows the region ID.

```
...
      "externalRegionId": "us-east-1",
      "cloudAccountId":
"c8c3c9bfdb449475-7f703c5265a63d87-5fa34c478df36b060e1ca3551254c4033013bf3283908e4661cd1c6f
b2f8b9ae-ce5aad01092b47558644f6b6615d",
      "id": "37d6c1acf4a8275586468873c739",
      "updatedAt": "2022-04-02",
...
```

3 Assign the AWS region ID.

```
$ aws_region_id='37d6c1acf4a8275586468873c739'
```

4 List all fabric flavors.

```
$ curl -X GET -H 'Content-Type: application/json' -H "Authorization: Bearer $access_token"
"$url/iaas/api/fabric-flavors/?apiVersion=$api_version" | jq "."
```

A snippet of the response shows a fabric flavor name with its resource size.

```
...
    {
      "id": "t2.micro",
      "name": "t2.micro",
      "cpuCount": 1,
      "memoryInMB": 1024,
      "storageType": "EBS",
      "networkType": "Low to Moderate"
    },
...
```

5 Select fabric flavor names with resources that fit your needs and create an AWS flavor profile named **`aws-flavor-profile`**.

```
$ curl -X POST \
  $url/iaas/api/flavor-profiles?apiVersion=$api_version \
  -H 'Content-Type: application/json' \
  -H "Authorization: Bearer $access_token" \
  -d '{
    "name":"aws-flavor-profile",
    "description":"Example AWS Compute flavors",
    "flavorMapping": {
      "small": {
        "name":"t2.micro"
      },
      "medium": {
        "name":"t2.medium"
      },
      "large":{
        "name":"t2.large"
      }
    },
    "regionId":"'$aws_region_id'"
  }' | jq "."
```

A snippet of the response shows the flavor profile ID.

```
...
  "externalRegionId": "us-east-1",
  "name": "aws-flavor-profile",
  "description": "Example AWS Compute flavors",
  "id": "835249077934b47558eca5963e068",
  "updatedAt": "2022-04-02",
...
```

Create a flavor mapping for a vSphere cloud account.

1 Assign the required variables including the cloud account ID for a vSphere cloud account.

```
$ url='https://appliance.domain.com'
$ api_version='2021-07-15'
$ cloud_account_id='c8c3c9bfdb449475-7f703c5265a63d87-5fa34c478df36b060e1ca3551254c403301
3bf3283908e4661cd1c6fb2f8b9ae-ce5aad01092b47558644f6b6615d'
```

2 Look up region IDs associated with the cloud account and in the external region ID **`Datacenter:datacenter-2`**.

```
$ curl -X GET -H 'Content-Type: application/json' -H "Authorization: Bearer $access_token"
"$url/iaas/api/regions/?
apiVersion=$api_version&"'$filter'"=externalRegionId%20eq%20'Datacenter:datacenter-2'%20and
%20cloudAccountId%20eq%20'"$cloud_account_id"'"  | jq "."
```

A snippet of the response shows the region ID.

```
...
      "externalRegionId": "Datacenter:datacenter-2",
      "cloudAccountId": "c8c3c9bfdb449475-7f703c5265a63d87-
d06bf79904ce5096492a2a2fc557fb0457d7d3c5b5e7ae20b29957788812bb3d-
d5a5e16bdc3eec7557245925e1b08",
      "id": "2aaf79b789eee8755724592b06d39",
      "updatedAt": "2022-04-02",
...
```

3 Assign the vSphere region ID.

```
$ vsphere_region_id='2aaf79b789eee8755724592b06d39'
```

4 Create a vSphere flavor profile named **`vcenter-flavor-profile`**.

```
$ curl -X POST \
  $url/iaas/api/flavor-profiles?apiVersion=$api_version \
  -H 'Content-Type: application/json' \
  -H "Authorization: Bearer $access_token" \
  -d '{
    "name":"vcenter-flavor-profile",
    "description":"Example vSphere Compute flavors",
    "flavorMapping": {
      "small": {
        "cpuCount":1,
        "memoryInMB": 1024
      },
      "medium": {
        "cpuCount":2,
        "memoryInMB": 2048
      },
      "large":{
        "cpuCount":4,
        "memoryInMB": 4096
```

```
      }
    },
    "regionId":"'$vsphere_region_id'"
  }' | jq "."
```

A snippet of the response shows the flavor profile ID.

```
...
  "externalRegionId": "Datacenter:datacenter-2",
  "name": "vcenter-flavor-profile",
  "description": "Example vSphere Compute flavors",
  "id": "cfb7246505319275572e9e68372d0",
  "updatedAt": "2022-04-02",
...
```

Create a flavor mapping with an Azure cloud account ID.

1 Assign the required variables including the cloud account ID for an Azure cloud account.

```
$ url='https://appliance.domain.com'
$ api_version='2021-07-15'
$ cloud_account_id='c8c3c9bfdb449475-7f703c5265a63d87-5fa34c478df36b060e1ca3551254c403301
3bf3283908e4661cd1c6fb2f8b9ae-ce5aad01092b47558644f6b6615d'
```

2 Look up region IDs associated with the cloud account and in the external region ID **`us-east-1`**.

```
$ curl -X GET -H 'Content-Type: application/json' -H "Authorization: Bearer $access_token"
"$url/iaas/api/regions/?apiVersion=$api_version&"'$filter'"=externalRegionId%20eq%20'us-
east-1'%20and%20cloudAccountId%20eq%20'"$cloud_account_id"'"  | jq "."
```

A snippet of the response shows the region ID.

```
...
      "externalRegionId": "us-east-1",
      "cloudAccountId":
"c8c3c9bfdb449475-7f703c5265a63d87-5fa34c478df36b060e1ca3551254c4033013bf3283908e4661cd1c6f
b2f8b9ae-ce5aad01092b47558644f6b6615d",
      "id": "37d6c1acf4a8275586468873c739",
      "updatedAt": "2022-04-02",
...
```

3 Assign the Azure region ID.

```
$ azure_region_id='37d6c1acf4a8275586468873c739'
```

4 List all fabric flavors.

```
$ curl -X GET -H 'Content-Type: application/json' -H "Authorization: Bearer $access_token"
"$url/iaas/api/fabric-flavors/?apiVersion=$api_version" | jq "."
```

A snippet of the response shows a fabric flavor name with its resource size.

```
...
    {
      "id": "Standard_A0",
      "name": "Standard_A0",
      "cpuCount": 1,
      "memoryInMB": 768,
      "bootDiskSizeInMB": 1047552,
      "dataDiskSizeInMB": 20480,
      "dataDiskMaxCount": 1
    },
...
```

5 Select fabric flavor names with resources that fit your needs and create an Azure flavor profile named **`azure-flavor-profile`**.

```
$ curl -X POST \
  $url/iaas/api/flavor-profiles?apiVersion=$api_version \
  -H 'Content-Type: application/json' \
  -H "Authorization: Bearer $access_token" \
  -d '{
    "name":"azure-flavor-profile",
    "description":"Example Azure Compute flavors",
    "flavorMapping": {
      "small": {
        "name":"Standard_A0"
      },
      "medium": {
        "name":"Standard_A1"
      },
      "large":{
        "name":"Standard_A2"
      }
    },
    "regionId":"'$azure_region_id'"
  }' | jq "."
```

A snippet of the response shows the flavor profile ID.

```
...
  "externalRegionId": "us-east-1",
  "name": "azure-flavor-profile",
  "description": "Example Azure Compute flavors",
  "id": "4965d34c3bfe0275574bc6e505b78",
  "updatedAt": "2022-04-02",
...
```

Create Image Mappings

To create an image mapping, you make a POST request with a region ID associated with a cloud account. The cloud account can be an AWS, vSphere, Azure, or GCP cloud account.

Cloud vendors use images to configure a VM based on OS settings, such as an ubuntu-16 configuration. When you build a cloud template, you pick an image that fits your needs and map an image name to a value for each account or region. You can also add constraints and configuration scripts to further control resource placement.

The same API calls create an image profile for an AWS, vSphere, Azure, or GCP cloud account. The following example shows how to create an image profile for a vSphere cloud account with the external region ID **`Datacenter:datacenter-3`**.

Prerequisites

- Verify that all general prerequisites and prerequisites for the Automation Assembler Infrastructure as a Service (IaaS) service have been satisfied. See Chapter 3 Prerequisites for API Use Case Examples.
- Verify that you have the cloud account ID for the new cloud account that you added. See Adding Cloud Accounts.

Procedure

1 Assign the cloud account ID variable.

```
cloud_account_id='<your_cloud_account_id>'
```

2 Look up region IDs associated with the cloud account and in the external region ID.

```
curl -X GET -H 'Content-Type: application/json' -H "Authorization: Bearer $access_token"
"$url/iaas/api/regions/?
apiVersion=$api_version&"'$filter'"=externalRegionId%20eq%20'Datacenter:datacenter-3'%20and
%20cloudAccountId%20eq%20'"$cloud_account_id"'"  | jq "."
```

3 Examine the response to find the ID for the region that you want.

4 Assign the region ID variable.

```
region_id='<your_region_id>'
```

5 Create an image profile with an image mapping that specifies the OVA/OVF links.

```
curl -X POST \
  $url/iaas/api/image-profiles?apiVersion=$api_version \
  -H 'Content-Type: application/json' \
  -H "Authorization: Bearer $access_token" \
  -d '{
    "name":"<your_image_profile>",
    "description":"<your_image_profile_description>",
    "imageMapping": {
      "ubuntu": {
         "externalId": "https://cloud-images.ubuntu.com/releases/16.04/release-20220305/
ubuntu-16.04-server-cloudimg-amd64.ova"
```

```
      }
    },
    "regionId":"'$region_id'"
  }' | jq "."
```

6 To obtain the image profile ID, examine the response.

Example: Create image mapping

Assign the required variables including a cloud account ID.

```
$ url='https://appliance.domain.com''
$ api_version='2021-07-15'
$ cloud_account_id='ff4e7585-4197-41fb-89cb-179ef4d24779'
```

Look up region IDs associated with the cloud account and in the external region ID **`Datacenter:datacenter-3`**.

```
$ curl -X GET -H 'Content-Type: application/json' -H "Authorization: Bearer $access_token"
"$url/iaas/api/regions/?
apiVersion=$api_version&"'$filter'"=externalRegionId%20eq%20'Datacenter:datacenter-3'%20and%20
cloudAccountId%20eq%20'"$cloud_account_id"'"  | jq "."
```

A snippet of the response shows the region ID.

```
...
          "externalRegionId": "Datacenter:datacenter-3",
          "name": "w01-vc08-DC",
          "cloudAccountId": "ff4e7585-4197-41fb-89cb-179ef4d24779",
          "id": "9b148b38-fc7c-4560-b413-5f47b30e57d8",
...
```

Assign the region ID.

```
$ region_id='9b148b38-fc7c-4560-b413-5f47b30e57d8'
```

Create an image profile named **`example-image-profile`**.

```
$ curl -X POST \
  $url/iaas/api/image-profiles?apiVersion=$api_version \
  -H 'Content-Type: application/json' \
  -H "Authorization: Bearer $access_token" \
  -d '{
    "name":"example-image-profile",
    "description":"Example image profile",
    "imageMapping": {
      "ubuntu": {
         "externalId": "https://cloud-images.ubuntu.com/releases/16.04/release-20220305/
ubuntu-16.04-server-cloudimg-amd64.ova"
      }
    },
    "regionId":"'$region_id'"
  }' | jq "."
```

A snippet of the response shows the image profile ID.

```
...
      "externalRegionId": "Datacenter:datacenter-3",
      "cloudAccountId": "ff4e7585-4197-41fb-89cb-179ef4d24779",
      "name": "example-image-profile",
      "id": "f670fdfc-66d6-4689-9793-d524e7066d1e-9b148b38-fc7c-4560-b413-5f47b30e57d8",
...
```

Working with Networks

As a Automation Assembler administrator, you can use the IaaS APIs to create network profiles and get information about data collected and networks.

Create Network Profiles

To create a network profile, you make a POST request with a region ID associated with a cloud account.

A Automation Assembler network profile describes the behavior of the network to be deployed. For example, a network might need to be Internet facing versus internal only. Networks and their profiles are cloud-specific.

For information on network profiles, see Learn more about network profiles in VMware Aria Automation.

The networks in this example are used for provisioning to existing or public networks. If you are working with on-demand or deployment networks, see Using the Network APIs.

If you are provisioning to a private network, or outbound networks with one-way access to upstream networks, you create a network profile with isolation enabled by either subnet or security group. See Create a Network Profile with Isolation.

If you want to add firewall rules to all machines provisioned with a network profile, you create a network profile with security groups. See Create a Network Profile with Security Groups.

Prerequisites

- Verify that all general prerequisites and prerequisites for the Automation Assembler Infrastructure as a Service (IaaS) service have been satisfied. See Chapter 3 Prerequisites for API Use Case Examples.
- Verify that you have the cloud account ID for the new cloud account that you added. See Adding Cloud Accounts.

Procedure

1 Assign the cloud account ID variable.

```
cloud_account_id='<your_cloud_account_id>'
```

2 Look up regions associated with the cloud account ID and with the region name **us-east-1**.

```
curl -X GET -H 'Content-Type: application/json' -H "Authorization: Bearer
$access_token" "$url/iaas/api/regions/?apiVersion=$api_version&"'$filter'"=name%20eq%20'us-
east-1'%20and%20cloudAccountId%20eq%20'"$cloud_account_id"'"  | jq "."
```

3 Examine the response to find the ID for the region that you want.

4 Assign the region ID variable.

```
region_id='<your_region_id>'
```

5 Filter for fabric networks associated with the cloud account ID and in the external region ID **us-east-1**.

```
curl -X GET -H 'Content-Type: application/
json' -H "Authorization: Bearer $access_token" $url/
iaas/api/fabric-networks?apiVersion=$api_version&"'$filter='"externalRegionId%20eq%20'us-
east-1'%20and%20cloudAccountId%20eq%20'$cloud_account_id'" | jq "."
```

For details on how to construct a filter, see Filtering Resources by Region ID.

6 Examine the response to find the IDs for the public networks that you want to include in your network profile.

7 Create a network profile.

```
curl -X POST \
  $url/iaas/api/network-profiles?apiVersion=$api_version \
  -H 'Content-Type: application/json' \
  -H "Authorization: Bearer $access_token" \
  -d '{
    "name":"<your-network-profile>",
    "description":"Example Network Profile",
    "regionId":"'$region_id'",
    "fabricNetworkIds": [
      "<network_id1_from_response>",
      "<network_id2_from_response>"
    ],
    "tags": [ { "key": "env", "value": "prod" } ]
  }' | jq "."
```

8 To obtain the network profile ID, examine the response.

9 Assign the network profile ID variable.

```
network_profile_id='<your_network_profile_id>'
```

10 (Optional) Look up the network profile you created with your network profile ID.

```
curl -X GET -H 'Content-Type: application/json' -H "Authorization: Bearer $access_token"
$url/iaas/api/network-profiles/$network_profile_id?apiVersion=$api_version | jq "."
```

The response shows the name and ID for the network profile you created.

Example: Create a network profile

Assign the required variables including a cloud account ID.

```
url='https://appliance.domain.com'
$ api_version='2021-07-15'
$ cloud_account_id='c8c3c9bfdb449475-7f703c5265a63d87-5fa34c478df36b060e1ca3551254c403301
3bf3283908e4661cd1c6fb2f8b9ae-ce5aad01092b47558644f6b6615d'
```

Look up region IDs associated with the cloud account and in the external region ID **us-east-1**.

```
$ curl -X GET -H 'Content-Type: application/json' -H "Authorization: Bearer $access_token"
"$url/iaas/api/regions/?apiVersion=$api_version&"'$filter'"=externalRegionId%20eq%20'us-
east-1'%20and%20cloudAccountId%20eq%20'"$cloud_account_id"'"  | jq "."
```

A snippet of the response shows the region ID.

```
...
      "externalRegionId": "us-east-1",
      "cloudAccountId":
"c8c3c9bfdb449475-7f703c5265a63d87-5fa34c478df36b060e1ca3551254c4033013bf3283908e4661cd1c6fb2f
8b9ae-ce5aad01092b47558644f6b6615d",
      "id": "37d6c1acf4a8275586468873c739",
      "updatedAt": "2022-04-02",
...
```

Assign the region ID.

```
$ region_id='37d6c1acf4a8275586468873c739'
```

Filter for fabric networks associated with the cloud account ID and in the external region ID **us-east-1**.

```
curl -X GET -H 'Content-Type: application/json' -H "Authorization: Bearer $access_token" $url/
iaas/api/fabric-networks?apiVersion=$api_version | jq "."
```

A snippet of the response shows the fabric network ID for a public network that you can include in your network profile.

```
...
      "isPublic": true,
      "isDefault": true,
      "cidr": "172.31.16.0/20",
      "externalRegionId": "us-east-1",
      "tags": [
        {
          "key": "vmware.enumeration.type",
          "value": "ec2_subnet"
        }
      ],
      "cloudAccountIds": [
        "c8c3c9bfdb449475-7f703c5265a63d87-
f8e705d89b2569e1aac66c6d00bf4fc7ef4b1c44100f0e944af31eb8ba3d2a5a-5a45a4b9d5c72475575931611aa28
```

```
",

"c8c3c9bfdb449475-7f703c5265a63d87-5fa34c478df36b060e1ca3551254c4033013bf3283908e4661cd1c6fb2f
8b9ae-ce5aad01092b47558644f6b6615d"
      ],
      "name": "subnet-0130834a",
      "id": "d43efed364ef18755759316540e3f",
...
```

Select the IDs of fabric networks that you want to include in your profile and create a network profile named **example-network-profile**.

```
$ curl -X POST \
  $url/iaas/api/network-profiles?apiVersion=$api_version \
  -H 'Content-Type: application/json' \
  -H "Authorization: Bearer $access_token" \
  -d '{
    "name":"example-network-profile",
    "description":"Example Network Profile",
    "regionId":"'$region_id'",
    "fabricNetworkIds": [
      "d43efed364ef18755759316540e3d",
      "d43efed364ef18755759316540e3f"
    ],
    "tags": [ { "key": "env", "value": "prod" } ]
  }' | jq "."
```

A snippet of the response shows the network profile ID.

```
...
  "name": "example-network-profile",
  "description": "Example Network Profile",
  "id": "9cb2d111c768927558f043ec13d70",
  "updatedAt": "2022-04-02",
...
```

Create a Network Profile with Isolation

To create either private networks without access to outside networks or outbound networks with one-way access to upstream networks, you create a network profile with isolation and specify the isolation type.

This procedure provides the steps to create a network that supports isolation using a subnet, and includes optional steps that show how to create the network using an external subnet, security groups, or a VLAN transport zone. The network profile that uses a VLAN transport zone only supports private networks.

Prerequisites

- Verify that all general prerequisites and prerequisites for the Automation Assembler Infrastructure as a Service (IaaS) service have been satisfied. See Chapter 3 Prerequisites for API Use Case Examples.

- Verify that you have the cloud account ID for the new cloud account that you added. See Adding Cloud Accounts.
- Verify that you have the region ID for the regions you want to include in the profile. See the procedure in Create Network Profiles.
- For simplicity, examples use **`us-east-1`** as the external region ID.
- Verify that you have the IDs for the non-public fabric networks you want to include in the profile. See the procedure in Create Network Profiles.

Procedure

1 Assign the cloud account ID variable.

```
cloud_account_id='<your_cloud_account_id>'
```

2 Assign the region ID variable.

```
region_id='<your_region_id>'
```

3 Filter for network domains associated with the cloud account ID and in the external region ID **`us-east-1`**.

```
curl -X GET -H 'Content-Type: application/
json' -H "Authorization: Bearer $access_token" "$url/
iaas/api/network-domains?apiVersion=$api_version&"'$filter='"externalRegionId%20eq%20'us-
east-1'%20and%20cloudAccountId%20eq%20'$cloud_account_id'" | jq "."
```

4 Examine the response to find the IDs for the network domain that you want to include in your network profile. If you are creating a network profile with a VLAN transport zone, select a network domain that has the custom property `"__transportZoneTrafficType":"VLAN_BACKED"`.

5 Create a network profile that supports isolation using a subnet and IDs for a non-public network.

```
curl -X POST \
  $url/iaas/api/network-profiles?apiVersion=$api_version \
  -H 'Content-Type: application/json' \
  -H "Authorization: Bearer $access_token" \
  -d '{
    "name":"<your-network-profile-with-isolation-by-subnet>",
    "description":"Example Network Profile",
    "regionId":"'$region_id'",
    "isolationType" : "SUBNET",
    "isolationNetworkDomainId" : "<network_domain_id_from_response>",
    "isolatedNetworkCIDRPrefix" : "27",
    "fabricNetworkIds": [
      "<non_public_network_id1>",
```

```
      "<non_public_network_id2>"
    ],
    "tags": [ { "key": "env", "value": "prod" } ]
  }' | jq "."
```

The response shows the name and ID for the network profile you created.

6 (Optional) Create a network profile that supports isolation using an external subnet.

```
curl -X POST \
  $url/iaas/api/network-profiles?apiVersion=$api_version \
  -H 'Content-Type: application/json' \
  -H "Authorization: Bearer $access_token" \
  -d '{
    "name":"<your-network-profile-with-isolation-by-external-subnet>",
    "description":"Example Network Profile",
    "regionId":"'$region_id'",
    "isolationType" : "SUBNET",
    "isolationNetworkDomainId" : "<network_domain_id_from_response>",
    "isolatedNetworkCIDRPrefix" : "27",
    "isolationExternalFabricNetworkId": "<non_public_network_id1>",
    "fabricNetworkIds": [
      "<non_public_network_id1>",
      "<non_public_network_id2>"
    ],
    "tags": [ { "key": "env", "value": "prod" } ]
  }' | jq "."
```

The response shows the name and ID for the network profile you created.

7 (Optional) Create a network profile that supports isolation using security groups.

```
curl -X POST \
  $url/iaas/api/network-profiles?apiVersion=$api_version \
  -H 'Content-Type: application/json' \
  -H "Authorization: Bearer $access_token" \
  -d '{
    "name":"<your-network-profile-with-isolation-by-security-group>",
    "description":"Example Network Profile",
    "regionId":"'$region_id'",
    "isolationType" : "SECURITY_GROUP",
    "fabricNetworkIds": [
      "<non_public_network_id1>",
      "<non_public_network_id2>"
    ],
    "tags": [ { "key": "env", "value": "prod" } ]
  }' | jq "."
```

The response shows the name and ID for the network profile you created.

Example: Create various types of network profiles with isolation

The following examples include the requests used to create a network profiles that support isolation using:

- A subnet.
- An external subnet.
- Security groups.
- A VLAN transport zone.

Assign the required variables including a cloud account ID and a region ID.

```
$ url='https://appliance.domain.com'
$ api_version='2021-07-15'
$ cloud_account_id='c8c3c9bfdb449475-7f703c5265a63d87-5fa34c478df36b060e1ca3551254c403301
3bf3283908e4661cd1c6fb2f8b9ae-ce5aad01092b47558644f6b6615d'
$ region_id='37d6c1acf4a8275586468873c739'
```

Filter for network domains associated with the cloud account ID and in the external region ID **`us-east-1`**.

```
curl -X GET -H 'Content-Type: application/json' -H "Authorization: Bearer $access_token"
"$url/iaas/api/network-domains?
apiVersion=$api_version&"'$filter='"externalRegionId%20eq%20'us-
east-1'%20and%20cloudAccountId%20eq%20'$cloud_account_id'" | jq "."
```

A snippet of the response shows the ID for a network domain that you can include in your network profile.

```
...
      "externalId": "vpc-4511a53d",
      "name": "rainpole-dev",
      "id": "233df662ec3b4875575931653ef00",
      "createdAt": "2022-04-02",
      "updatedAt": "2022-04-02",
      "organizationId": "8327d53f-91ea-420a-8613-ba8f3149db95",
      "orgId": "8327d53f-91ea-420a-8613-ba8f3149db95",
      "_links": {
        "cloud-accounts": {
          "hrefs": [
            "/iaas/api/cloud-accounts/c8c3c9bfdb449475-7f703c5265a63d87-
f8e705d89b2569e1aac66c6d00bf4fc7ef4b1c44100f0e944af31eb8ba3d2a5a-5a45a4b9d5c72475575931611aa28
",
            "/iaas/api/cloud-accounts/
c8c3c9bfdb449475-7f703c5265a63d87-5fa34c478df36b060e1ca3551254c4033013bf3283908e4661cd1c6fb2f8
b9ae-ce5aad01092b47558644f6b6615d"
          ]
        },
        "self": {
          "href": "/iaas/api/network-domains/233df662ec3b4875575931653ef00"
...
```

To create a network profile with a VLAN transport zone, ensure that the network domain you choose includes a custom property for the transport zone.

```
...
      "customProperties": {
        "__path": "/infra/sites/default/enforcement-points/default/transport-zones/
9a358e99-5734-4926-b718-37cf4862f4bf",
        "__host_identifier": "[\"host-16\",\"host-23\",\"host-21\"]",
        "__cluster_identifier": "[\"domain-c8\"]",
        "__transportZoneTrafficType": "VLAN_BACKED",
        "path": "/infra/sites/default/enforcement-points/default/transport-zones/
9a358e99-5734-4926-b718-37cf4862f4bf"
            }
...
```

With the IDs of fabric networks that you want to include in your profile and the network domain ID you want to include, create a network profile named **example-network-profile-with-isolation-by-subnet**.

```
$ curl -X POST  \
  $url/iaas/api/network-profiles?apiVersion=$api_version  \
  -H 'Content-Type: application/json'  \
  -H "Authorization: Bearer $access_token"  \
  -d '{
    "name":"example-network-profile-with-isolation-by-subnet",
    "description":"Example Network Profile",
    "regionId":"'$region_id'",
    "isolationType" : "SUBNET",
    "isolationNetworkDomainId" : "233df662ec3b4875575931653ef00",
    "isolatedNetworkCIDRPrefix" : "27",
    "fabricNetworkIds": [
      "c19bd2921af95075575931654066a",
      "8fe650cc09d0627558d55c9ba1793"
    ],
    "tags": [ { "key": "env", "value": "prod" } ]
}' | jq "."
```

A snippet of the response shows the network profile ID.

```
...
  "name": "example-network-profile-with-isolation-by-subnet",
  "description": "Example Network Profile",
  "id": "2065036880e1c47558f1693558870",
  "updatedAt": "2022-04-02",
...
```

Provide the **isolationExternalFabricNetworkId** to create a network profile with isolation using an external subnet.

```
$ curl -X POST  \
  $url/iaas/api/network-profiles?apiVersion=$api_version  \
  -H 'Content-Type: application/json'  \
  -H "Authorization: Bearer $access_token"  \
```

```
  -d '{
    "name":"example-network-profile-with-isolation-by-external-subnet",
    "description":"Example Network Profile",
    "regionId":"'$region_id'",
    "isolationType" : "SUBNET",
    "isolationNetworkDomainId" : "233df662ec3b4875575931653ef00",
    "isolatedNetworkCIDRPrefix" : "27",
    "isolationExternalFabricNetworkId": "c19bd2921af95075575931654066a",
    "fabricNetworkIds": [
      "c19bd2921af95075575931654066a",
      "8fe650cc09d0627558d55c9ba1793"
    ],
    "tags": [ { "key": "env", "value": "prod" } ]
}' | jq "."
```

A snippet of the response shows the network profile ID.

```
...
  "name": "example-network-profile-with-isolation-by-external-subnet",
  "description": "Example Network Profile",
  "id": "2065036880e1c47558f16bd085288",
  "updatedAt": "2022-04-02",
...
```

Use the **"isolationType": "SECURITY_GROUP"** to create a network profile with isolation using a security group. Because this isolation does not use a subnet, this request does not use a network domain ID.

```
$ curl -X POST  \
  $url/iaas/api/network-profiles?apiVersion=$api_version  \
  -H 'Content-Type: application/json'  \
  -H "Authorization: Bearer $access_token"  \
  -d '{
    "name":"example-network-profile-with-isolation-by-security-group",
    "description":"Example Network Profile",
    "regionId":"'$region_id'",
    "isolationType" : "SECURITY_GROUP",
    "fabricNetworkIds": [
     "c19bd2921af95075575931654066a",
     "8fe650cc09d0627558d55c9ba1793"
  ],
  "tags": [ { "key": "env", "value": "prod" } ]
}' | jq "."
```

A snippet of the response shows the network profile ID.

```
...
  "name": "example-network-profile-with-isolation-by-security-group",
  "description": "Example Network Profile",
  "id": "bdab0d4c28af6e7558f16c78f5468",
  "updatedAt": "2022-04-02",
...
```

Create a network profile with isolation using a VLAN transport zone.

```
$ curl -X POST  \
  $url/iaas/api/network-profiles?apiVersion=$api_version  \
  -H 'Content-Type: application/json'  \
  -H "Authorization: Bearer $access_token"  \
  -d '{
    "name":"example-network-profile-with-VLAN-transport-zone",
    "description":"Example Network Profile",
    "regionId":"'$region_id'",
    "isolationNetworkDomainId" : "233df662ec3b4875575931653ef00",
    "isolatedNetworkCIDRPrefix" : "27",
    "fabricNetworkIds": [
      "c19bd2921af95075575931654066a",
      "8fe650cc09d0627558d55c9ba1793"
    ],
    "tags": [ { "key": "env", "value": "prod" } ]
}' | jq "."
```

A snippet of the response shows the network profile ID.

```
...
  "name": "example-network-profile-with-VLAN-transport-zone",
  "description": "Example Network Profile",
  "id": "9cb2d111c768927558f043ec13d70",
  "updatedAt": "2022-04-02",
...
```

Create a Network Profile with Security Groups

To create a network profile with security groups, you make a POST request and provide security group IDs.

You create a network profile with security groups so that you can add firewall rules to all machines provisioned with that network profile.

Prerequisites

- Verify that all general prerequisites and prerequisites for the Automation Assembler Infrastructure as a Service (IaaS) service have been satisfied. See Chapter 3 Prerequisites for API Use Case Examples.
- Verify that you have the cloud account ID for the new cloud account that you added. See Adding Cloud Accounts.
- Verify that you have the region ID for the regions you want to include in the profile. See the procedure in Create Network Profiles.
- Verify that you have the IDs for the networks you want to include in the profile. See the procedure in Create Network Profiles.

Procedure

1 Assign the region ID variable.

```
region_id='<your_region_id>'
```

2 Filter for security groups associated with the cloud account ID and in the external region ID **us-east-1**.

```
curl -X GET -H 'Content-Type: application/
json' -H "Authorization: Bearer $access_token" "$url/
iaas/api/security-groups?apiVersion=$api_version&"'$filter='"externalRegionId%20eq%20'us-
east-1'%20and%20cloudAccountId%20eq%20'$cloud_account_id'" | jq "."
```

3 Examine the response to find the IDs for the security groups that you want to include in your network profile.

4 Create a network profile with security groups using network IDs for a non-public network.

```
curl -X POST \
  $url/iaas/api/network-profiles?apiVersion=$api_version \
  -H 'Content-Type: application/json' \
  -H "Authorization: Bearer $access_token" \
   -d '{
    "name":"example-network-profile-with-security-groups",
    "description":"Example Network Profile",
    "regionId":"'$region_id'",
    "fabricNetworkIds": [
      "<network_id1>",
      "<network_id1>"
    ],
    "securityGroupIds": [
      "<security_group_id1_from_response>",
      "<security_group_id2_from_response>"
    ],
    "tags": [ { "key": "env", "value": "prod" } ]
  }' | jq "."
```

The response shows the name and ID for the network profile you created.

Example: Create a network profile with security groups

Assign the required variables including a cloud account ID and a region ID.

```
$ url='https://appliance.domain.com'
$ api_version='2021-07-15'
$ cloud_account_id='c8c3c9bfdb449475-7f703c5265a63d87-5fa34c478df36b060e1ca3551254c403301
3bf3283908e4661cd1c6fb2f8b9ae-ce5aad01092b47558644f6b6615d'
$ region_id='37d6c1acf4a8275586468873c739'
```

Filter for security groups associated with the cloud account ID and in the external region ID **us-east-1**.

```
curl -X GET -H 'Content-Type: application/json' -H "Authorization: Bearer $access_token"
"$url/iaas/api/security-groups?
apiVersion=$api_version&"'$filter='"externalRegionId%20eq%20'us-
east-1'%20and%20cloudAccountId%20eq%20'$cloud_account_id'" | jq "."
```

A snippet of the response shows the ID for a security group that you can include in your network profile.

```
...
     "externalId": "sg-0305bc072a9f2727b",
      "name": "OC-LB-mcm681186-113024780265_SG",
      "id": "bdab0d4c28af6e7558f061f772518",
      "createdAt": "2022-04-02",
      "updatedAt": "2022-04-02",
      "organizationId": "8327d53f-91ea-420a-8613-ba8f3149db95",
      "orgId": "8327d53f-91ea-420a-8613-ba8f3149db95",
      "_links": {
        "cloud-accounts": {
          "hrefs": [
            "/iaas/api/cloud-accounts/
c8c3c9bfdb449475-7f703c5265a63d87-5fa34c478df36b060e1ca3551254c4033013bf3283908e4661cd1c6fb2f8
b9ae-ce5aad01092b47558644f6b6615d"
          ]
        },
        "self": {
          "href": "/iaas/api/security-groups/bdab0d4c28af6e7558f061f772518"
...
```

With the IDs of fabric networks that you want to include in your profile and and the security group IDs you want to include, create a network profile named **example-network-profile-with-security-groups**.

```
$ curl -X POST  \
  $url/iaas/api/network-profiles?apiVersion=$api_version  \
  -H 'Content-Type: application/json'  \
  -H "Authorization: Bearer $access_token"  \
  -d '{
    "name":"example-network-profile-with-security-groups",
    "description":"Example Network Profile",
    "regionId":"'$region_id'",
    "fabricNetworkIds": [
      "d43efed364ef18755759316540e3d",
      "d43efed364ef18755759316540e3f"
    ],
    "securityGroupIds": [
      "bdab0d4c28af6e7558f061f772518",
      "ebdab0d4c28af6e7558efe6edd71c9"
    ],
    "tags": [ { "key": "env", "value": "prod" } ]
}' | jq "."
```

A snippet of the response shows the network profile ID.

```
...
 "name": "example-network-profile-with-security-groups",
  "description": "Example Network Profile",
  "id": "9cb2d111c768927558f1799bf9e48",
  "updatedAt": "2022-04-02",
...
```

Using the Network APIs

As a Automation Assembler administrator, you must understand the difference between using the fabric network APIs and the network APIs.

You use the fabric network APIs such as `/iaas/api/fabric-networks/` to get information about existing networks that you use to create a network profile.

You use the network APIs such as `/iaas/api/networks/` to get information about on-demand networks that are provisioned with a deployment. If provisioned, you can check these networks in the Automation Assembler UI by going to **Resources > Deployments**.

To see a list of both public and on-demand networks in the Automation Assembler UI, go to **Infrastructure > Resources > Networks**.

Creating Storage Profiles

Using different input variables, you can use the Automation Assembler IaaS API to create storage profiles for AWS, vSphere, or Azure cloud accounts.

A Automation Assembler storage profile describes the kind of storage to be deployed. Storage is usually profiled according to characteristics such as service level or cost, performance, or purpose, such as backup.

As a cloud administrator, you organize your storage profiles under cloud-specific regions. One cloud account can have multiple regions, with multiple storage profiles under each. Using a storage profile, you define your storage for the region.

Storage profiles include disk customizations, and a means of identifying the type of storage by capability tags. Tags are matched against provisioning service request constraints to create the desired storage at deployment time.

Create an Amazon Web Services Storage Profile

To create a storage profile, you make a POST request with a region ID associated with a cloud account. The request body includes an Amazon Web Services fabric volume type.

As an alternative to using the `storage-profiles` API call to create an Amazon Web Services storage profile, you can also use the `storage-profiles-aws` API call. Optional procedure steps show how to use the `storage-profiles-aws` API call. The example only includes the steps required to create an Amazon Web Services storage profile using the `storage-profiles` API call.

Prerequisites

- Verify that all general prerequisites and prerequisites for the Automation Assembler Infrastructure as a Service (IaaS) service have been satisfied. See Chapter 3 Prerequisites for API Use Case Examples.
- Verify that you have the cloud account ID for the new cloud account that you added. See Adding Cloud Accounts.

Procedure

1 Assign the cloud account ID variable.

```
aws_cloud_account_id='<your_cloud_account_id>'
```

2 Look up region IDs associated with the cloud account and in the external region ID **`us-east-1`**.

```
curl -X GET -H 'Content-Type: application/json' -H "Authorization: Bearer $access_token"
"$url/iaas/api/regions/?apiVersion=$api_version&"'$filter'"=externalRegionId%20eq%20'us-
east-1'%20and%20cloudAccountId%20eq%20'"$aws_cloud_account_id"'"  | jq "."
```

3 Examine the response to find the ID for the region that you want.

4 Assign the region ID variable.

```
aws_region_id='<your_region_id>'
```

5 List all Amazon Web Services fabric volume types.

```
curl -X GET -H 'Content-Type: application/json' -H "Authorization: Bearer $access_token"
"$url/iaas/api/fabric-aws-volume-types/?apiVersion=$api_version" | jq "."
```

6 To select a volume type, examine the response.

7 Create a storage profile for the selected region.

```
curl -X POST \
  $url/iaas/api/storage-profiles?apiVersion=$api_version \
  -H 'Content-Type: application/json' \
  -H "Authorization: Bearer $access_token" \
  -d '{
    "defaultItem": false,
    "supportsEncryption": false,
    "tags": [ { "key": "env", "value": "dev" } ],
    "diskProperties": {
      "deviceType": "ebs",
```

```
      "volumeType": "<volume_type_from_response>",
      "iops": "400"
    },
    "regionId": "'$region_id'",
    "name": "<your-aws-storage-profile>",
    "description": "Example AWS storage profile"
}' | jq "."
```

8 (Optional) Create a storage profile for the selected region using the `storage-profiles-aws` API call.

```
curl -X POST \
  $url/iaas/api/storage-profiles-aws?apiVersion=$api_version \
  -H 'Content-Type: application/json' \
  -H "Authorization: Bearer $access_token" \
  -d '{
    "defaultItem": false,
    "supportsEncryption": false,
    "tags": [ { "key": "env", "value": "dev" } ]
    "deviceType": "ebs",
    "volumeType": "<volume_type_from_response>",
    "iops": "1000",
    "regionId": "'$region_id'",
    "name": "<your-aws-storage-profile>",
    "description": "Example AWS storage profile"
}' | jq "."
```

9 To obtain the storage profile ID, examine the response.

10 Assign the storage profile ID variable.

```
aws_storage_profile_id='<your_storage_profile_id>'
```

11 (Optional) Look up the storage profile you created with your storage profile ID.

```
curl -X GET -H 'Content-Type: application/json' -H "Authorization: Bearer $access_token"
$url/iaas/api/storage-profiles/$aws_storage_profile_id?apiVersion=$api_version | jq "."
```

The response shows the name and ID for the storage profile you created.

Note Using the external region ID and the cloud account ID, you can also filter for the storage profile with a query that does not require the storage profile ID. See Filtering Resources by Region ID.

12 (Optional) List all storage profiles using the `storage-profiles-aws` API call.

```
curl -X GET -H 'Content-Type: application/json' -H "Authorization: Bearer $access_token"
$url/iaas/api/storage-profiles-aws?apiVersion=$api_version | jq "."
```

13 (Optional) Delete an Amazon Web Services storage profile. Alternatively, you can use the `storage-profiles-aws` API call.

```
curl -X DELETE -H 'Content-Type: application/json'
-H "Authorization: Bearer $access_token" $url/iaas/api/storage-profiles/
$aws_storage_profile_id?apiVersion=$api_version | jq "."
```

Example: Create an Amazon Web Services storage profile

Assign the required variables including the cloud account ID for an Amazon Web Services cloud account.

```
$ url='https://appliance.domain.com'
$ api_version='2021-07-15'
$ aws_cloud_account_id='c8c3c9bfdb449475-7f703c5265a63d87-5fa34c478df36b060e1ca3551254c403301
3bf3283908e4661cd1c6fb2f8b9ae-ce5aad01092b47558644f6b6615d'
```

Look up region IDs associated with the cloud account and in the external region ID **`us-east-1`**.

```
$ curl -X GET -H 'Content-Type: application/json' -H "Authorization: Bearer $access_token"
"$url/iaas/api/regions/?apiVersion=$api_version&"'$filter'"=externalRegionId%20eq%20'us-
east-1'%20and%20cloudAccountId%20eq%20'"$aws_cloud_account_id"'"  | jq "."
```

A snippet of the response shows the region ID.

```
...
      "externalRegionId": "us-east-1",
      "cloudAccountId":
"c8c3c9bfdb449475-7f703c5265a63d87-5fa34c478df36b060e1ca3551254c4033013bf3283908e4661cd1c6fb2f
8b9ae-ce5aad01092b47558644f6b6615d",
      "id": "37d6c1acf4a8275586468873c739",
      "updatedAt": "2022-04-02",
...
```

Assign the Amazon Web Services region ID.

```
$ aws_region_id='37d6c1acf4a8275586468873c739'
```

List all Amazon Web Services fabric volume types.

```
$ curl -X GET -H 'Content-Type: application/json' -H "Authorization: Bearer $access_token"
"$url/iaas/api/fabric-aws-volume-types/?apiVersion=$api_version" | jq "."
```

A snippet of the response shows the volume types.

```
...
{
  "volumeTypes": [
    "standard",
    "io1",
    "gp2",
    "sc1",
    "st1"
```

```
  ]
},
...
```

Select volume type and create an AWS storage profile named **aws-storage-profile**.

```
$ curl -X POST \
  $url/iaas/api/storage-profiles?apiVersion=$api_version \
  -H 'Content-Type: application/json' \
  -H "Authorization: Bearer $access_token" \
  -d '{
    "defaultItem": false,
    "supportsEncryption": false,
    "tags": [ { "key": "env", "value": "dev" } ],
    "diskProperties": {
      "deviceType": "ebs",
      "volumeType": "io1",
      "iops": "400"
    },
    "regionId": "'$aws_region_id'",
    "name": "aws-storage-profile",
    "description": "Example AWS storage profile"
    "regionId":"'$aws_region_id'"
  }' | jq "."
```

A snippet of the response shows the storage profile ID.

```
...
  "externalRegionId": "us-east-1",
  "name": "aws-storage-profile",
  "description": "Example AWS storage profile",
  "id": "3e3dc378-a090-4b7e-af41-57b1735d9526",
  "createdAt": "2022-04-02",
  "updatedAt": "2022-04-02",
...
```

Create a vSphere Storage Profile

To create a vSphere storage profile, you make a POST request with a region ID. Optional request body input includes a vSphere storage policy and a vSphere datastore.

As an alternative to using the `storage-profiles` API call to create a vSphere storage profile, you can also use the `storage-profiles-vsphere` API call. Optional procedure steps show how to use the `storage-profiles-vsphere` API call. The example only includes the steps required to create a vSphere storage profile using the `storage-profiles` API call.

Prerequisites

- Verify that all general prerequisites and prerequisites for the Automation Assembler Infrastructure as a Service (IaaS) service have been satisfied. See Chapter 3 Prerequisites for API Use Case Examples.

- Verify that you have the cloud account ID for the new cloud account that you added. See Adding Cloud Accounts.

Procedure

1 Assign the cloud account ID variable.

```
vsphere_cloud_account_id='<your_cloud_account_id>'
```

2 Look up region IDs associated with the cloud account and in the external region ID **`Datacenter:datacenter-10`**.

```
curl -X GET -H 'Content-Type: application/json' -H "Authorization: Bearer $access_token"
"$url/iaas/api/regions/?
apiVersion=$api_version&"'$filter'"=externalRegionId%20eq%20'Datacenter:datacenter-10'%20an
d%20cloudAccountId%20eq%20'"$vsphere_cloud_account_id"'"  | jq "."
```

3 Examine the response to find the ID for the region that you want.

4 Assign the region ID variable.

```
vsphere_region_id='<your_region_id>'
```

5 (Optional) If using a vSphere storage policy, list all vSphere storage policies. If using a default storage policy, skip this step.

```
curl -X GET -H 'Content-Type: application/json' -H "Authorization: Bearer $access_token"
"$url/iaas/api/fabric-vsphere-storage-policies/?apiVersion=$api_version" | jq "."
```

Examine the response and assign the vSphere storage policy ID.

```
vsphere_storage_policy_id='<your_vsphere_storage_policy_id>'
```

6 (Optional) If using a vSphere datastore, list all vSphere datastores. If provisioning any datastore or cluster, skip this step.

```
curl -X GET -H 'Content-Type: application/json' -H "Authorization: Bearer $access_token"
"$url/iaas/api/fabric-vsphere-datastores/?apiVersion=$api_version" | jq "."
```

Examine the response and assign the vSphere datastore ID.

```
vsphere_datastore_id='<your_vsphere_datastore_id>'
```

7 Create a vSphere storage profile.

```
curl -X POST \
  $url/iaas/api/storage-profiles?apiVersion=$api_version \
  -H 'Content-Type: application/json' \
  -H "Authorization: Bearer $access_token" \
  -d '{
    "name": "<your-vsphere-storage-profile>",
    "description": "Example vSphere storage profile",
```

```
    "defaultItem": true,
    "supportsEncryption": true,
    "tags": [ { "key" : "env", "value": "dev" } ],
    "diskProperties": {
      "provisioningType": "thin",
      "independent":"true",
      "persistent":"true",
      "sharesLevel": "low",
      "shares": "500",
      "limitIops": "500"
    },
    "diskTargetProperties": {
      "storagePolicyId": "'$vsphere_storage_policy_id'",
      "datastoreId": "'$vsphere_datastore_id'" },
      "regionId": "'$vsphere_region_id'"
    }' | jq "."
```

8 To obtain the storage profile ID, examine the response.

9 Assign the storage profile ID variable.

```
vsphere_storage_profile_id='<your_storage_profile_id>'
```

10 (Optional) Look up the storage profile you created with your storage profile ID.

```
curl -X GET -H 'Content-Type: application/json' -H "Authorization: Bearer $access_token"
$url/iaas/api/storage-profiles/$vsphere_storage_profile_id?apiVersion=$api_version | jq "."
```

The response shows the name and ID for the storage profile you created.

Note Using the external region ID and the cloud account ID, you can also filter for the storage profile with a query that does not require the storage profile ID. See Filtering Resources by Region ID.

11 (Optional) List all storage profiles using the `storage-profiles-vsphere` API call. API.

```
curl -X GET -H 'Content-Type: application/json' -H "Authorization: Bearer $access_token"
$url/iaas/api/storage-profiles-vsphere?apiVersion=$api_version | jq "."
```

12 (Optional) Delete a vSphere storage profile. Alternatively, you can use the `storage-profiles-vsphere` API call..

```
curl -X DELETE -H 'Content-Type: application/json' -H "Authorization:
Bearer $access_token" $url/iaas/api/storage-profiles/$vsphere_storage_profile_id?
apiVersion=$api_version | jq "."
```

Example: Create vSphere storage profile

Assign the required variables including the cloud account ID for a vSphere cloud account.

```
$ url='https://appliance.domain.com'
$ api_version='2021-07-15'
$ vsphere_cloud_account_id='515684ccebafde75-7f703c5265a63d87-
e78aab87e9c8d5cd4cd1da1a285403f0f4e77a5240720d093e147b830b172542-23b5c527d7083675572f5099a8da0
'
```

Look up region IDs associated with the cloud account and in the external region ID **`Datacenter:datacenter-10`**.

```
$ curl -X GET -H 'Content-Type: application/json' -H "Authorization: Bearer $access_token"
"$url/iaas/api/regions/?
apiVersion=$api_version&"'$filter'"=externalRegionId%20eq%20'Datacenter:datacenter-10'%20and%2
0cloudAccountId%20eq%20'"$vsphere_cloud_account_id"'" | jq "."
```

A snippet of the response shows the region ID.

```
...
      "externalRegionId": "Datacenter:datacenter-10",
      "cloudAccountId":
"c8c3c9bfdb449475-7f703c5265a63d87-809fe6fef311fdd63aa6dac546574aa898213265e988e34cc851db19b8c
05b96-f405bb370210c875572d26445252e",
      "id": "cfb7246505319275572d26466a749",
...
```

Assign the vSphere region ID.

```
$ vsphere_region_id='cfb7246505319275572d26466a749'
```

If using a vSphere storage policy, perform the following steps.

1 List all vSphere storage policies.

```
$ curl -X GET -H 'Content-Type: application/json' -H "Authorization: Bearer $access_token"
"$url/iaas/api/fabric-vsphere-storage-policies/?apiVersion=$api_version" | jq "."
```

A snippet of the response shows the storage policy.

```
...
      "externalId": "f31f2442-8247-4517-87c2-8d69d7a6c696",
      "name": "Management Storage Policy - Stretched",
      "description": "Management Storage policy used for VMC stretched cluster",
      "id": "4aad51f0b02b5275572d264d28490",
...
```

2 Examine the response to assign the vSphere storage policy ID.

```
$ vsphere_storage_policy_id='4aad51f0b02b5275572d264d28490'
```

If provisioning a specific datastore or cluster, perform the following steps.

1 List all vSphere datastores.

```
$ curl -X GET -H 'Content-Type: application/json' -H "Authorization: Bearer $access_token"
"$url/iaas/api/fabric-vsphere-datastores/?apiVersion=$api_version" | jq "."
```

A snippet of the response shows the datastore.

```
...
      "externalId": "WorkloadDatastore",
      "name": "WorkloadDatastore",
      "id": "c4f1dd4741d05e75572d264dcc590",
      "createdAt": "2022-04-02",
      "updatedAt": "2022-04-02",
...
```

2 Examine the response to assign the vSphere datastore ID.

```
$ vsphere_datastore_id='c4f1dd4741d05e75572d264dcc590'
```

Create a vSphere storage profile named **vsphere-storage-profile**.

```
$ curl -X POST \
  $url/iaas/api/storage-profiles?apiVersion=$api_version \
  -H 'Content-Type: application/json' \
  -H "Authorization: Bearer $access_token" \
  -d '{
    "name": "vsphere-storage-profile",
    "description": "Example vSphere storage profile",
    "defaultItem": true,
    "supportsEncryption": true,
    "tags": [ { "key" : "env", "value": "dev" } ],
    "diskProperties": {
      "provisioningType": "thin",
      "independent":"true",
      "persistent":"true",
      "sharesLevel": "low",
      "shares": "500",
      "limitIops": "500"
    },
    "diskTargetProperties": {
      "storagePolicyId": "'$vsphere_storage_policy_id'",
      "datastoreId": "'$vsphere_datastore_id'"
    },
    "regionId": "'$vsphere_region_id'"
}' | jq "."
```

A snippet of the response shows the storage profile ID.

```
...
  "externalRegionId": "Datacenter:datacenter-10",
  "name": "vsphere-storage-profile",
  "description": "Example vSphere storage profile",
```

```
  "id": "b4fbd25e-a2dd-4fde-9186-0f7bd34a1df2",
  "createdAt": "2022-04-02",
  "updatedAt": "2022-04-02",
...
```

Create a vSphere Storage Profile for a First Class Disk

To create a vSphere Storage Profile that supports First Class Disk (FCD) storage, you make a POST request with a region ID and you include first class as the disk type property. Optional request body input includes a vSphere storage policy and a vSphere datastore.

Prerequisites

- Verify that all general prerequisites and prerequisites for the Automation Assembler Infrastructure as a Service (IaaS) service have been satisfied. See Chapter 3 Prerequisites for API Use Case Examples.
- Verify that you have the cloud account ID for the vSphere cloud account that you added. See Add a vSphere Cloud Account .

Procedure

1 Assign the cloud account ID variable.

```
vsphere_cloud_account_id='<your_cloud_account_id>'
```

2 Look up region IDs associated with the cloud account and in the external region ID **`Datacenter:datacenter-3`**.

```
curl -X GET -H 'Content-Type: application/json' -H "Authorization: Bearer $access_token"
"$url/iaas/api/regions/?
apiVersion=$api_version&"'$filter'"=externalRegionId%20eq%20'Datacenter:datacenter-3'%20and
%20cloudAccountId%20eq%20'"$vsphere_cloud_account_id"'"  | jq "."
```

3 Examine the response to find the ID for the region that you want.

4 Assign the region ID variable.

```
vsphere_region_id='<your_region_id>'
```

5 (Optional) If using a vSphere storage policy, list all vSphere storage policies. If using a default storage policy, skip this step.

```
curl -X GET -H 'Content-Type: application/json' -H "Authorization: Bearer $access_token"
"$url/iaas/api/fabric-vsphere-storage-policies/?apiVersion=$api_version" | jq "."
```

Examine the response and assign the vSphere storage policy ID.

```
vsphere_storage_policy_id='<your_vsphere_storage_policy_id>'
```

6 (Optional) If using a vSphere datastore, list all vSphere datastores. If provisioning any datastore or cluster, skip this step.

```
curl -X GET -H 'Content-Type: application/json' -H "Authorization: Bearer $access_token"
"$url/iaas/api/fabric-vsphere-datastores/?apiVersion=$api_version" | jq "."
```

Examine the response and assign the vSphere datastore ID.

```
vsphere_datastore_id='<your_vsphere_datastore_id>'
```

7 Create a vSphere Storage Profile with the FCD property.

```
curl -X POST \
  $url/iaas/api/storage-profiles?apiVersion=$api_version \
  -H 'Content-Type: application/json' \
  -H "Authorization: Bearer $access_token" \
  -d '{
    "name": "<your_vsphere_storage_profile_with_FCD>",
    "description": "Example First Class Disk vSphere Storage Profile",
    "defaultItem": true,
    "provisioningType": "thin",
    "diskType": "firstClass",
    "regionId": "'$vsphere_region_id'",
    "tags": [ { "key": "type", "value": "fcd" } ]
    }' | jq "."
```

8 Examine the response.

- `"defaultItem": true` indicates that this storage profile is the default for the region.
- Tags help you to locate, manage, and work with the infrastructure resources.

Example: Create vSphere Storage Profile with FCD storage

Assign the required variables including the cloud account ID for a vSphere cloud account.

```
$ url='https://appliance.domain.com'
$ api_version='2022-04-02'
$ vsphere_cloud_account_id='683c647b-413d-4673-a236-08b3694cd652'
```

Look up region IDs associated with the cloud account and in the external region ID **`Datacenter:datacenter-3`**.

```
$ curl -X GET -H 'Content-Type: application/json' -H "Authorization: Bearer $access_token"
"$url/iaas/api/regions/?
apiVersion=$api_version&"'$filter'"=externalRegionId%20eq%20'Datacenter:datacenter-10'%20and%2
0cloudAccountId%20eq%20'"$vsphere_cloud_account_id"'" | jq "."
```

A snippet of the response shows the region ID.

```
...
      "externalRegionId": "Datacenter:datacenter-3",
      "name": "Example external region name",
      "cloudAccountId": "683c647b-413d-4673-a236-08b3694cd652",
```

```
      "id": "0f182edc-1155-4df1-a53a-2c46be7bc373",
...
```

Assign the vSphere region ID.

```
$ vsphere_region_id='0f182edc-1155-4df1-a53a-2c46be7bc373'
```

If using a vSphere storage policy, perform the following steps.

1 List all vSphere storage policies.

```
$ curl -X GET -H 'Content-Type: application/json' -H "Authorization: Bearer $access_token"
"$url/iaas/api/fabric-vsphere-storage-policies/?apiVersion=$api_version" | jq "."
```

A snippet of the response shows the storage policy.

```
...
      "externalId": "f31f2442-8247-4517-87c2-8d69d7a6c696",
      "name": "Management Storage Policy - Stretched",
      "description": "Management Storage policy used for VMC stretched cluster",
      "id": "4aad51f0b02b5275572d264d28490",
...
```

2 Examine the response to assign the vSphere storage policy ID.

```
$ vsphere_storage_policy_id='4aad51f0b02b5275572d264d28490'
```

If provisioning a specific datastore or cluster, perform the following steps.

1 List all vSphere datastores.

```
$ curl -X GET -H 'Content-Type: application/json' -H "Authorization: Bearer $access_token"
"$url/iaas/api/fabric-vsphere-datastores/?apiVersion=$api_version" | jq "."
```

A snippet of the response shows the datastore.

```
...
      "externalId": "WorkloadDatastore",
      "name": "WorkloadDatastore",
      "id": "c4f1dd4741d05e75572d264dcc590",
      "createdAt": "2022-04-02",
      "updatedAt": "2022-04-02",
...
```

2 Examine the response to assign the vSphere datastore ID.

```
$ vsphere_datastore_id='c4f1dd4741d05e75572d264dcc590'
```

Create a vSphere Storage Profile named **`vsphere-storage-profile-with-FCD`**.

```
$ curl -X POST \
  $url/iaas/api/storage-profiles?apiVersion=$api_version \
  -H 'Content-Type: application/json' \
```

```
  -H "Authorization: Bearer $access_token" \
  -d '{
    "name": "vsphere-storage-profile-with-FCD",
    "description": "Example First Class Disk vSphere Storage Profile",
    "defaultItem": true,
    "provisioningType": "thin",
    "diskType": "firstClass",
    "regionId": "'$vsphere_region_id'",
    "tags": [ { "key": "type", "value": "fcd" } ]
}' | jq "."
```

Included with storage profile ID, a snippet of the response shows the tags that you defined for the storage profile.

```
{
  "defaultItem": true,
  "tags": [
    {
      "key": "type",
      "value": "fcd"
    }
  ],
  "provisioningType": "thin",
  "externalRegionId": "Datacenter:datacenter-3",
  "cloudAccountId": "683c647b-413d-4673-a236-08b3694cd652",
  "diskType": "firstClass",
  "name": "vsphere-storage-profile-with-FCD",
  "description": "Example First Class Disk vSphere Storage Profile",
  "id": "6037ac02-83e0-41bb-ba6e-ed5784ae1101",
  "createdAt": "2022-04-02",
  "updatedAt": "2022-04-02",
...
```

What to do next

Use the tags to create a First Class Disk. See Create a First Class Disk.

Create a Microsoft Azure Storage Profile

To create a Microsoft Azure storage profile, you make a POST request with a region ID. The request body includes a Microsoft Azure fabric storage account ID.

As an alternative to using the `storage-profiles` API call to create a Microsoft Azure storage profile, you can also use the `storage-profiles-azure` API call. Optional procedure steps show how to use the `storage-profiles-azure` API call. The example only includes the steps required to create a Microsoft Azure storage profile using the `storage-profiles` API call.

Prerequisites

- Verify that all general prerequisites and prerequisites for the Automation Assembler Infrastructure as a Service (IaaS) service have been satisfied. See Chapter 3 Prerequisites for API Use Case Examples.

- Verify that you have the cloud account ID for the new cloud account that you added. See Adding Cloud Accounts.

Procedure

1 Assign the cloud account ID variable.

```
azure_cloud_account_id='<your_cloud_account_id>'
```

2 Look up region IDs associated with the cloud account.

```
curl -X GET -H 'Content-Type: application/json' -H "Authorization: Bearer $access_token"
"$url/iaas/api/regions/?apiVersion=$api_version&"'$filter'"=externalRegionId%20eq%20'us-
east-1'%20and%20cloudAccountId%20eq%20'"$zaure_cloud_account_id"'"  | jq "."
```

3 Examine the response to find the ID for the region that you want.

4 Assign the region ID variable.

```
azure_region_id='<your_region_id>'
```

5 List all Azure fabric storage accounts.

```
curl -X GET -H 'Content-Type: application/json' -H "Authorization: Bearer $access_token"
"$url/iaas/api/fabric-azure-storage-accounts/?apiVersion=$api_version" | jq "."
```

6 To select a storage account ID, examine the response.

7 Create a storage profile for the selected region.

```
curl -X POST \
  $url/iaas/api/storage-profiles?apiVersion=$api_version \
  -H 'Content-Type: application/json' \
  -H "Authorization: Bearer $access_token" \
  -d '{
    "defaultItem": false,
    "supportsEncryption": false,
    "tags": [ { "key": "env", "value": "dev" } ],
    "diskProperties": {
    "azureOsDiskCaching": "None",
    "azureDataDiskCaching": "None"
  },
  "diskTargetProperties": { "storageAccountId": "<storage_account_id_from_response>" },
  "regionId": "'$azure_region_id'",
  "name": "<your-azure-storage-profile>",
  "description": "Example Azure storage profile"
}' | jq "."
```

8 (Optional) Create a storage profile for the selected region using the `storage-profiles-azure` API call.

```
curl -X POST \
  $url/iaas/api/storage-profiles-azure?apiVersion=$api_version \
  -H 'Content-Type: application/json' \
```

```
  -H "Authorization: Bearer $access_token" \
  -d '{
    "defaultItem": false,
    "supportsEncryption": false,
    "osDiskCaching": "None",
    "dataDiskCaching": "None",
    "storageAccountId": "<storage_account_id_from_response>",
    "regionId": "'$azure_region_id'",
    "name": "<your-azure-storage-profile>",
    "description": "Example Azure storage profile"
    "tags": [ { "key": "env", "value": "dev" } ]
}' | jq "."
```

9 To obtain the storage profile ID, examine the response.

10 Assign the storage profile ID variable.

```
azure_storage_profile_id='<your_storage_profile_id>'
```

11 (Optional) Look up the storage profile you created with your storage profile ID.

```
curl -X GET -H 'Content-Type: application/json' -H "Authorization: Bearer $access_token"
$url/iaas/api/storage-profiles/$azure_storage_profile_id?apiVersion=$api_version | jq "."
```

The response shows the name and ID for the storage profile you created.

Note Using the external region ID and the cloud account ID, you can also filter for the storage profile with a query that does not require the storage profile ID. See Filtering Resources by Region ID.

12 (Optional) List all Azure storage profiles using the `storage-profiles-azure` API call.

```
curl -X GET -H 'Content-Type: application/json' -H "Authorization: Bearer $access_token"
$url/iaas/api/storage-profiles-azure?apiVersion=$api_version | jq "."
```

13 (Optional) Delete a Microsoft Azure storage profile. Alternatively, you can use the `storage-profiles-azure` API call.

```
curl -X DELETE -H 'Content-Type: application/json' -H "Authorization:
Bearer $access_token" $url/iaas/api/storage-profiles/$azure_storage_profile_id?
apiVersion=$api_version | jq "."
```

Example: Create a Microsoft Azure storage profile

Assign the required variables including the cloud account ID for a Microsoft Azure cloud account.

```
$ url='https://appliance.domain.com'
$ api_version='2021-07-15'
$ azure_cloud_account_id='c8c3c9bfdb449475-7f703c5265a63d87-5fa34c478df36b060e1ca3551254
c4033013bf3283908e4661cd1c6fb2f8b9ae-ce5aad01092b47558644f6b6615d'
```

Look up region IDs associated with the cloud account and in the external region ID **us-east-1**

```
$ curl -X GET -H 'Content-Type: application/json' -H "Authorization: Bearer $access_token"
"$url/iaas/api/regions/?apiVersion=$api_version&"'$filter'"=externalRegionId%20eq%20'us-
east-1'%20and%20cloudAccountId%20eq%20'"$azure_cloud_account_id"'"  | jq "."
```

A snippet of the response shows the region ID.

```
...
      "externalRegionId": "us-east-1",
      "cloudAccountId":
"c8c3c9bfdb449475-7f703c5265a63d87-5fa34c478df36b060e1ca3551254c4033013bf3283908e4661cd1c6fb2f
8b9ae-ce5aad01092b47558644f6b6615d",
      "id": "37d6c1acf4a8275586468873c739",
      "updatedAt": "2022-04-02",
...
```

Assign the Azure region ID.

```
$ azure_region_id='37d6c1acf4a8275586468873c739'
```

List all fabric storage accounts.

```
$ curl -X GET -H 'Content-Type: application/json' -H "Authorization: Bearer $access_token"
"$url/iaas/api/fabric-azure-storage-accounts/?apiVersion=$api_version" | jq "."
```

A snippet of the response shows the storage accounts.

```
...
      "externalId": "/subscriptions/b8ef63a7-a5e3-44fa-8745-1ead33fa1f25/resourceGroups/
default-rg/providers/Microsoft.Storage/storageAccounts/azbasicsa80370",
      "name": "azbasicsa80370",
      "id": "f81c26bf-51b1-49cc-865c-de2ab3821c1d",
...
```

Select storage account ID and create an Azure storage profile named **azure-storage-profile**.

```
$ curl -X POST \
  $url/iaas/api/storage-profiles?apiVersion=$api_version \
  -H 'Content-Type: application/json' \
  -H "Authorization: Bearer $access_token" \
  -d '{
    "defaultItem": false,
    "supportsEncryption": false,
    "tags": [ { "key": "env", "value": "dev" } ],
    "diskProperties": {
    "azureOsDiskCaching": "None",
    "azureDataDiskCaching": "None"
  },
  "diskTargetProperties": { "storageAccountId": "f81c26bf-51b1-49cc-865c-de2ab3821c1d" },
  "regionId": "'$azure_region_id'",
  "name": "azure-storage-profile",
```

```
  "description": "Example Azure storage profile"
}' | jq "."
```

A snippet of the response shows the storage profile ID.

```
...
  "externalRegionId": "us-east-1",
  "name": "azure-storage-profile",
  "description": "Example Azure storage profile",
  "id": "f83d0fd4-45de-4ca7-a699-c98bc141ecaa",
  "createdAt": "2022-04-02",
  "updatedAt": "2022-04-02",
...
```

Create a Microsoft Azure Storage Profile for a Managed Disk

To create a Microsoft Azure storage profile for a managed disk, you make a POST request with a region ID and include disk properties to specify the managed disk type.

Prerequisites

- Verify that all general prerequisites and prerequisites for the Automation Assembler Infrastructure as a Service (IaaS) service have been satisfied. See Chapter 3 Prerequisites for API Use Case Examples.
- Verify that you have the cloud account ID for the new cloud account that you added. See Adding Cloud Accounts.

Procedure

1 Assign the cloud account ID variable.

```
azure_cloud_account_id='<your_cloud_account_id>'
```

2 Look up region IDs associated with the cloud account and in the external region ID **eastus**.

```
curl -X GET -H 'Content-Type: application/json' -H "Authorization: Bearer $access_token"
"$url/iaas/api/regions/?
apiVersion=$api_version&"'$filter'"=externalRegionId%20eq%20'eastus'%20and%20cloudAccountId
%20eq%20'"$zaure_cloud_account_id"'"  | jq "."
```

3 Examine the response to find the ID for the region that you want.

4 Assign the region ID variable.

```
azure_region_id='<your_region_id>'
```

5 Create a storage profile for the selected region.

```
curl -X POST \
  $url/iaas/api/storage-profiles?apiVersion=$api_version \
  -H 'Content-Type: application/json' \
  -H "Authorization: Bearer $access_token" \
```

```
  -d '{
    "defaultItem": false,
    "supportsEncryption": false,
    "tags": [ { "key": "type", "value": "managed" } ],
    "diskProperties": {
       "azureManagedDiskType": "Standard_LRS",
       "azureOsDiskCaching": "ReadWrite",
       "azureDataDiskCaching": "ReadWrite"
  },
  "regionId": "'$azure_region_id'",
  "name": "<your-azure-managed-disk-storage-profile>",
  "description": "Example Azure managed disk"
}' | jq "."
```

6 To obtain the storage profile ID, examine the response.

7 Assign the storage profile ID variable.

```
azure_storage_profile_id='<your_storage_profile_id>'
```

8 (Optional) Look up the storage profile you created with your storage profile ID.

```
curl -X GET -H 'Content-Type: application/json' -H "Authorization: Bearer $access_token"
$url/iaas/api/storage-profiles/$azure_storage_profile_id?apiVersion=$api_version | jq "."
```

The response shows the name and ID for the storage profile you created.

Note Using the external region ID and the cloud account ID, you can also filter for the storage profile with a query that does not require the storage profile ID. See Filtering Resources by Region ID.

Example: Create a Microsoft Azure storage profile

Assign the required variables including the cloud account ID for a Microsoft Azure cloud account.

```
$ url='https://appliance.domain.com'
$ api_version='2021-07-15'
$ azure_cloud_account_id='c8c3c9bfdb449475-7f703c5265a63d87-5fa34c478df36b060e1ca3551254
c4033013bf3283908e4661cd1c6fb2f8b9ae-ce5aad01092b47558644f6b6615d'
```

Look up region IDs associated with the cloud account and in the external region ID **eastus**

```
$ curl -X GET -H 'Content-Type: application/json' -H "Authorization: Bearer $access_token"
"$url/iaas/api/regions/?apiVersion=$api_version&"'$filter'"=externalRegionId%20eq%20'us-
east-1'%20and%20cloudAccountId%20eq%20'"$azure_cloud_account_id"'"  | jq "."
```

A snippet of the response shows the region ID.

```
...
      "externalRegionId": "eastus",
      "cloudAccountId":
"c8c3c9bfdb449475-7f703c5265a63d87-5fa34c478df36b060e1ca3551254c4033013bf3283908e4661cd1c6fb2f
8b9ae-ce5aad01092b47558644f6b6615d",
```

```
      "id": "20d6c1abc4a8275586468873c721",
      "updatedAt": "2022-04-02",
...
```

Assign the Azure region ID.

```
$ azure_region_id='20d6c1abc4a8275586468873c721'
```

Create a Microsoft Azure storage profile named **azure-with-managed-disks-storage-profile**.

```
$ curl -X POST \
  $url/iaas/api/storage-profiles?apiVersion=$api_version \
  -H 'Content-Type: application/json' \
  -H "Authorization: Bearer $access_token" \
  -d '{
    "defaultItem": false,
    "supportsEncryption": false,
    "tags": [ { "key": "type", "value": "managed" } ],
    "diskProperties": {
       "azureManagedDiskType": "Standard_LRS",
       "azureOsDiskCaching": "ReadWrite",
       "azureDataDiskCaching": "ReadWrite"
  },
  "regionId": "'$azure_region_id'",
  "name": "azure-with-managed-disks-storage-profile",
  "description": "Example Azure with managed disks storage profile"
}' | jq "."
```

A snippet of the response shows the storage profile ID.

```
...
  "externalRegionId": "eastus",
  "name": "azure-with-managed-disks-storage-profile",
  "description": "Example Azure with managed disks storage profile",
  "id": "f83d0fd4-45de-4ca7-a699-c20bc121abcd",
  "createdAt": "2022-04-02",
  "updatedAt": "2022-04-022021-08-02",
...
```

Managing Your Projects Using the Project APIs

7

You use the Project APIs to manage projects that can be used in any of the Automation component services. In addition to basic operations such as creating, updating, or deleting projects, you can use the Project APIs to modify principals assigned to a project or modify a project cost.

Read the following topics next:

- Create a Project with the Project Service API
- Add Users to Your Project Using the Project Service API

Create a Project with the Project Service API

Using the Project Service API, you can create a project. You can also modify or delete a project and list all projects in an organization.

Before creating a project, it is a good practice to get a list of projects so that you can verify that the project you plan to create does not exist. Then you create the project with users assigned to project roles.

Prerequisites

- Verify that all general prerequisites and prerequisites for the Project service have been satisfied. See Chapter 3 Prerequisites for API Use Case Examples.
- Verify that the project roles that you plan to assign have sufficient permissions to perform project-related tasks.

  Note A user with the project administrator or project member role can perform a limited number of project-related tasks. For a complete list of tasks and roles required, see Organization and service user roles in VMware Aria Automation.

- Prepare parameters including the project name, description, and email addresses for administrators, members, or viewers.

Procedure

1 Get a list of projects.

```
curl -X GET -H 'Accept: application/json' -H "Authorization: Bearer $access_token" "$url/
project-service/api/projects?apiVersion=$api_version" | jq "."
```

2 To verify that the project you plan to create is not already listed, examine the response.

3 Assign the project name variable.

```
project_name='<your_project_name>'
```

your_project_name is a name that you choose.

4 Create a project.

```
curl -X POST \
  "$url/project-service/api/projects?apiVersion=$api_version" \
  -H 'Content-Type: application/json' \
  -H "Authorization: Bearer $access_token" \
  -d '{
    "name" : "'$project_name'",
    "description" : "your-description",
    "administrators" : [{ "email" : "<admin_email>", ["type" : <"user" | "group">]}],
    "members" : [{ "email" : "<member_email>", ["type" : <"user" | "group">]}],
    "viewers" : [{ "email" : "<viewer_email>", ["type" : <"user" | "group">]}],
  }' | jq "."
```

- *admin_email*, *member_email*, and *viewer_email* are email addresses of an administrator, member, and viewer user or name of the group in the project.
- The type parameter is optional. It assigns the administrator, member, or viewer to a user or group type. If unspecified, the value defaults to `user`.

Example: Create a Project

Create a project named **`Example-project`** with administrators, members, and viewers at **`mycompany.com`**. This example assumes that **`Example-project`** does not exist.

```
$ url='https://appliance.domain.com'
$ api_version='2019-01-15'
$ project_name='Example-project'
```

Create a project with an administrator, member, and viewer assigned to user type roles.

```
$ curl -X POST \
  "$url/project-service/api/projects?apiVersion=$api_version" \
  -H 'Content-Type: application/json' \
  -H "Authorization: Bearer $access_token" \
  -d '{
    "name" : "'$project_name'",
    "description" : "This is an example project",
    "administrators" : [{"email" : "adminX@mycompany.com", "type" : "user"}],
```

```
    "members" : [{"email" : "memberX@mycompany.com", "type" : "user"}],
    "viewers" : [{"email" : "viewerX@mycompany.com", "type" : "user"}]
  }' | jq "."
```

The response shows the administrators, members, and viewers related to the project and the project ID.

```
{
      "id": "094a2fab-7715-4844-94f9-71b45452da27",
      "name": "Example-project",
      "description": "This is an example project",
      "orgId": "f670fdfc-66d6-4689-9793-d524e7066d1e",
      "administrators": [
        {
          "email": "adminX@mycompany.com",
          "type": "user"
        }
      "members": [
        {
          "email": "memberX@mycompany.com",
          "type": "user"
        },
      ],
      "viewers": [
        {
          "email": "viewerX@mycompany.com",
          "type": "user"
        }
      ]
      "supervisors": [],
      "constraints": {
        "network": {
          "conditions": []
        }
      },
      "properties": {},
      "cost": {
        "cost": 0,
        "costSyncTime": "2019-05-13T12:47:10.624Z",
        "costUnit": "USD"
      },
      "operationTimeout": 0,
      "sharedResources": true
    },
...
```

Add Users to Your Project Using the Project Service API

As a service administrator, you can use a PATCH request to add, modify, or remove a project user.

Prerequisites

- Verify that all general prerequisites and prerequisites for the Project service have been satisfied. See Chapter 3 Prerequisites for API Use Case Examples.
- Verify that you have the project administrator role in your project and you have the project ID. See Create a Project with the Project Service API.
- Verify that the project roles that you plan to assign have sufficient permissions to perform project-related tasks.

  Note A user with the project administrator or project member role can perform a limited number of project-related tasks. For a complete list of tasks and roles required, see Organization and service user roles in VMware Aria Automation.

- Prepare parameters including additional email addresses for administrators, members, or viewers that you want to add to the project.

Procedure

1 Assign the project ID variable.

```
project_id='<your_project_id>'
```

your_project_id is the ID of the new project you created.

2 List the details of your project.

```
curl -X GET -H 'Accept: application/json' -H "Authorization: Bearer $access_token" "$url/
project-service/api/projects/$project_id?apiVersion=$api_version" | jq "."
```

3 Examine the response to see the administrators and users who are already in your project.

4 Submit a request to add a new project administrator.

```
curl -X PATCH \
  "$url/project-service/api/projects/$project_id/principals?apiVersion=$api_version" \
  -H 'Content-Type: application/json' \
  -H "Authorization: Bearer $access_token" \
  -d '{
    "modify" : [
      {
        "email" : "<your_new_administrator_email>",
        "role" : "administrator",
        "type" : "user"
      }
    ]
  }' | jq "."
```

5 Submit a request to add a new project member.

```
curl -X PATCH \
  "$url/project-service/api/projects/$project_id/principals?apiVersion=$api_version" \
  -H 'Content-Type: application/json' \
```

```
  -H "Authorization: Bearer $access_token" \
  -d '{
    "modify" : [
      {
        "email" : "<your_new_member_email>",
        "role" : "member",
        "type" : "user"
      }
    ]
  }' | jq "."
```

Example: Add Users to Your Project

For the project **`Example-project`**, add a user with the administrator role and a user with the member role at **`mycompany.com`**.

```
$ url='https://appliance.domain.com'
$ api_version='2019-01-15'
$ project_id='094a2fab-7715-4844-94f9-71b45452da27'
```

List the details of your project.

```
$ curl -X GET -H 'Accept: application/json' -H "Authorization: Bearer $access_token" "$url/
project-service/api/projects/$project_id?apiVersion=$api_version" | jq "."
```

The response shows existing administrators, members, and viewers.

```
{
      "id": "094a2fab-7715-4844-94f9-71b45452da27",
      "name": "Example-project",
      "description": "This is an example project",
      "orgId": "f670fdfc-66d6-4689-9793-d524e7066d1e",
      "administrators": [
        {
          "email": "adminX@mycompany.com",
          "type": "user"
        }
      "members": [
        {
          "email": "memberX@mycompany.com",
          "type": "user"
        },
      ],
      "viewers": [
        {
          "email": "viewerX@mycompany.com",
          "type": "user"
        }
      ]
      "supervisors": [],
      "constraints": {
        "network": {
          "conditions": []
        }
```

```
      },
      "properties": {},
      "cost": {
        "cost": 0,
        "costSyncTime": "2019-05-13T12:47:10.624Z",
        "costUnit": "USD"
      },
      "operationTimeout": 0,
      "sharedResources": true
    },
...
```

Add the administrator.

```
curl -X PATCH \
  "$url/project-service/api/projects/$project_id/principals?apiVersion=$api_version" \
  -H 'Content-Type: application/json' \
  -H "Authorization: Bearer $access_token" \
  -d '{
    "modify" : [
      {
        "email" : "newadminX@mycompany.com",
        "role" : "administrator",
        "type" : "user"
      }
    ]
  }' | jq "."
```

Add the member.

```
curl -X PATCH \
  "$url/project-service/api/projects/$project_id/principals?apiVersion=$api_version" \
  -H 'Content-Type: application/json' \
  -H "Authorization: Bearer $access_token" \
  -d '{
    "modify" : [
      {
        "email" : "newmemberX@mycompany.com",
        "role" : "member",
        "type" : "user"
      }
    ]
  }' | jq "."
```

The response lists the project including the added administrator and added member.

```
{
      "id": "094a2fab-7715-4844-94f9-71b45452da27",
      "name": "Example-project",
      "description": "This is an example project",
      "orgId": "f670fdfc-66d6-4689-9793-d524e7066d1e",
      "administrators": [
        {
          "email": "newadminX@mycompany.com",
```

```
          "type": "user"
        }
        {
          "email": "adminX@mycompany.com",
          "type": "user"
        }
      "members": [
        {
          "email": "newmemberX@mycompany.com",
          "type": "user"
        }
        {
          "email": "memberX@mycompany.com",
          "type": "user"
        },
      ],
      "viewers": [
        {
          "email": "viewerX@mycompany.com",
          "type": "user"
        }
      ]
      "supervisors": [],
      "constraints": {
        "network": {
          "conditions": []
        }
      },
      "properties": {},
      "cost": {
        "cost": 0,
        "costSyncTime": "2019-05-13T12:47:10.624Z",
        "costUnit": "USD"
      },
      "operationTimeout": 0,
      "sharedResources": true
    }
```

Working with Blueprints/Cloud Templates

8

To create and update cloud templates, version cloud templates, and deploy cloud templates, you use the Automation Assembler Blueprint APIs. Blueprint is the term used in the API specifications. In the product, blueprints are renamed to Automation Assembler Templates.

Read the following topics next:

- Create and Update a Cloud Template
- Setting up Policies
- Version and Release a Cloud Template to a VMware Aria Automation Service Broker Catalog
- Edit and Version a Custom Form in your Cloud Template
- Remove a Cloud Template Version from a VMware Aria Automation Service Broker Catalog
- Test Your Cloud Template Deployment
- Deploy Your Cloud Template
- Specify SCSI disk placement using the Automation API
- How to Create Custom Naming Templates

Create and Update a Cloud Template

To create a cloud template, you make a POST request. The request body includes the name of the new cloud template and the project ID of an existing project. To update a cloud template, you make a PUT request that changes one of the properties of the cloud template.

Before creating a cloud template, you get a list of cloud templates to verify that the cloud template you plan to create does not already exist. After a cloud template is created, you can update it to change the cloud template definition.

Prerequisites

- Verify that all general prerequisites and prerequisites for the Automation Assembler Blueprint service have been satisfied. See Chapter 3 Prerequisites for API Use Case Examples.
- Verify that you have a project ID for the project that includes cloud zones configured to support the resource requirements of your cloud template. See Create a Project to use in Automation Assembler.

Procedure

1 Assign the project ID variable.

```
project_id='<your_project_id>'
```

2 Assign your cloud template name variable.

```
cloud_template_name='<your_cloud_template_name>'
```

your_cloud_template_name is a name that you choose.

3 Get a list of cloud templates.

```
curl -X GET $url/blueprint/api/blueprints -H "Authorization: Bearer $access_token"  | jq
"."
```

4 To verify that the cloud template you plan to create is not already listed, examine the response.

5 Validate the cloud template before creating it.

```
curl -X POST \
  $url/blueprint/api/blueprint-validation?apiVersion=$api_version \
  -H 'Content-Type: application/json' \
  -H "Authorization: Bearer $access_token"  \
  -d '{
    "name" : "'"$cloud_template_name"'",
    "description" : "Basic Cloud Machine cloud template",
    "content" : "formatVersion: 1\ninputs:\n  flavor:\n    type: string\n
title: Flavor\n    description: Flavor Mapping Name\n  image:\n    type:
string\n    title: Image\n    description: Image Mapping Name\n  count:\n
type: integer\n    minimum: 1\n    default: 1\n    maximum: 2\n    title:
Number of Instances\nresources:\n  BasicCloudMachine:\n    type: Cloud.Machine\n
properties:\n      name: BasicCloudMachine\n      flavor: '\''${input.flavor}'\''
\n      image: '\''${input.image}'\''\n      count: '\''${input.count}'\''",
    "projectId" : "'"$project_id"'",
    "requestScopeOrg": false
}' | jq "."
```

6 Examine the response to confirm that you see `"valid":true`.

7 Create a new cloud template.

```
curl -X POST \
  $url/blueprint/api/blueprints?apiVersion=$api_version \
  -H 'Content-Type: application/json' \
  -H "Authorization: Bearer $access_token"  \
  -d '{
    "name" : "'"$cloud_template_name"'",
    "description" : "Basic Cloud Machine cloud template",
    "content" : "formatVersion: 1\ninputs:\n  flavor:\n    type: string\n
title: Flavor\n    description: Flavor Mapping Name\n  image:\n    type:
string\n    title: Image\n    description: Image Mapping Name\n  count:\n
```

```
type: integer\n    minimum: 1\n    default: 1\n    maximum: 2\n    title:
Number of Instances\nresources:\n  BasicCloudMachine:\n    type: Cloud.Machine\n
properties:\n      name: BasicCloudMachine\n      flavor: '\''${input.flavor}'\''
\n      image: '\''${input.image}'\''\n      count: '\''${input.count}'\''",
    "projectId" : "'"$project_id"'",
    "requestScopeOrg": false
}' | jq "."
```

8 Examine the response and record the ID of your newly created cloud template.

9 Assign the cloud template ID variable.

```
cloud_template_id='<your_cloud_template_id>'
```

your_cloud_template_id is the ID of the new cloud template that you created.

10 To verify that the cloud template has been created, get a list of cloud templates and filter for *your_cloud_template_name*.

```
curl -X GET $url/blueprint/api/blueprints?name=$cloud_template_name -H "Authorization:
Bearer $access_token"  | jq "."
```

11 (Optional) To update the cloud template, use a PUT request and specify *your_cloud_template_id*.

```
curl -X PUT \
  $url/blueprint/api/blueprints/$cloud_template_id?apiVersion=$api_version \
  -H 'Content-Type: application/json' \
  -H "Authorization: Bearer $access_token"  \
  -d '{
    "name" : "'"$cloud_template_name"'",
    "description" : "Basic Cloud Machine cloud template",
    "content" : "formatVersion: 1\ninputs:\n  flavor:\n    type: string\n
title: Flavor\n    description: Flavor Mapping Name\n  image:\n    type:
string\n    title: Image\n    description: Image Mapping Name\n  count:\n
type: integer\n    minimum: 1\n    default: 1\n    maximum: 2\n    title:
Number of Instances\nresources:\n  BasicCloudMachine:\n    type: Cloud.Machine\n
properties:\n      name: BasicCloudMachine\n      flavor: '\''${input.flavor}'\''
\n      image: '\''${input.image}'\''\n      count: '\''${input.count}'\''\n      tags:
[\n        {\n          \"key\": \"env\",\n          \"value\": \"prod\"\n        }
\n      ]\n",
    "projectId" : "'"$project_id"'",
    "requestScopeOrg": false
}' | jq "."
```

Example: Create a Cloud Template and Update it

Create a cloud template named **`MyExampleCloudTemplate`**. This example assumes that **`MyExampleCloudTemplate`** does not already exist.

Assign variables.

```
$ url='https://appliance.domain.com'
$ api_version='2019-09-12'
```

```
$ project_id='394a4ccb-22c6-4ef0-8c75-8b77efbefb51'
$ cloud_template_name='MyExampleCloudTemplate'
```

Validate the cloud template.

```
$ curl -X POST \
  $url/blueprint/api/blueprint-validation?apiVersion=$api_version \
  -H 'Content-Type: application/json' \
  -H "Authorization: Bearer $access_token"  \
  -d '{
    "name" : "'"$cloud_template_name"'",
    "description" : "Basic cloud machine cloud template",
    "content" : "formatVersion: 1\ninputs:\n  flavor:\n    type: string\n
title: Flavor\n    description: Flavor Mapping Name\n  image:\n    type:
string\n    title: Image\n    description: Image Mapping Name\n  count:\n    type:
integer\n    minimum: 1\n    default: 1\n    maximum: 2\n    title: Number of
Instances\nresources:\n  BasicCloudMachine:\n    type: Cloud.Machine\n    properties:\n
name: BasicCloudMachine\n      flavor: '\''${input.flavor}'\''      \n      image: '\''$
{input.image}'\''\n      count: '\''${input.count}'\''",
    "projectId" : "'"$project_id"'",
    "requestScopeOrg": false
}' | jq "."
```

Examine the response to confirm that `"valid":true`.

```
{
  "valid": true,
  "validationMessages": []
}
```

Create the cloud template.

```
$ curl -X POST \
  $url/blueprint/api/blueprints?apiVersion=$api_version \
  -H 'Content-Type: application/json' \
  -H "Authorization: Bearer $access_token"  \
  -d '{
    "name" : "'"$cloud_template_name"'",
    "description" : "Basic cloud machine cloud template",
    "content" : "formatVersion: 1\ninputs:\n  flavor:\n    type: string\n
title: Flavor\n    description: Flavor Mapping Name\n  image:\n    type:
string\n    title: Image\n    description: Image Mapping Name\n  count:\n    type:
integer\n    minimum: 1\n    default: 1\n    maximum: 2\n    title: Number of
Instances\nresources:\n  BasicCloudMachine:\n    type: Cloud.Machine\n    properties:\n
name: BasicCloudMachine\n      flavor: '\''${input.flavor}'\''      \n      image: '\''$
{input.image}'\''\n      count: '\''${input.count}'\''",
    "projectId" : "'"$project_id"'",
    "requestScopeOrg": false
}' | jq "."
```

The response from your request to create the cloud template shows the cloud template ID.

```
{
  "id": "1f170637-81a3-4257-b1cd-b2219ee8034c",
```

```
  "createdAt": "2019-10-10T23:43:27.001Z",
  ...
  "selfLink": "/blueprint/api/blueprints/1f170637-81a3-4257-b1cd-b2219ee8034c"
  ...
}
```

Assign the cloud template ID variable.

```
$ cloud_template_id='1f170637-81a3-4257-b1cd-b2219ee8034c'
```

Update the cloud template to set a tag on the machine properties in the cloud template.

```
$ curl -X PUT \
  $url/blueprint/api/blueprints/$cloud_template_id?apiVersion=$api_version \
  -H 'Content-Type: application/json' \
  -H "Authorization: Bearer $access_token"  \
  -d '{
    "name" : "'"$cloud_template_name"'",
    "description" : "Basic cloud machine cloud template",
    "content" : "formatVersion: 1\ninputs:\n  flavor:\n    type: string\n
title: Flavor\n    description: Flavor Mapping Name\n  image:\n    type:
string\n    title: Image\n    description: Image Mapping Name\n  count:\n    type:
integer\n    minimum: 1\n    default: 1\n    maximum: 2\n    title: Number of
Instances\nresources:\n  BasicCloudMachine:\n    type: Cloud.Machine\n    properties:\n
name: BasicCloudMachine\n      flavor: '\''${input.flavor}'\''      \n      image: '\''$
{input.image}'\''\n      count: '\''${input.count}'\''\n      tags: [\n        {\n
\"key\": \"env\",\n          \"value\": \"prod\"\n        }\n      ]\n",
    "projectId" : "'"$project_id"'",
    "requestScopeOrg": false
}' | jq "."
```

What to do next

Use the cloud template ID to version and release the cloud template to a catalog.

Setting up Policies

As a cloud administrator, you can use the Automation Service Broker APIs to create policies with rules or parameters that you apply to deployments in Automation Service Broker and Automation Assembler.

Create an Approval Policy

To add a layer of governance to deployment requests before they are run, you can create an approval policy using the Policies API. The policy controls who must agree to a deployment or Day 2 action before a request is provisioned.

You create an approval policy based on certain deployment criteria, such as deployments created from a specific cloud template. For example, if you specify a cloud template ID, you can create a policy that requires a specified level of approval for deployments created from that cloud template.

The following procedure shows how to use the Approval API to get the cloud template ID and list approval actions for a deployment before creating the approval policy using the Policy API.

Prerequisites

- Verify that all general prerequisites and prerequisites for the Automation Service Broker Policies service have been satisfied. See Chapter 3 Prerequisites for API Use Case Examples.
- Assign an API version variable for the Approvals API.

```
api_version_approval='2020-11-01'
```

Note The Approvals APIs and Policies APIs have different API version values. You set the API version value for the Policies APIs when you satisfied the general prerequisites.

- Verify that you know the name of the Cloud Template to which you want to apply the approval policy.

Procedure

1 List the cloud templates.

```
curl -X GET \
  $url/approval/api/policy/data/blueprints?apiVersion=$api_version_approval \
  -H "Authorization: Bearer $access_token" | jq "."
```

2 Examine the response to find the ID of the cloud template for the approval policy.

3 Assign the cloud template variable.

```
cloudtemplateId = "<your_cloud_template_ID>"
```

4 (Optional) If you do not know the actions to specify in the policy, list the IDs of deployment actions.

```
curl -X GET \
  $url/approval/api/policy/data/actions?apiVersion=$api_version_approval&search=Deployment
\
  -H "Authorization: Bearer $access_token" | jq "."
```

Note the action IDs.

5 Create an approval policy with hard enforcement that is applied to deployments created from the cloud template with **`cloudtemplateId`**.

- For an approval policy, you specify **`"typeId": "com.vmware.policy.approval"`**
- **`autoApprovalExpiry`** specifies the number of days that the approvers have to act before triggering the **`autoApprovalDecision`**

- **`level`** specifies the order in which the policy is applied with values 1-99. For example, level 1 approvals are applied first, followed by level 2 approvals and so forth.

```
curl -X POST \
  $url/policy/api/policies?apiVersion=$api_version \
  -H "Authorization: Bearer $access_token" \
  -H 'Content-Type: application/json' \
  -d '{
    "name": "<your_approval_policy_name>",
    "enforcementType": "HARD",
    "typeId": "com.vmware.policy.approval"
    "definition": {
      "level": <policy_level>,
      "approverType": "USER",
      "approvalMode": "ALL_OF",
      "autoApprovalDecision": "APPROVE",
      "approvers": [
          "USER:<approver1_ID>",
          "USER:<approver2_ID>"
      ],
      "autoApprovalExpiry": <number_of_days>,
      "actions": [
          "<actionID_1>",
          "<actionID_2",
          "<actionID_3>"
      ]
    },
    "criteria": {
      "matchExpression": [
        {
           "key": "blueprintId",
           "operator": "eq",
           "value": "'$cloudtemplateId'"
        }
      ]
    }
  }' | jq "."
```

Example: Create an approval policy

Create a policy named **`Sample Approval Policy`** to apply to deployments created from a cloud template named **`template-1`**.

Assign variables.

```
$ url='https://appliance.domain.com'
$ api_version='2020-08-25'
$ api_version_approval='2020-11-01'
```

List the cloud templates.

```
curl -X GET \
  $url/approval/api/policy/data/blueprints?apiVersion=$api_version_approval \
  -H "Authorization: Bearer $access_token" | jq "."
```

Examine the response to find the cloud template named **`template-1`**.

```
...
   {
      "id": "77265efc-6d06-428e-9fad-3ad8f31441f3",
      "name": "template-1",
      "description": ""
    }
...
```

Assign the cloud template ID variable.

```
$ cloudtemplateId = "77265efc-6d06-428e-9fad-3ad8f31441f3"
```

List the deployment actions.

```
curl -X GET \
  $url/approval/api/policy/data/actions?apiVersion=$api_version_approval&search=Deployment \
  -H "Authorization: Bearer $access_token" | jq "."
```

Examine the response to find the IDs of the deployment actions that you want to specify in your approval policy.

```
...
   {
      "id": "Deployment.Create",
      "name": "Create",
      "description": "Create a deployment",
      "resourceType": "Deployment"
    },
...
   {
      "id": "Cloud.Azure.Machine.PowerOn",
      "name": "Power On",
      "description": "Power on a machine",
      "resourceType": "Cloud.Azure.Machine"
    },
...
   {
      "id": "Cloud.Azure.Machine.PowerOff",
      "name": "Power Off",
      "description": "Power off a machine",
      "resourceType": "Cloud.Azure.Machine"
    },
...
```

Use the cloud template ID to create the approval policy of level 2 with hard enforcement named **`Sample Approval Policy`**. When a deployment is requested, users listed will act as approvers for the actions: provision, power on, and power off an Azure machine. If approvers do not act within three days, then the deployment actions are automatically approved.

```
$ curl -X POST \
  $url/policy/api/policies?apiVersion=$api_version \
  -H "Authorization: Bearer $access_token" \
  -H 'Content-Type: application/json' \
  -d '{
    "name": "Sample Approval Policy",
    "enforcementType": "HARD",
    "typeId": "com.vmware.policy.approval"
    "definition": {
      "level": 2,
      "approverType": "USER",
      "approvalMode": "ALL_OF",
      "autoApprovalDecision": "APPROVE",
      "approvers": [
          "USER:mary@mycompany.com",
          "USER:susan@mycompany.com"
       ],
      "autoApprovalExpiry": 3,
      "actions": [
          "Deployment.Create",
          "Cloud.Azure.Machine.PowerOn",
          "Cloud.Azure.Machine.PowerOff"
       ]
    },
    "criteria": {
      "matchExpression": [
        {
          "key": "blueprintId",
          "operator": "eq",
          "value": "'$cloudtemplateId'"
        }
      ]
    }
  }' | jq "."
```

The response shows the approval policy.

```
{
  "id": "62ad2f02-0b2a-4ed8-a739-a6c40d761e49",
  "name": "Sample Approval Policy",
  "typeId": "com.vmware.policy.approval",
  "enforcementType": "HARD",
  "orgId": "d2994f92-bd52-45b1-9220-686b20944c2c",
  "definition": {
    "level": 2,
    "approverType": "USER",
    "approvalMode": "ALL_OF",
    "autoApprovalDecision": "APPROVE",
    "approvers": [
```

```
            "USER:mary@mycompany.com",
            "USER:susan@mycompany.com"
        ],
      "autoApprovalExpiry": 3,
      "actions": [
            "Deployment.Create",
            "Cloud.Azure.Machine.PowerOn",
            "Cloud.Azure.Machine.PowerOff"
      ]
    },
    "criteria": {
      "matchExpression": [
        {
          "key": "blueprintId",
          "operator": "eq",
          "value": "77265efc-6d06-428e-9fad-3ad8f31441f3"
        }
      ]
    },
    "createdAt": "2021-11-08T09:45:38.108885Z",
    "createdBy": "admin@mycompany.com",
    "lastUpdatedAt": "2021-11-08T09:45:38.108885Z",
    "lastUpdatedBy": "admin@mycompany.com"
}
```


How to Create Resource Quota Policies

To limit the resource consumed by each user, project, or organization, you can create a resource quota policy using the Policies API. Applying the policy ensures that users do not consume all available resources.

With the policy defined, any new deployment requests are evaluated against the resource quota. If a resource request exceeds the quota, the deployment request fails with a message that indicates the reason for the failure. The resource quota does not apply to deployment requests that are already in-progress when the policy is defined.

In cases where multiple policies are defined for a resource at the same scope level, the request is evaluated against the quota with the lowest limit value.

For information about policy scope, see How do I configure scope in Automation Service Broker policies.

The following examples show how to define a resource quota policy with:

- Organization level limits.
- Project level limits.

For both definitions, you use `POST /policy/api/policies` to create the policy with the input parameter **`"typeId": "com.vmware.policy.resource.quota"`**.

Prerequisites for creating a resource quota policy

- Verify that all general prerequisites and prerequisites for the Automation Service Broker Policies service have been satisfied. See Chapter 3 Prerequisites for API Use Case Examples.
- Verify that you know the resource quotas that you want to apply.

Resource quota policy with organizational level limits

This example shows how to define a resource quota policy with hard enforcement and limits on resource consumption by the organization and each user within the organization.

```
curl -k -X POST \
  "$url/policy/api/policies?apiVersion=$api_version" \
  -H "Authorization: Bearer $access_token" \
  -H 'Content-Type: application/json' \
  -d '{
    "name": "Sample Resource Quota Policy 1",
    "enforcementType": "HARD",
    "typeId": "com.vmware.policy.resource.quota",
    "definition": {
      "orgLevel": {
          "limits": {
              "cpu": {
                  "value": 50
              },
              "storage": {
                  "value": 150,
                  "unit": "GB"
              },
              "memory": {
                  "value": 150,
                  "unit": "GB"
              },
              "instances": {
                  "value": 50
              }
          },
          "userLevel": {
              "limits": {
                  "cpu": {
                      "value": 4
                  },
                  "storage": {
                      "value": 10,
                      "unit": "GB"
                  },
                  "memory": {
                      "value": 10,
                      "unit": "GB"
                  },
                  "instances": {
                      "value": 2
                  }
              }
```

```
          }
       }
     }
   }' | jq "."
```

The response shows the resource quota policy with organization and organization user level limits.

```
{
    "id": "52f67297-3e15-4e0a-9336-35894d2be0bc",
    "name": "Sample Resource Quota Policy 1",
    "typeId": "com.vmware.policy.resource.quota",
    "enforcementType": "HARD",
    "orgId": "74a191fc-27e0-4f09-af0d-6acd04d60832",
    "definition": {
      "orgLevel": {
          "limits": {
              "cpu": {
                  "value": 50
              },
              "storage": {
                  "value": 150,
                  "unit": "GB"
              },
              "memory": {
                  "value": 150,
                  "unit": "GB"
              },
              "instances": {
                  "value": 50
              }
          },
          "userLevel": {
              "limits": {
                  "cpu": {
                      "value": 4
                  },
                  "storage": {
                      "value": 10,
                      "unit": "GB"
                  },
                  "memory": {
                      "value": 10,
                      "unit": "GB"
                  },
                  "instances": {
                      "value": 2
                  }
              }
          }
      }
    },
    "createdAt": "2021-11-08T11:51:47.742311Z",
    "createdBy": "admin@mycompany.com",
    "lastUpdatedAt": "2021-11-08T11:51:47.742311Z",
```

```
    "lastUpdatedBy": "admin@mycompany.com"
}
```

Resource quota policy with project level limits

This example shows how to define a resource quota policy with hard enforcement and limits on resource consumption by a project and each project user.

```
curl -k -X POST \
  "$url/policy/api/policies?apiVersion=$api_version" \
  -H "Authorization: Bearer $access_token" \
  -H 'Content-Type: application/json' \
  -d '{
    "name": "Sample Resource Quota Policy 2",
    "enforcementType": "HARD",
    "projectId": "6df55bb1-7444-4c13-9997-4004ba0a321d",
    "scopeCriteria": null,
    "typeId": "com.vmware.policy.resource.quota",
    "definition": {
      "projectLevel": {
          "limits": {
              "cpu": {
                  "value": 50
              },
              "storage": {
                  "value": 80,
                  "unit": "GB"
              },
              "memory": {
                  "value": 80,
                  "unit": "GB"
              },
              "instances": {
                  "value": 40
              }
          },
          "userLevel": {
              "limits": {
                  "cpu": {
                      "value": 5
                  },
                  "storage": {
                      "value": 8,
                      "unit": "GB"
                  },
                  "memory": {
                      "value": 8,
                      "unit": "GB"
                  },
                  "instances": {
                      "value": 4
                  }
              }
          }
```

```
      }
    }
  }' | jq "."
```

The response shows the resource quota policy with project and project user level limits.

```
{
    "id": "5809ec88-16fb-4553-98c3-5b588ebff322",
    "name": "Sample Resource Quota Policy 2",
    "typeId": "com.vmware.policy.resource.quota",
    "enforcementType": "HARD",
    "orgId": "74a191fc-27e0-4f09-af0d-6acd04d60832",
    "projectId": "6df55bb1-7444-4c13-9997-4004ba0a321d",
    "definition": {
      "projectLevel": {
          "limits": {
              "cpu": {
                  "value": 50
              },
              "storage": {
                  "value": 80,
                  "unit": "GB"
              },
              "memory": {
                  "value": 80,
                  "unit": "GB"
              },
              "instances": {
                  "value": 40
              }
          },
          "userLevel": {
              "limits": {
                  "cpu": {
                      "value": 5
                  },
                  "storage": {
                      "value": 8,
                      "unit": "GB"
                  },
                  "memory": {
                      "value": 8,
                      "unit": "GB"
                  },
                  "instances": {
                      "value": 4
                  }
              }
          }
      }
    },
    "createdAt": "2021-11-08T12:09:34.160569Z",
    "createdBy": "admin@mycompany.com",
    "lastUpdatedAt": "2021-11-08T12:09:34.160569Z",
```

```
    "lastUpdatedBy": "admin@mycompany.com"
}
```

Create a Deployment Limit Policy

To limit resource consumption when users deploy cloud templates in Automation Assembler and request catalog items in Automation Service Broker, you can create a limit policy using the Policies API. The policy applies limits to all deployments in an organization by default.

You create a deployment limit policy based on certain resource criteria, such as account names, account types, cloud templates, cloud zones, flavors, and many more. For example, if you specify a cloud template ID as the resource criteria, you can restrict the policy so that it only applies limits to deployments created from a specific cloud template.

Note If approval policy or resource quota policy definitions affect deployments within the policy scope, deployment limits are enforced before the other policy types.

The following procedure shows how to use the Deployment limit API to get the cloud template ID before creating the deployment limit policy using the Policy API.

Prerequisites

- Verify that all general prerequisites and prerequisites for the Automation Service Broker Policies service have been satisfied. See Chapter 3 Prerequisites for API Use Case Examples.
- Assign an API version variable for the Deployment Limit API.

  ```
  api_version_deploymentlimit='2020-11-01'
  ```

  Note The Deployment Limit APIs and Policies APIs have different API version values. You set the API version value for the Policies APIs when you satisfied the general prerequisites. The Deployment Limit APIs are grouped with the Approvals services. See API Services.
- Verify you know the resource criteria that you want to use to restrict the policy.

Procedure

1. List the cloud templates.

   ```
   curl -X GET \
     $url/deploymentlimit/api/policy/data/blueprints?apiVersion=$api_version_deploymentlimit \
     -H "Authorization: Bearer $access_token" | jq "."
   ```

2. Examine the response to find the ID of the cloud template used to create the deployments where you want to limit resource usage.
3. Assign the cloud template variable.

   ```
   cloudtemplateId = "<your_cloud_template_ID>"
   ```

4 Create a deployment limit policy with hard enforcement that is applied to deployments created from the cloud template with **`cloudtemplateId`**. For the deployment limit policy, you specify **`"typeId": "com.vmware.policy.deployment.limit"`**.

```
curl -X POST \
  $url/policy/api/policies?apiVersion=$api_version \
  -H "Authorization: Bearer $access_token" \
  -H 'Content-Type: application/json' \
  -d '{
    "name": "<your_limit_policy_name>",
    "enforcementType": "HARD",
    "typeId": "com.vmware.policy.deployment.limit"
    "definition": {
      "deploymentLimits": {
        "cpu": {
          "value": 6
        },
        "memory": {
          "unit": "GB",
          "value": 5
        },
        "instances": {
          "value": 3
        },
        "storage": {
          "unit": "GB",
          "value": "20
        }
      },
      "deploymentResourceLimits": {
        "resources": [
          {
            "name": "vSphere-Machine-Limits",
            "limits": {
              "cpu": {
                "value": 2
              },
              "memory": {
                "unit": "GB",
                "value": 2
              },
              "storage": {
                "unit": "GB",
                "value": "20
              }
            },
            "criteria": {
              "matchExpression": [
                {
                  "key": "type",
                  "value": "Cloud.vSphere.Machine",
                  "operator": "eq"
                }
              ]
```

```
            }
          }
        ]
      },
      "criteria": {
        "matchExpression": [
          {
            "key": "blueprintId",
            "operator": "eq",
            "value": "'$cloudtemplateId'"
          }
        ]
      }
    }
  }
| jq "."
```

Example: Create a deployment limit policy

Create a deployment limit policy named **`Sample Limit Policy`** that is applied to limit resource usage in deployments created from a cloud template named **`template2`**.

Assign variables.

```
$ url='https://appliance.domain.com'
$ api_version='2020-08-25'
$ api_version_deploymentlimit='2020-11-01'
$ orgId='394a4ccb-22c6-4ef0-8c75-8b77efbefb51'
```

List the cloud templates.

```
$ curl -X GET \
  $url/deploymentlimit/api/policy/data/blueprints?apiVersion=$api_version_deploymentlimit \
  -H "Authorization: Bearer $access_token" | jq "."
```

Examine the response to find the cloud template named **`template2`**.

```
...
 {
      "id": "3d3c714f-0aeb-423d-a494-97e85e4a8566",
      "name": "template2",
      "description": "Cloud template for example deployment"
    },
...
```

Assign the cloud template ID variable.

```
$ cloudtemplateId = "3d3c714f-0aeb-423d-a494-97e85e4a8566"
```

Use the cloud template ID to create the deployment limit policy with hard enforcement named **`Sample Limit Policy`**. The value for the type ID is fixed as **`com.vmware.policy.deployment.limit`**.

```
$ curl -X POST \
  $url/policy/api/policies?apiVersion=$api_version \
  -H "Authorization: Bearer $access_token" \
  -H 'Content-Type: application/json' \
  -d '{
    "name": "Sample Limit Policy",
    "enforcementType": "HARD",
    "typeId": "com.vmware.policy.deployment.limit"
    "definition": {
      "deploymentLimits": {
        "cpu": {
          "value": 6
        },
        "memory": {
          "unit": "GB",
          "value": 5
        },
        "instances": {
          "value": 3
        },
        "storage": {
          "unit": "GB",
          "value": "20
        },
      },
      "deploymentResourceLimits": {
        "resources": [
          {
            "name": "vSphere-Machine-Limits",
            "limits": {
              "cpu": {
                "value": 2
              },
              "memory": {
                "unit": "GB",
                "value": 2
              },
              "storage": {
                "unit": "GB",
                "value": "20
              }
            },
            "criteria": {
              "matchExpression": [
                {
                  "key": "type",
                  "value": "Cloud.vSphere.Machine",
                  "operator": "eq"
                }
              ]
```

```
            }
          }
        ]
      },
      "criteria": {
        "matchExpression": [
          {
            "key": "blueprintId",
            "operator": "eq",
            "value": "'$cloudtemplateId'"
          }
        ]
      }
    }
  }
| jq "."
```

The response shows the deployment limit policy.

```
{
  "id": "62ad2f02-0b2a-4ed8-a739-a6c40d761e49",
  "name": "Sample Limit Policy",
  "typeId": "com.vmware.policy.deployment.limit",
  "enforcementType": "HARD",
  "orgId": "d2994f92-bd52-45b1-9220-686b20944c2c",
  "definition": {
    "deploymentLimits": {
      "cpu": {
        "value": 6
      },
      "memory": {
        "unit": "GB",
        "value": 5
      },
      "instances": {
        "value": 3
      },
      "storage": {
        "unit": "GB",
        "value": 20
      }
    },
    "deploymentResourceLimits": {
      "resources": [
        {
          "name": "vSphere-Machine-Limits",
          "limits": {
            "cpu": {
              "value": 2
            },
            "memory": {
              "unit": "GB",
              "value": 2
            },
            "storage": {
```

```
              "unit": "GB",
              "value": "20
            }
          },
          "criteria": {
            "matchExpression": [
              {
                "key": "type",
                "value": "Cloud.vSphere.Machine",
                "operator": "eq"
              }
            ]
          }
        }
      ]
    }
  },
  "criteria": {
    "matchExpression": [
      {
        "key": "blueprintId",
        "operator": "eq",
        "value": "7950795a-4f66-451c-a79f-be9ef6bd723c"
      }
    ]
  },
  "createdAt": "2021-11-088T09:45:38.108885Z",
  "createdBy": "user@mycompany.com",
  "lastUpdatedAt": "2021-11-08T09:45:38.108885Z",
  "lastUpdatedBy": "user@mycompany.com"
}
```

Create a Content Sharing Policy

As a Automation Service Broker administrator, you can create a content sharing policy that entitles all Automation Service Broker users in a project to shared content defined in the policy.

The following procedure shows how to use the Policies API to create a content sharing policy that defines shared content for users or groups that are part of the same project. To get the input required for the project, project users, and project groups, you use the Projects API. To get the catalog sources or catalog items to share in the policy, you use the Catalog API.

The steps to get catalog sources or items to share are optional, because a content sharing policy may not share both sources and items. However, the policy must share at least one catalog source or item.

Prerequisites

- Verify that all general prerequisites and prerequisites for the Automation Service Broker Policies service have been satisfied. See Chapter 3 Prerequisites for API Use Case Examples.

  Note The Policies API and the Catalog API have the same API version. You set the API version value for both services when you satisfy the prerequisites for the Policies service.

- Assign an API version variable for the Projects API.

```
api_version_projects='2019-01-15'
```

Note The Projects APIs and Policies APIs have different API version values. You set the API version value for the Policies APIs when you satisfied the prerequisites for the Policies service.

Procedure

1 Get a list of all projects in your organization.

```
curl -X GET "$url/project-service/api/projects?apiVersion=$api_version_projects" -H
"Authorization: Bearer $access_token"  | jq "."
```

Examine the response to get the ID of the project for the shared policy. The project must include the users or groups for which you want to entitle content. See Create a Project with the Project Service API.

2 Assign the project ID variable.

```
project_id='<your_project_id>'
```

3 Get a list of users in the project.

```
curl -X GET \
$url/project-service/api/projects/$project_id/principals?
apiVersion=$api_version_projects&expandGroups=true& \
'$filter='"(substringof('{}',tolower(acct)))" \
-H "Authorization: Bearer $access_token"  | jq "."
```

Examine the response to get the user email addresses for the project users. Users selected in the policy are entitled to items shared with the project.

4 Get a list of groups in the project.

```
curl -X GET \
$url/project-service/api/projects/$project_id/groups?
apiVersion=$api_version_projects&page=0 \
'$filter='"(substringof('{}',tolower(displayName)))" \
-H "Authorization: Bearer $access_token"  | jq "."
```

Examine the response to get the group display name for the project users that are part of the group. Groups selected in the policy are entitled to items shared with the project.

5 (Optional) Get a list of catalog sources for your project.

```
curl -X GET $url/catalog/api/admin/sources?projectId=$project_id&apiVersion=$api_version
-H "Authorization: Bearer $access_token"  | jq "."
```

Examine the response to find catalog sources that you want to share.

6 (Optional) If you are sharing a catalog source, assign the catalog source ID variable.

```
catalog_source_id='<shared_catalog_source_id>'
```

7 (Optional) Get a list of catalog items for your project.

```
curl -X GET $url/catalog/api/items?projectId=$project_id&apiVersion=$api_version -H
"Authorization: Bearer $access_token"  | jq "."
```

Examine the response to find catalog items that you want to share.

8 (Optional) If you are sharing a catalog item, the catalog item ID variable.

```
catalog_item_id='<shared_catalog_item_id>'
```

9 Create a content sharing policy with hard enforcement.

- The value for user type is always **`USER`**.
- The group's display name is always in the format **`groupDisplayName@groupDomain`**.
- The typeId for the content sharing policy is always **`com.vmware.policy.catalog.entitlement`**.

```
curl -X POST \
  $url/policy/api/policies?apiVersion=$api_version \
  -H "Authorization: Bearer $access_token" \
  -H 'Content-Type: application/json' \
  -d '{
    "name": "<your_content_sharing_policy_name>",
    "projectId": "'$project_id'",
    "definition": {
        "entitledUsers": [
            {
                "userType": "USER",
                "principals": [
                    {
                        "type": "USER",
                        "referenceId": "<user1_email_address>"
                    },
                    {
                        "type": "USER",
                        "referenceId": "<user2_email_address>"
                    },
                    {
                        "type": "PROJECT",
                        "referenceId": "$project_id'"
                    },
                    {
                        "type": "GROUP",
                        "referenceId": "<groupDisplayName1@groupDomain>"
                    }
                ],
                "items": [
                    {
```

```
                        "id": "'$catalog_source_id'",
                        "type": "CATALOG_SOURCE_IDENTIFIER"
                    },
                    {
                        "id": "'$catalog_item_id'",
                        "type": "CATALOG_ITEM_IDENTIFIER"
                    }
                ]
            }
        ]
    },
    "enforcementType": "HARD",
    "typeId": "com.vmware.policy.catalog.entitlement"
  }
}' | jq "."
```

Example: Create a content sharing policy

Create a policy named **`Sample Sharing Policy`**.

Assign variables.

```
$ url='https://appliance.domain.com'
$ api_version='2020-08-25'
$ api_version_projects='2019-01-15'
```

List the projects in your organization.

```
$ curl -X GET "$url/project-service/api/projects?apiVersion=$api_version_projects" -H
"Authorization: Bearer $access_token"  | jq "."
```

Examine the response to find the project that you want to use for the shared content policy.

```
{
      "id": "1d0bcd42-4d8f-4a8f-8b31-f34a4707533e",
      "name": "Example-project",
      "description": "This is an example project",
      "orgId": "f670fdfc-66d6-4689-9793-d524e7066d1e",
...
```

Assign the project ID.

```
$ project_id='1d0bcd42-4d8f-4a8f-8b31-f34a4707533e'
```

List the users in the project.

```
$ curl -X GET \
 $url/project-service/api/projects/$project_id/principals?
apiVersion=$api_version_projects&expandGroups=true& \
 '$filter='"(substringof('{}',tolower(acct)))" \
 -H "Authorization: Bearer $access_token"  | jq "."
```

A snippet of the response shows the email addresses for the users in the project.

```
...
    "content": [
        {
            "id": "ab373898-d29b-4e3b-8703-58023cadd140",
            "acct": "user1@mycompany.com",
            "domain": "mycompany.com"
        }
    ],
...
```

Get a list of groups in the project.

```
$ curl -X GET \
$url/project-service/api/projects/$project_id/groups?apiVersion=$api_version_projects&page=0 \
'$filter='"(substringof('{}',tolower(displayName)))" \
-H "Authorization: Bearer $access_token"  | jq "."
```

A snippet of the response shows the group display name for the users that are part of the group.

```
...
  "content": [
        {
            "id": "a9da96e7-ba19-47d4-9f38-dd1983e29424",
            "displayName": "test@mycompany.com",
            "groupType": "USER_GROUP",
            "usersCount": 2
        },
...
```

Get a list of catalog sources for your project.

```
$ curl -X GET $url/catalog/api/admin/sources?projectId=$project_id&apiVersion=$api_version -H
"Authorization: Bearer $access_token"  | jq "."
```

Examine the response to find the catalog sources that you want to share.

```
...
    "content": [
        {
            "id": "600026c6-3155-4395-a990-580ff1159e82",
            "name": "BpContent-Quality Engineering",
            "description": "For Project-Quality Engineering",
            "typeId": "com.vmw.blueprint",
            "createdAt": "2022-10-12T10:37:01.751799Z",
            "createdBy": "admin@mycompany.com",
            "lastUpdatedAt": "2022-10-17T05:06:33.976796Z",
            "lastUpdatedBy": "system-user",
            "config": {
                "sourceProjectId": "1d0bcd42-4d8f-4a8f-8b31-f34a4707533e"
            },
      ...
```

Assign the ID of the catalog source to share.

```
$ catalog_source_id='600026c6-3155-4395-a990-580ff1159e82'
```

Get a list of catalog items for your project.

```
$ curl -X GET $url/catalog/api/admin/items?projectId=$project_id&apiVersion=$api_version -H
"Authorization: Bearer $access_token"  | jq "."
```

Examine the response to find the catalog item.

```
...
     "content": [
        {
            "id": "b2d0fba7-5f62-3c79-b1b8-a2aa7d38063b",
            "name": "CF-SQAVC67-Centos-MultiMachine",
            "description": "CF-SQAVC67-Centos-MultiMachine vsphere components, disk,
networks",
            "sourceId": "600026c6-3155-4395-a990-580ff1159e82",
            "sourceName": "BpContent-Quality Engineering",
...
```

Assign the ID of the catalog item to share.

```
$ catalog_item_id='b2d0fba7-5f62-3c79-b1b8-a2aa7d38063b'
```

Create a content sharing policy with hard enforcement.

The following example shows the group's display name which is of the format **`groupDisplayName@groupDomain`** where:

- `groupDisplayName` is **`test@mycompany.com`**
- `groupDomain` is **`mycompany.com`**

```
$ curl -X POST \
  $url/policy/api/policies?apiVersion=$api_version \
  -H "Authorization: Bearer $access_token" \
  -H 'Content-Type: application/json' \
  -d '{
    "name": "Sample Sharing Policy",
    "projectId": "'$project_id'",
    "definition": {
        "entitledUsers": [
            {
                "userType": "USER",
                "principals": [
                    {
                        "type": "USER",
                        "referenceId": "user1@mycompany.com"
                    },
                    {
                        "type": "PROJECT",
                        "referenceId": "'$project_id'"
```

```
                    },
                    {
                        "type": "GROUP",
                        "referenceId": "test@mycompany.com@mycompany.com"
                    }
                ],
                "items": [
                    {
                        "id": "'$catalog_source_id'",
                        "type": "CATALOG_SOURCE_IDENTIFIER"
                    },
                    {
                        "id": "'$catalog_item_id'",
                        "type": "CATALOG_ITEM_IDENTIFIER"
                    }
                ]
            }
        ]
    },
    "enforcementType": "HARD",
    "typeId": "com.vmware.policy.catalog.entitlement"
}
```

The response shows the content sharing policy.

```
{
    "id": "0cecca9a-d778-47b5-acdf-c08248406052",
    "name": "Sample Sharing Policy",
    "projectId": "1d0bcd42-4d8f-4a8f-8b31-f34a4707533e",
    "definition": {
        "entitledUsers": [
            {
                "userType": "USER",
                "principals": [
                    {
                        "type": "USER",
                        "referenceId": "user1@mycompany.com"
                    },
                    {
                        "type": "PROJECT",
                        "referenceId": "1d0bcd42-4d8f-4a8f-8b31-f34a4707533e"
                    },
                    {
                        "type": "GROUP",
                        "referenceId": "test@mycompany.com@mycompany.com"
                    }
                ],
                "items": [
                    {
                        "id": "600026c6-3155-4395-a990-580ff1159e82",
                        "type": "CATALOG_SOURCE_IDENTIFIER"
                    },
                    {
                        "id": "b2d0fba7-5f62-3c79-b1b8-a2aa7d38063b",
                        "type": "CATALOG_ITEM_IDENTIFIER"
```

```
                    }
                ]
            }
        ]
    },
    "enforcementType": "HARD",
    "typeId": "com.vmware.policy.catalog.entitlement",
    "orgId": "10ea6be1-7723-4bf0-a221-8b4f3c7a26f7",
    "createdAt": "2022-10-24T07:52:22.731448Z",
    "createdBy": "admin@mycompany.com",
    "lastUpdatedAt": "2022-10-24T07:52:22.731448Z",
    "lastUpdatedBy": "admin@mycompany.com"
}
```

Version and Release a Cloud Template to a VMware Aria Automation Service Broker Catalog

After creating a cloud template, you version and release your cloud template using a POST request. The request body includes the ID of an existing cloud template and the number of the version to release.

By versioning and releasing the cloud template, you mark a cloud template version as ready to be consumed by VMware Aria Automation Service Broker. To show the released cloud template in VMware Aria Automation Service Broker, you must have a catalog source.

Prerequisites

- Verify that all general prerequisites and prerequisites for the Automation Assembler Blueprint service have been satisfied. See Chapter 3 Prerequisites for API Use Case Examples.
- Verify that you have the cloud template ID for the cloud template you want to version and release. See Create and Update a Cloud Template.
- Verify that you know the version of the cloud template that you want to create and release to the catalog.

Procedure

1 Assign the cloud template ID variable.

```
cloud_template_id='<your_cloud_template_id>'
```

2 Assign a cloud template version variable.

```
cloud_template_version='<your_cloud_template_version>'
```

your_cloud_template_version is the version that you want to create.

3 To version the cloud template without releasing it, submit the request with `"release": false`.

```
curl -X POST \
  $url/blueprint/api/blueprints/$cloud_template_id/versions?apiVersion=$api_version \
  -H "Authorization: Bearer $access_token" \
  -H 'Content-Type: application/json' \
  -d '{
  "changeLog": "Creating a version '"$cloud_template_version"'",
  "description": "Creating a version from the current draft",
  "release": false,
  "version": "'"$cloud_template_version"'"
}' | jq "."
```

4 Release the cloud template.

```
curl -X POST \
  $url/blueprint/api/blueprints/$cloud_template_id/versions/$cloud_template_version/
actions/release?apiVersion=$api_version" \
  -H "Authorization: Bearer $access_token" \
  -H 'Content-Type: application/json' | jq "."
```

Example: Version and Release a Cloud Template

Release version 5 of your cloud template with ID **`1f170637-81a3-4257-b1cd-b2219ee8034c`**.

Assign variables.

```
$ url='https://appliance.domain.com'
$ api_version='2019-09-12'
$ cloud_template_id='1f170637-81a3-4257-b1cd-b2219ee8034c'
$ cloud_template_version='v5'
```

Version the cloud template without releasing it.

```
$ curl -X POST \
  $url/blueprint/api/blueprints/$cloud_template_id/versions?apiVersion=$api_version \
  -H "Authorization: Bearer $access_token" \
  -H 'Content-Type: application/json' \
  -d '{
  "changeLog": "Creating a version '"$cloud_template_version"'",
  "description": "Creating a version from the current draft",
  "release": false,
  "version": "'"$cloud_template_version"'"
}' | jq "."
```

Release the cloud template.

```
$ curl -X POST \
  $url/blueprint/api/blueprints/$cloud_template_id/versions/$cloud_template_version/actions/
release?apiVersion=$api_version \
  -H "Authorization: Bearer $access_token" \
  -H 'Content-Type: application/json' | jq "."
```

A snippet of the response shows the new cloud template version with a `RELEASED` status.

```
...
  "blueprintId": "1f170637-81a3-4257-b1cd-b2219ee8034c",
  "name": "MyExampleCloudTemplate",
  "description": "Basic Cloud Machine cloud template",
  "version": "v5",
  "tags": [],
  "content": "formatVersion: 1\ninputs:\n  flavor:\n    type: string\n    title: Flavor\n
description: Flavor Mapping Name\n  image:\n    type: string\n    title: Image\n
description: Image Mapping Name\n  count:\n    type: integer\n    minimum: 1\n    default:
1\n    maximum: 2\n    title: Number of Instances\nresources:\n  BasicCloudMachine:\n
type: Cloud.Machine\n    properties:\n      name: BasicCloudMachine\n      flavor: '$
{input.flavor}'      \n      image: '${input.image}'\n      count: '${input.count}'\n
tags: [\n        {\n          \"key\": \"env\",\n          \"value\": \"prod\"\n        }
\n      ]\n",
  "status": "RELEASED",
  "versionDescription": "Creating a version from the current draft",
  "versionChangeLog": "Creating a version v5",
  "valid": true
}
```

Edit and Version a Custom Form in your Cloud Template

After creating a cloud template in Automation Assembler, versioning it, and importing it into your Automation Service Broker content, you can add a custom request form. Then you can use the form designer to edit and version the form, so that you can choose the version and use it later.

The following procedure assumes that your Automation Service Broker content includes a versioned Automation Assembler cloud template. Using the Catalog service API, you find the ID of the cloud template and get the cloud template schema for one of the template versions. To create a custom form from the schema, you use the Form Service API. Additional steps show how to customize and version the custom form, and how to list and restore a versioned form.

For more information about Custom Forms, see Learn more about Automation Service Broker custom forms.

Prerequisites

- Verify that you have satisfied all general prerequisites and prerequisites for the Automation Service Broker Catalog service. See Chapter 3 Prerequisites for API Use Case Examples.
- Verify that you have versioned and released your cloud template in Automation Assembler. See Version and Release a Cloud Template to a VMware Aria Automation Service Broker Catalog.
- Verify that you have imported your cloud template into Automation Service Broker. See Add Automation Assembler templates to the Automation Service Broker catalog.

Procedure

1 List the items in your catalog.

```
curl -X GET -H 'Content-Type: application/json' -H "Authorization: Bearer $access_token"
"$url/catalog/api/admin/items?apiVersion=$api_version"  | jq "."
```

Examine the response to find the catalog ID of the cloud template that you want to use.

2 Assign a variable for the cloud template catalog ID.

```
item_id='<cloud_template_catalog_ID>'
```

3 Get information about the catalog item.

```
curl -X GET -H 'Content-Type: application/json' -H "Authorization: Bearer $access_token"
"$url/catalog/api/admin/items/$item_id?apiVersion=$api_version"  | jq "."
```

Examine the result to choose the version of the template that you want to use for the custom form.

4 Assign a variable for the cloud template version ID.

The form that you create and customize a form will be associated with this cloud template version ID.

```
version_id='<your_cloud_template_version_ID>'
```

5 Using the catalog ID and version ID, get the schema for the the cloud template.

```
$ schema='curl -k -X GET \
"$url/catalog/api/admin/items/$item_id/versions/$version_id \
-H "Authorization: Bearer $access_token" \
-H 'Content-Type: application/json' | jq -r .schema'
```

6 Use the schema to create a custom form.

```
curl -k -X POST \
   "$url/form-service/api/forms/designer/request?
sourceType=com.vmw.blueprint.version&sourceId=$item_id/$version_id&formType=requestForm" \
   -H "Authorization: Bearer $access_token" \
   -H 'Content-Type: application/json' \
   -d '$schema' | jq "."
```

You have a current version of your custom form.

7 Customize the form.

- *your_modified_form_json_as_string* Used to update the form field properties. For information about form field properties, see Custom form designer field properties in Automation Service Broker.

- *your_modified_styles_string* Used to update a cascading style sheet written in CSS Syntax Module Level 3. For information about the CSS file format, see Learn more about Automation Service Broker custom forms.

```
curl -k -X POST \
-H "Authorization: Bearer $access_token" \
-H 'Content-Type: application/json' \
"$url/form-service/api/forms" \
-d '{
   "name":"Demo Blueprint / 2.0.0",
   "form":"<your_modified_form_json_as_string>",
   "styles":"<your_modified_styles_string>",
   "status":"ON", // set ON to enable the custom form, OFF to disable
   "type":"requestForm",
   "sourceId":"$item_id/$version_id",
   "sourceType":"com.vmw.blueprint.version"
 }' | jq "."
```

A response with the status `201 'Created'` indicates a successful form update.

8 To save the current form with a new version name, you version the form.

```
curl -k -X POST \
"$url/form-service/api/forms/versions" \
-H "Authorization: Bearer $token" \
-H 'Content-Type: application/json' \
-d '{
   "name":"<your_form_version_name>",
   "sourceId":"$item_id/$version_id",
   "sourceType":"com.vmw.blueprint.version",
   "formType":"requestForm"
}' | jq "."
```

The form is saved with version name.

To create more versions, you repeat the steps to customize and version the form. If you customize without versioning, the changes you make are only applied to the current form.

9 (Optional) If you have at least one form version in addition to the current form, you can list and restore the version that you want to use.

- To list form versions:

```
curl -k -X GET \
"$url/form-service/api/forms/versions?
sourceType=com.vmw.blueprint.version&sourceId=$item_id/
$version_id&formType=requestForm" \
-H "Authorization: Bearer $token" \
-H 'Content-Type: application/json' | jq "."
```

The response lists the names of stored form versions and their IDs or **`formVersionId`**.

- To restore a previously versioned form, use the **formVersionId**:

```
curl -k -X PATCH \
"$url/form-service/api/forms/versions/$formVersionId/restore" \
-H "Authorization: Bearer $token" \
-H 'Content-Type: application/json' | jq "."
```

Example: Edit and Version a Custom Form

Edit the custom form for a cloud template with the name **Demo Blueprint**. Then version the custom form it so that you can save it to use later.

Assign variables.

```
$ url='https://appliance.domain.com'
$ api_version='2020-08-25'
```

List the items in your catalog.

```
$ curl -X GET -H 'Content-Type: application/json' -H "Authorization: Bearer $access_token"
"$url/catalog/api/admin/items?apiVersion=$api_version" | jq "."
```

Examine the response to find the cloud template named **Demo Blueprint** and get the catalog ID.

```
[
  {
    "id": "890275e3-cbc6-3778-af0d-d9eddf4664a0",
    "name": "Demo Blueprint",
    "description": "",
    "sourceId": "d4c24bba-68cc-4923-9155-338c4bf3e663",
    "sourceName": "template-content-source",
    "type": {
      "id": "com.vmw.blueprint",
      "link": "/catalog/api/types/com.vmw.blueprint",
      "name": "VMware Aria Automation Templates"
    },
    "createdAt": "2023-05-03T15:15:02.805617Z",
    "createdBy": "user@mycompany.com",
    "lastUpdatedAt": "2023-05-03T15:15:02.805617Z",
    "iconId": "1495b8d9-9428-30d6-9626-10ff9281645e",
    "bulkRequestLimit": 1
  }
]
```

Assign the item ID variable with the catalog ID of the cloud template.

```
$ item_id='890275e3-cbc6-3778-af0d-d9eddf4664a0'
```

Get catalog information about the cloud template.

```
$ curl -X GET \
   "$url/catalog/api/admin/items/$item_id?apiVersion=$api_version" \
```

```
    -H 'Content-Type: application/json' \
    -H "Authorization: Bearer $access_token"  | jq .content
```

A snippet of the response lists the imported versions.

```
[
  {
    "id": "2.0.0",
    "description": "Added numberInput field",
    "createdAt": "2023-03-29T06:19:48.748349Z",
    "externalId": "/blueprint/api/blueprints/bbbb30b8-a73a-44bb-92db-27a504e0ec85"
  },
  {
    "id": "1.0.0",
    "description": "Added stringInput field",
    "createdAt": "2023-03-29T06:19:30.845516Z",
    "externalId": "/blueprint/api/blueprints/bbbb30b8-a73a-44bb-92db-27a504e0ec85"
  }
]
```

The cloud template in this example has two versions in the catalog. To get the schema and create a custom form for version 2.0.0, assign a variable.

```
$ version_id='2.0.0'
```

Use the catalog ID and the version ID to get the schema for the cloud template.

```
$ schema='curl -k -X GET \
"$url/catalog/api/admin/items/$item_id/versions/$version_id \
-H "Authorization: Bearer $access_token" \
-H 'Content-Type: application/json' | jq -r .schema'
```

Use the schema to create a custom form.

```
$ curl -k -X POST \
"$url/form-service/api/forms/designer/request?
sourceType=com.vmw.blueprint.version&sourceId=$item_id/$version_id&formType=requestForm" \
-H "Authorization: Bearer $access_token" \
-H 'Content-Type: application/json' \
-d '$schema' | jq "."
```

The response shows the schema for **`Demo Blueprint`** version 2.0.0. It also shows the form ID or `4e383140-51a4-44a7-afcd-26296079596d`.

```
{
  "tenant": "733178c1-0620-478c-96b4-2b04ece1478d",
  "id": "4e383140-51a4-44a7-afcd-26296079596d",
  "name": "Demo Blueprint / 2.0.0",
  "form": "{\"layout\":{\"pages\":
[{\"id\":\"page_general\",\"title\":\"General\",\"sections\":
[{\"id\":\"section_project\",\"fields\":
[{\"id\":\"project\",\"display\":\"dropDown\",\"signpostPosition\":\"right-middle\"}]},
{\"id\":\"section_deploymentName\",\"fields\":
```

```
[{\"id\":\"deploymentName\",\"display\":\"textField\",\"signpostPosition\":\"right-
middle\"}]},{\"id\":\"section_stringInput\",\"fields\":
[{\"id\":\"stringInput\",\"display\":\"textField\",\"state\":{\"visible\":true,\"read-
only\":false},\"signpostPosition\":\"right-middle\"}]},
{\"id\":\"section_e0bcefb5\",\"fields\":
[{\"id\":\"numberInput\",\"display\":\"dropDown\",\"state\":{\"visible\":true,\"read-
only\":false},\"signpostPosition\":\"right-middle\"}]}]}]},\"schema\":{\"project\":
{\"label\":\"Project\",\"type\":{\"dataType\":\"string\",\"isMultiple\":false},\"valueList\":
{\"id\":\"projects\",\"type\":\"scriptAction\"},\"constraints\":
{\"required\":true}},\"numberInput\":{\"label\":\"numberInput\",\"type\":
{\"dataType\":\"integer\",\"isMultiple\":false},\"valueList\":{\"id\":\"bdimov/
actionThatReturnsLong\",\"type\":\"scriptAction\",\"parameters\":[]},\"constraints\":
{\"required\":true}},\"stringInput\":{\"label\":\"stringInput\",\"type\":
{\"dataType\":\"string\",\"isMultiple\":false},\"constraints\":
{\"required\":true}},\"deploymentName\":{\"label\":\"Deployment Name\",\"type\":
{\"dataType\":\"string\",\"isMultiple\":false},\"constraints\":{\"required\":true,\"max-
value\":900}}},\"options\":{\"externalValidations\":[]}}",
  "sourceType": "com.vmw.blueprint.version",
  "sourceId": "890275e3-cbc6-3778-af0d-d9eddf4664a0/2.0.0",
  "type": "requestForm",
  "status": "ON",
  "formFormat": "JSON",
  "styles": ""
}
```


Update the form schema to change the visibility of the Deployment Name from true to false.

```
$ curl -k -X POST \
-H "Authorization: Bearer $access_token" \
-H 'Content-Type: application/json' \
"$url/form-service/api/forms" \
-d '{
   "name":"Demo Blueprint / 2.0.0",

"form":'{"layout":{"pages":[{"id":"page_general","sections":[{"id":"section_project","fields":
[{"id":"project","display":"dropDown","signpostPosition":"right-middle","state":
{"visible":true,"read-only":false}}]},{"id":"section_deploymentName","fields":
[{"id":"deploymentName","display":"textField","state":{"visible":"false","read-
only":false},"signpostPosition":"right-middle"}]}]}]},"schema":{"project":
{"label":"Project","type":{"dataType":"string","isMultiple":false},"valueList":
{"id":"projects","type":"scriptAction"},"constraints":{"required":true}},"deploymentName":
{"label":"Deployment Name","type":{"dataType":"string","isMultiple":false},"constraints":
{"required":true,"max-value":900}},"options":{"externalValidations":[]}}}'
   "status":"ON", // set ON to enable the custom form, OFF to disable
   "type":"requestForm",
   "sourceId":"$item_id/$version_id",
   "sourceType":"com.vmw.blueprint.version"
 }'
```

If you want to see the updated form, use the form ID from the schema.

```
curl -k -X GET "$url/form-service/api/forms/4e383140-51a4-44a7-afcd-26296079596d" \
-H "Authorization: Bearer $token" \
-H 'Content-Type: application/json' | jq "."
```

Version the form and give it a name that will help you to identify it in a list later.

```
$ curl -k -X POST \
"$url/form-service/api/forms/versions" \
-H "Authorization: Bearer $token" \
-H 'Content-Type: application/json' \
-d '{
   "name":"Version1_Deployment_Visibility_FALSE",
   "sourceId":"$item_id/$version_id",
   "sourceType":"com.vmw.blueprint.version",
   "formType":"requestForm"
}' | jq "."
```

You can repeatedly update and version the form.

- Each time you update the form, you are updating the current version.
- Each time you version the form, you save the updated form with a new name.

If you want to restore a version of the form, list your form versions to find the version of the form that you want.

```
$ curl -k -X GET \
"$url/form-service/api/forms/versions?sourceType=com.vmw.blueprint.version&sourceId=$item_id/
$version_id&formType=requestForm" \
-H "Authorization: Bearer $token" \
-H 'Content-Type: application/json'
```

The response includes the IDs and names of the custom forms.

```
[
  {
    "id": "4bfdd822-8975-4883-a8b2-22396ddf9395",
    "createdDate": "2023-05-17T14:54:07.187+0000",
    "createdBy": "user@mycompany.com",
    "name": "Version1_Deployment_Visibility_FALSE"
  },
  {
    "id": "bed58877-e408-41ce-ad20-3bc48c569d84",
    "createdDate": "2023-05-17T14:39:16.267+0000",
    "createdBy": "user@mycompany.com",
    "name": "Version2_TextField_Added"
  }
]
```

Assign the ID of the form you want to restore to the form version ID variable.

```
$ formVersionID='4bfdd822-8975-4883-a8b2-22396ddf9395'
```

Restore the form.

```
$ curl -k -X PATCH \
"$url/form-service/api/forms/versions/$formVersionId/restore" \
```

```
-H "Authorization: Bearer $token" \
-H 'Content-Type: application/json'
```

What to do next

You have learned how to create, update, version, and restore a custom form.

Remove a Cloud Template Version from a VMware Aria Automation Service Broker Catalog

If you want to remove a cloud template version, you make a POST request. The body of the input indicates the version to remove.

If you have a versioned and released cloud template in VMware Aria Automation Service Broker, it appears in the catalog source. If you do not want that cloud template version to be available for deployment, you must remove it from the catalog.

Prerequisites

- Verify that all general prerequisites and prerequisites for the Automation Assembler Blueprint service have been satisfied. See Chapter 3 Prerequisites for API Use Case Examples.
- Verify that you have the cloud template ID for the cloud template you want to remove. See Create and Update a Cloud Template.
- Verify that you have the cloud template version that you want to remove from the catalog. See Version and Release a Cloud Template to a VMware Aria Automation Service Broker Catalog.

Procedure

1 Assign the cloud template ID variable.

```
cloud_template_id='<your_cloud_template_id>'
```

2 Assign the cloud template version variable.

```
cloud_template_version='<your_cloud_template_version>'
```

your_cloud_template_version is the version that you want to remove.

3 Remove a version of your cloud template from the catalog.

```
curl -X POST \
  $url/blueprint/api/blueprints/$cloud_template_id/versions/$cloud_template_version/action/
unrelease?apiVersion=$api_version \
  -H "Authorization: Bearer $access_token" \
  -H 'Content-Type: application/json' | jq "."
```

4 To see the change in the Automation Service Broker UI, select the **Content & Policies** tab.

   a Select **Content Sources**.

   b Select the name of the content source with your cloud template.

   c On the Content Source Details screen that appears, click **Save and Import**.

Results

The cloud template version you specified is removed from the Automation Service Broker catalog. Any other released cloud template versions remain listed.

Example: Remove a Cloud Template Version

Remove version 5 of your cloud template with ID **`fa6b42d5-ac46-451d-8917-b2f7e527b785`**.

Assign variables.

```
$ url='https://appliance.domain.com'
$ api_version='2019-09-12'
$ cloud_template_id='fa6b42d5-ac46-451d-8917-b2f7e527b785'
$ cloud_template_version='v5'
```

Remove the cloud template version.

```
$ curl -X POST \
  $url/blueprint/api/blueprints/$cloud_template_id/versions/$cloud_template_version/action/
unrelease?apiVersion=$api_version \
  -H "Authorization: Bearer $access_token" \
  -H 'Content-Type: application/json' | jq "."
```

A snippet of the response shows the cloud template version with a `VERSIONED` status.

```
...
  "blueprintId": "1f170637-81a3-4257-b1cd-b2219ee8034c",
  "name": "MyExampleCloudTemplate",
  "description": "Basic Cloud Machine cloud template",
  "version": "v5",
  "tags": [],
  "content": "formatVersion: 1\ninputs:\n  flavor:\n    type: string\n    title: Flavor\n
description: Flavor Mapping Name\n  image:\n    type: string\n    title: Image\n
description: Image Mapping Name\n  count:\n    type: integer\n    minimum: 1\n    default:
1\n    maximum: 2\n    title: Number of Instances\nresources:\n  BasicCloudMachine:\n
type: Cloud.Machine\n    properties:\n      name: BasicCloudMachine\n      flavor: '$
{input.flavor}'      \n      image: '${input.image}'\n      count: '${input.count}'\n
tags: [\n        {\n          \"key\": \"env\",\n          \"value\": \"prod\"\n        }
\n      ]\n",
  "status": "VERSIONED",
  "versionDescription": "Creating a version from the current draft",
  "versionChangeLog": "Creating a version v5",
  "valid": true
}
```

Test Your Cloud Template Deployment

To test the deployment of a cloud template, you use the Blueprint APIs to make a POST request with the cloud template ID as input.

Before deploying a cloud template, you can test the syntax and placement of your cloud template to ensure deployment viability. If errors are reported in the test, you must fix the errors and test again before deploying the cloud template.

Prerequisites

- Verify that all general prerequisites and prerequisites for the Automation Assembler Blueprint service have been satisfied. See Chapter 3 Prerequisites for API Use Case Examples.
- Verify that the flavor mapping and image mapping specified in the released Automation cloud template version exist in your cloud account. See Create Flavor Mappings and Create Image Mappings.
- Verify that you have the ID for the cloud template you want to deploy. See Create and Update a Cloud Template.

Procedure

1 Assign the cloud template ID variable.

```
cloud_template_id='<your_cloud_template_id>'
```

2 Assign image mapping and flavor mapping variables for the cloud template you intend to deploy.

```
image_mapping='<your_image_mapping_name>'
flavor_mapping='<your_flavor_mapping_name>'
```

The image mapping specifies the OS image for a VM. The flavor mapping specifies the CPU count and RAM of a VM.

3 Test the cloud template deployment.

```
curl -X POST \
  $url/blueprint/api/blueprint-requests?apiVersion=$api_version \
  -H "Authorization: Bearer $access_token" \
  -H 'Content-Type: application/json' \
  -d '{
    "simulate":true,
    "blueprintId": "'"$cloud_template_id"'",
    "inputs": {
        "count": 2,
        "image":"'"$image_mapping"'",
        "flavor":"'"$flavor_mapping"'"
    }
}' | jq "."
```

4 Examine the response and assign the cloud template request ID.

```
cloud_template_request_id='<your_cloud_template_request_id>'
```

5 Get the status of the test request.

```
curl -X GET \
  $url/blueprint/api/blueprint-requests/$cloud_template_request_id?apiVersion=$api_version
\
  -H "Authorization: Bearer $access_token" | jq "."
```

Example: Test a Deployment

For a cloud template with ID **`1f170637-81a3-4257-b1cd-b2219ee8034c`**, test the deployment with image mapping set to **`ubuntu`** and flavor mapping set to **`small`**.

Assign variables.

```
$ url='https://appliance.domain.com'
$ api_version='2019-09-12'
$ cloud_template_id='1f170637-81a3-4257-b1cd-b2219ee8034c'
$ image_mapping='ubuntu'
$ flavor_mapping='small'
```

Test the cloud template deployment.

```
$ curl -X POST \
  $url/blueprint/api/blueprint-requests?apiVersion=$api_version \
  -H "Authorization: Bearer $access_token" \
  -H 'Content-Type: application/json' \
  -d '{
    "simulate":true,
    "blueprintId": "'"$cloud_template_id"'",
    "inputs": {
        "count": 2,
        "image":"'"$image_mapping"'",
        "flavor":"'"$flavor_mapping"'"
    }
}' | jq "."
```

A snippet of the response shows the cloud template request ID.

```
{
  "id": "5c33355e-fc52-4a30-97c3-3752cf9b644e",
  "createdAt": "2019-10-11T00:11:55.544Z",
...
  "blueprintId": "1f170637-81a3-4257-b1cd-b2219ee8034c",
...
```

Assign the cloud template request ID variable.

```
$ cloud_template_request_id='5c33355e-fc52-4a30-97c3-3752cf9b644e'
```

Request the status of the deployment.

```
$ curl -X GET \
  $url/blueprint/api/blueprint-requests/$cloud_template_request_id?apiVersion=$api_version \
  -H "Authorization: Bearer $access_token" | jq "."
```

A snippet of the response shows the status of the deployment test request.

```
...
  "blueprintId": "1f170637-81a3-4257-b1cd-b2219ee8034c",
  "inputs": {
    "count": 2,
    "image": "ubuntu",
    "flavor": "small"
  },
  "status": "FINISHED",
...
```

What to do next

If your test deployment is successful, you are ready to deploy your cloud template.

Deploy Your Cloud Template

To request the deployment of a cloud template, you use the Blueprint APIs to make a POST request with the cloud template ID as input.

The prerequisites of this task call for verifying that you have the cloud template ID of the cloud template you want to deploy. In addition, this procedure includes an optional step to deploy a cloud template without a cloud template ID by providing contents inline instead.

Prerequisites

- Verify that all general prerequisites and prerequisites for the Automation Assembler Blueprint service have been satisfied. See Chapter 3 Prerequisites for API Use Case Examples.
- Verify that you have a project ID for the project that includes cloud zones configured to support the resource requirements of your cloud template. See Create a Project to use in Automation Assembler.
- Verify that the flavor mapping and image mapping specified in the released Automation cloud template version exist in your cloud account. See Create Flavor Mappings and Create Image Mappings.
- Verify that you have the ID for the cloud template you want to deploy. See Create and Update a Cloud Template.
- Verify that you have tested your cloud template deployment to ensure deployment viability. See Test Your Cloud Template Deployment.

Procedure

1 Assign the cloud template ID variable.

```
cloud_template_id='<your_cloud_template_id>'
```

2 To deploy a cloud template, assign variables for image mapping and flavor mapping.

```
image_mapping='<your_image_mapping_name>'
flavor_mapping='<your_flavor_mapping_name>'
```

The image mapping specifies the OS image for a VM. The flavor mapping specifies the CPU count and RAM of a VM.

3 Request a deployment of a cloud template.

```
curl -X POST \
  $url/blueprint/api/blueprint-requests?apiVersion=$api_version \
  -H "Authorization: Bearer $access_token" \
  -H 'Content-Type: application/json' \
  -d '{
    "description": "requesting deployment from cloud template",
    "blueprintId": "'"$cloud_template_id"'",
    "inputs": {
          "count": 2,
        "image":"'"$image_mapping"'",
        "flavor":"'"$flavor_mapping"'"
    }
}' | jq "."
```

4 Examine the response to get the cloud template request ID and the deployment ID.

5 Assign the cloud template request ID and the deployment ID.

```
cloud_template_request_id='<your_cloud_template_request_id>'
deployment_id='<your_deployment_id>'
```

6 (Optional) If you do not have a cloud template ID, you can also request a cloud template deployment with contents inline.

a Validate the cloud template before creating it.

```
curl -X POST \
  $url/blueprint/api/blueprint-validation?apiVersion=$api_version \
  -H 'Content-Type: application/json' \
  -H "Authorization: Bearer $access_token"  \
  -d '{
    "name" : "'"$cloud_template_id"'",
    "description" : "Basic Cloud Machine cloud template",
    "content" : "formatVersion: 1\ninputs:\n  flavor:\n    type: string\n
title: Flavor\n    description: Flavor Mapping Name\n  image:\n    type:
string\n    title: Image\n    description: Image Mapping Name\n  count:\n
type: integer\n    minimum: 1\n    default: 1\n    maximum: 2\n    title:
Number of Instances\nresources:\n  BasicCloudMachine:\n    type: Cloud.Machine\n
properties:\n      name: BasicCloudMachine\n      flavor: '\''${input.flavor}'\''
\n      image: '\''${input.image}'\''\n      count: '\''${input.count}'\''",
    "projectId" : "'"$project_id"'",
    "requestScopeOrg": false
}' | jq "."
```

b Examine the response to confirm that you see `"valid":true`.

c Request the cloud template deployment.

```
curl -X POST \
  $url/blueprint/api/blueprint-requests \
  -H "Authorization: Bearer $access_token" \
  -H 'Content-Type: application/json' \
  -d '{
    "description": "requesting deployment from inline cloud template",
    "projectId": "'"$project_id"'",
    "inputs": {
          "count": "2",
        "image":"'"$image_mapping"'",
        "flavor":"'"$flavor_mapping"'"
    },
    "content" : "formatVersion: 1\ninputs:\n  flavor:\n    type: string\n
title: Flavor\n    description: Flavor Mapping Name\n  image:\n    type:
string\n    title: Image\n    description: Image Mapping Name\n  count:\n
type: integer\n    minimum: 1\n    default: 1\n    maximum: 2\n    title:
Number of Instances\nresources:\n  BasicCloudMachine:\n    type: Cloud.Machine\n
properties:\n      name: BasicCloudMachine\n      flavor: '\''${input.flavor}'\''
\n      image: '\''${input.image}'\''\n      count: '\''${input.count}'\''"
}' | jq "."
```

7 Look up the status of the cloud template deployment.

```
curl -X GET \
  $url/api/blueprint-requests/$cloud_template_request_id?apiVersion=$api_version \
  -H "Authorization: Bearer $access_token" | jq "."
```

Example: Deploy a Cloud Template

For a cloud template with ID **`1f170637-81a3-4257-b1cd-b2219ee8034c`**, request the deployment with image mapping set to **`ubuntu`** and flavor mapping set to **`small`**.

Assign variables.

```
$ url='https://appliance.domain.com'
$ api_version='2019-09-12'
$ cloud_template_id='1f170637-81a3-4257-b1cd-b2219ee8034c'
$ image_mapping='ubuntu'
$ flavor_mapping='small'
```

Request the deployment of a cloud template.

```
$ curl -X POST \
  $url/blueprint/api/blueprint-requests?apiVersion=$api_version \
  -H "Authorization: Bearer $access_token" \
  -H 'Content-Type: application/json' \
  -d '{
    "description": "requesting deployment from cloud template",
    "blueprintId": "'"$cloud_template_id"'",
    "inputs": {
          "count": 2,
        "image":"'"$image_mapping"'",
        "flavor":"'"$flavor_mapping"'"
    }
}' | jq "."
```

A snippet of the response shows the cloud template request ID, the deployment ID, and the cloud template ID.

```
{
  "id": "889f95a8-79a3-4b2f-b19e-32d1536dd69a",
  "createdAt": "2019-10-11T00:11:55.544Z",
...
  "projectName": "Example-project",
  "deploymentId": "15454178-63fc-42ea-b4ad-7ed8a5cdb128",
  "requestTrackerId": "889f95a8-79a3-4b2f-b19e-32d1536dd69a",
...
  "blueprintId": "1f170637-81a3-4257-b1cd-b2219ee8034c",
...
```

Assign the cloud template request ID variable.

```
$ cloud_template_request_id='889f95a8-79a3-4b2f-b19e-32d1536dd69a'
```

Request the status of the deployment.

```
$ curl -X GET \
  $url/blueprint/api/blueprint-requests/$cloud_template_request_id?apiVersion=$api_version \
  -H "Authorization: Bearer $access_token" | jq "."
```

A snippet of the response shows the cloud template request ID and the cloud template ID with the status of the deployment request. If the deployment fails, the failure message indicates the reason for the failure.

```
{
  "id": "889f95a8-79a3-4b2f-b19e-32d1536dd69a"
  ...
  "blueprintId": "1f170637-81a3-4257-b1cd-b2219ee8034c",
  "inputs": {
    "count": 2,
    "image": "ubuntu",
    "flavor": "small"
  },
  "status": "FINISHED",
...
```

What to do next

Use the deployment ID to look up resource information for your deployment.

Specify SCSI disk placement using the Automation API

When attaching a disk object to a vSphere VM, you can specify the SCSI controller and the logical unit number (LUN) disk properties so that you can identify the disk when taking day 2 actions.

To attach a disk to a VM, you need the block device ID of the disk and and the ID of the VM. The following procedure includes steps to:

- List all machines in your environment to find the ID of the vSphere VM that you want to attach a disk to.
- List all block devices in your environment to find the disk you want to attach.
- Attach the disk specifying the SCSI controller and LUN.

Prerequisites

- Verify that all general prerequisites and prerequisites for the Automation Assembler Infrastructure as a Service (IaaS) service have been satisfied. See Chapter 3 Prerequisites for API Use Case Examples.

Procedure

1 List all the machines in your environment.

```
curl -X GET -H 'Content-Type: application/json' -H "Authorization: Bearer $access_token"
$url/iaas/api/machines?apiVersion=$api_version | jq "."
```

To identify the vSphere VM where you want to attach your disk, look for the name and the region ID in the response. The `self:href` path includes the machine ID.


```
{
      "powerState": "OFF",
      "externalRegionId": "Datacenter:datacenter-2",
      "cloudAccountIds": [

"e0f23c91d5ecca75-7f703c5265a63d87-7e3d8d60a55d1306cc791422547ead9153c3bdf1c802400819ad45a3
41cba1f3-b37e594a2e813475574a7c3c42b5d"
      ],
      "provisioningStatus": "READY",
      "customProperties": {
        "osType": "LINUX",
        "vcUuid": "1f9678f0-90d1-4347-82ee-f1ac2fac4216",
        "memoryGB": "1",
        "datacenter": "Datacenter:datacenter-2",
        "instanceUUID": "503a00ea-5ce5-3ae8-db3d-ebbd537eed5f",
        "softwareName": "Ubuntu Linux (64-bit)",
        "cpuCount": "2",
        "memoryInMB": "1024"
      },
      "externalId": "503a00ea-5ce5-3ae8-db3d-ebbd537eed5f",
      "name": "wordpress-mcm827-142063808276-ovf-backing",
      "id": "0355cee4-5d88-36b6-9a87-5f37b5baa6e2",
      "createdAt": "2022-04-02",
      "updatedAt": "2022-04-02",
      "organizationId": "f670fdfc-66d6-4689-9793-d524e7066d1e",
      "orgId": "f670fdfc-66d6-4689-9793-d524e7066d1e",
      "_links": {
        "network-interfaces": {
          "hrefs": [
            "/iaas/api/machines/0355cee4-5d88-36b6-9a87-5f37b5baa6e2/network-interfaces/
1c31ee02-c83c-3229-84ec-929f3592494a"
          ]
        },
        "cloud-accounts": {
          "hrefs": [
            "/iaas/api/cloud-accounts/
e0f23c91d5ecca75-7f703c5265a63d87-7e3d8d60a55d1306cc791422547ead9153c3bdf1c802400819ad45a34
1cba1f3-b37e594a2e813475574a7c3c42b5d"
          ]
        },
        "operations": {
          "hrefs": [
            "/iaas/api/machines/0355cee4-5d88-36b6-9a87-5f37b5baa6e2/operations/power-on",
            "/iaas/api/machines/0355cee4-5d88-36b6-9a87-5f37b5baa6e2/operations/snapshots",
            "/iaas/api/machines/0355cee4-5d88-36b6-9a87-5f37b5baa6e2/operations/resize",
            "/iaas/api/machines/0355cee4-5d88-36b6-9a87-5f37b5baa6e2/disks",
            "/iaas/api/machines/0355cee4-5d88-36b6-9a87-5f37b5baa6e2/disks/{id}"
          ]
        },
        "disks": {
          "hrefs": [
```

```
            "/iaas/api/machines/0355cee4-5d88-36b6-9a87-5f37b5baa6e2/disks/
effb94a6-41ad-3553-bb9a-22896785a283",
            "/iaas/api/machines/0355cee4-5d88-36b6-9a87-5f37b5baa6e2/disks/
0815ed4c-0a33-3058-a5f5-3032a426ded4",
            "/iaas/api/machines/0355cee4-5d88-36b6-9a87-5f37b5baa6e2/disks/
3804a2b2-99fc-3fca-bd2f-f0cfc3845b54"
          ]
        },
        "self": {
          "href": "/iaas/api/machines/0355cee4-5d88-36b6-9a87-5f37b5baa6e2"
        }
      }
    },
```

In this response example, the machine ID is `0355cee4-5d88-36b6-9a87-5f37b5baa6e2`.

2 Assign the machine ID.

```
machine_id='example-machineID-alphanumeric-string'
```

3 List the block devices in your environment.

```
curl -X GET -H 'Content-Type: application/json' -H "Authorization: Bearer $access_token"
$url/iaas/api/block-devices?apiVersion=$api_version | jq "."
```

Examine the response to find the block device that you want to attach. The block device must be available and in the same external region as the vSphere VM. The `self:href` path includes the block device ID.

```
...
    {
      "capacityInGB": 30,
      "status": "AVAILABLE",
      "type": "HDD",
      "persistent": false,
      "externalRegionId": "Datacenter:datacenter-2",
      "cloudAccountIds": [
        "e0f23c91d5ecca75-7f703c5265a63d87-
e78aab87e9c8d5cd4cd1da1a285403f0f4e77a5240720d093e147b830b172542-6b4f9990d36ee87558f04e6e8e
0ca"
      ],
      "provisioningStatus":"READY",
      "customProperties": {
        "diskKind": "Unmanaged"
      },
      "externalId": "74213f6d-ee11-4549-996a-772c0621f7d1",
      "name": "Hard disk 1",
      "id": "02c39b60-32f5-4a7e-9317-88bf8a3fe20c",
      "createdAt": "2022-04-02",
      "updatedAt": "2022-04-02",
      "organizationId": "f670fdfc-66d6-4689-9793-d524e7066d1e",
      "orgId": "f670fdfc-66d6-4689-9793-d524e7066d1e",
      "_links": {
        "cloud-accounts": {
```

```
          "hrefs": [
            "/iaas/api/cloud-accounts/e0f23c91d5ecca75-7f703c5265a63d87-
e78aab87e9c8d5cd4cd1da1a285403f0f4e77a5240720d093e147b830b172542-6b4f9990d36ee87558f04e6e8e
0ca"
          ]
        },
        "operations": {
          "hrefs": [
            "/iaas/api/block-devices/02c39b60-32f5-4a7e-9317-88bf8a3fe20c?
capacityInGB={capacityInGB}"
          ]
        },
        "self": {
          "href": "/iaas/api/block-devices/02c39b60-32f5-4a7e-9317-88bf8a3fe20c"
        }
      }
    },
...
```

In this response example, the block device ID is `02c39b60-32f5-4a7e-9317-88bf8a3fe20c`.

4 Assign the block device ID variable.

```
block_device_id='example-blockdeviceID-alphanumeric-string'
```

5 Attach the disk to the VM, specifying both the SCSI controller number and the LUN disk properties.

- For the SCSI controller number, you can specify any of four values: **`SCSI_Controller_0`**, **`SCSI_Controller_1`**, **`SCSI_Controller_2`**, **`SCSI_Controller_3`**.
- For the LUN, you can specify any integer value from 0 through 15. Unit 0 is the boot disk. Disks with higher LUN values attach after disks with lower LUN values.

Note If you do not specify a LUN, the disk attaches to the first available unit number. If you specify neither the SCSI controller nor the LUN, the disk attaches to the first available SCSI controller and first available unit number.

This request example uses `"scsiController": "SCSI_Controller_0"` and `"unitNumber": "0"`.

```
curl -X POST \
  $url/iaas/api/machines/$machine_id/disks?apiVersion=$api_version \
  -H 'Content-Type: application/json' \
  -H "Authorization: Bearer $access_token" \
  -d '{
    "blockDeviceId": "'$block_device_id'",
    "scsiController": "SCSI_Controller_0",
    "name": "BootDisk ",
    "description": "Unit 0 is the boot disk",
    "unitNumber": "0"
}' | jq "."
```

The response includes a selfLink value.

```
{
  "progress": 0,
  "status": "INPROGRESS",
  "name": "Provisioning",
  "id": "example-selfLink-alphanumeric-string",
  "selfLink": "/iaas/api/request-tracker/example-selfLink-alphanumeric-string"
}
```

6 Assign the selfLink variable.

```
selfLink_id='example-selfLink-alphanumeric-string'
```

7 Use the selfLink variable to track the progress of the disk attachment.

```
curl -X GET -H 'Content-Type: application/json' -H "Authorization: Bearer $access_token"
$url/iaas/api/request-tracker/$selfLink_id?apiVersion=$api_version | jq "."
```

After the request completes successfully, the response includes a list of resources with a machine that has your machine ID in the path.

```
{
  "progress": 100,
  "message": "success",
  "status": "FINISHED",
  "resources": [
    "/iaas/api/machines/example-machineID-alphanumeric-string"
  ],
  ...
}
```

8 (Optional) To specify SCSI controller and LUN when attaching additional disks, repeat Step 5 to Step 7.

Results

If you deploy a cloud template that includes a VM with multiple vSphere disk objects, the disks are assigned to the SCSI controller and LUN that you specified. In this way, you can identify the disks when taking day 2 actions such as formatting or resizing a disk.

How to Create Custom Naming Templates

If you enroll in custom naming in the Automation Assembler UI, you can create custom naming templates for your organization or project using the IaaS API. Custom naming provides a way for you to name deployed resources using conventions that you define.

If you create a custom name using `POST /iaas/api/naming`, any project that does not already have a template applied can use the naming template. An organization-level naming template applies to any project that is added to an organization.

A maximum of two resource naming templates can be applied to a project: one project-level naming template and one organization-level naming template. Each can have multiple resource types. If naming templates exist at both the project and organization level, the project uses the naming convention used in the project-level naming template.

Note If you created a project and specified a custom naming template as part of a previous project definition, that template is deprecated when you enroll in custom naming. To replace that template, create a custom name at the project level and add a new naming template.

The following examples show how to define custom naming templates for an organization and a project.

Prerequisites for creating a custom name

- Verify that all general prerequisites and prerequisites for the Automation Assembler Infrastructure as a Service (IaaS) service have been satisfied. See Chapter 3 Prerequisites for API Use Case Examples.
- Verify that you have enrolled in Custom Naming.

  For information about enrolling in custom naming, see Custom naming deployed resources in Automation Assembler.
- Verify that you know your organization ID. If you do not know your organization ID, perform the following steps:

  a Log in to the organization console of your VMware Aria Automation appliance, for example `https://appliance.domain.com/csp/gateway/portal/`

  b Click the drop-down arrow by your name to display the organization ID below the organization name.

  c Right click the icon next to the organization ID to copy the long string.
- If you are adding a project-level template, verify that you know the name and ID of the project that you want to assign to the template. For information on how to get a list of projects, see Create a Project to use in Automation Assembler.

How to create a custom name with organization scope

This example shows how to define an organization-level custom name with a naming template for a machine resource type in **`"orgId": "8327d53f-91ea-420a-8613-ba8f3149db95"`** .

```
curl -X POST \
  "$url/iaas/api/naming?apiVersion=$api_version" \
  -H "Authorization: Bearer $access_token" \
  -H 'Content-Type: application/json' \
  -d '{
    "name": "org-level custom name",
    "description": "Example organization-level custom name",
    "projects": [
      {
```

```
        "defaultOrg": true,
        "active": true,
        "projectName": "*",
        "projectId": "*",
        "orgId": "8327d53f-91ea-420a-8613-ba8f3149db95"
      }
    ],
    "templates": [
      {
        "uniqueName": true,
        "staticPattern": "",
        "counters": [],
        "incrementStep": 1,
        "pattern": "myvm-${##}",
        "startCounter": 1,
        "resourceTypeName": "Machine",
        "resourceType": "COMPUTE",
        "resourceDefault": true
      }
    ]
  }' | jq "."
```

The response shows the organization-level custom name with the custom name ID.

```
{
    "name": "org-level custom name",
    "description": "Example organization-level custom name",
    "id": "6ca7be62-627b-41f0-9505-fc29c1349a85",
    "_links": {
        "self": {
            "href": "/iaas/api/naming/6ca7be62-627b-41f0-9505-fc29c1349a85"
        }
    }
}
```

How to create a custom name with project scope

This example shows how to define a project-level custom name with naming templates for network and machine resource types, and assigned to a project that has not already been added to a custom name with project scope.

```
curl -X POST \
  "$url/iaas/api/naming?apiVersion=$api_version" \
  -H "Authorization: Bearer $access_token" \
  -H 'Content-Type: application/json' \
  -d '{
    "name": "proj-level custom name ",
    "description": "Example project-level custom name",
    "projects": [
      {
        "defaultOrg": false,
        "active": true,
        "projectName": "Example-CA-project",
        "projectId": "5944aacb-91de-4541-bb9e-ef2a5403f81b",
```

```
        "orgId": "8327d53f-91ea-420a-8613-ba8f3149db95"
      }
    ],
    "templates": [
      {
        "uniqueName": true,
        "staticPattern": "",
        "counters": [],
        "incrementStep": 1,
        "pattern": "myCAproject-${##}",
        "startCounter": 1,
        "resourceTypeName": "Network",
        "resourceType": "NETWORK",
        "resourceDefault": true
      },
      {
        "uniqueName": true,
        "staticPattern": "",
        "counters": [],
        "incrementStep": 1,
        "pattern": "myCAproject-${##}",
        "startCounter": 1,
        "resourceTypeName": "Machine",
        "resourceType": "COMPUTE",
        "resourceDefault": true
      }
    ]
 }' | jq "."
```

The response shows the project-level custom name with the custom name ID.

```
{
    "name": "proj-level custom name",
    "description": "Example project-level custom name",
    "id": "9afa9636-9536-4867-a325-fc70eb073a86",
    "_links": {
        "self": {
            "href": "/iaas/api/naming/9afa9636-9536-4867-a325-fc70eb073a86"
        }
    }
}
```

Requesting a Deployment from a Catalog Item Using Automation Service Broker APIs

9

To create catalog items and request deployments, you use the Automation Service Broker Catalog and Deployment APIs.

In this use case, you create a catalog source with a project that you previously used to create a cloud template, so that you can request a deployment from the catalog item. Optionally, if you want to expire your deployment, you can create a lease policy.

Read the following topics next:

- Create a Catalog Source and List Discovered Items
- Request Deployment
- Create a Lease Policy

Create a Catalog Source and List Discovered Items

To create a catalog source, you make a POST request with a project ID that has a cloud template version released to the project.

Because you are requesting the deployment of a cloud template from the catalog, this example lists the steps required to create a Automation Assembler Templates source type.

Prerequisites

- Verify that all general prerequisites and prerequisites for the Automation Service Broker Catalog service have been satisfied. See Chapter 3 Prerequisites for API Use Case Examples.
- Verify that you have the ID for a project that has the cloud template versioned and released to it. See the prerequisites section of Create and Update a Cloud Template.

Procedure

1 Assign the project ID variable.

```
project_id='<your_project_id>'
```

2 List all your catalog sources.

```
curl -X GET \
  $url/catalog/api/admin/sources?apiVersion=$api_version \
  -H "Authorization: Bearer $access_token" | jq "."
```

3 Examine the response to confirm that the name of the catalog source that you plan to create is not listed.

4 List all catalog source types.

```
curl -X GET \
  $url/catalog/api/types?apiVersion=$api_version \
  -H "Authorization: Bearer $access_token" | jq "."
```

5 Examine the response to find the catalog source type that you want to create.

6 Assign the catalog item type ID variable for the Automation Assembler Templates source type.

```
catalog_type_id='com.vmw.blueprint'
```

7 Create a catalog source for your cloud template.

```
curl -X POST \
  $url/catalog/api/admin/sources?apiVersion=$api_version \
  -H "Authorization: Bearer $access_token" \
  -H "Content-Type: application/json" \
  -d '{
    "config":{
    "sourceProjectId":"'$project_id'"
  },
    "typeId":"'$catalog_type_id'",
    "name":"<your_catalog_source_name>"
}' | jq "."
```

8 To obtain the catalog source ID, examine the response.

9 List all items discovered in the project.

```
curl -X GET \
  $url/catalog/api/admin/items?projectId=$project_id&apiVersion=$api_version \
  -H "Authorization: Bearer $access_token" | jq "."
```

10 To obtain the catalog item ID, examine the response.

11 (Optional) Assign the cloud template name variable.

```
cloud_template_name='<your_cloud_template_name_that_was_released_to_catalog>'
```

12 (Optional) To get the catalog item ID, you can also list discovered items by name.

```
curl -X GET \
  $url/catalog/api/admin/items?
projectId=$project_id&search=$cloud_template_name&apiVersion=$api_version \
  -H "Authorization: Bearer $access_token" | jq "."
```

Example: Create a catalog source and list discovered items

Create a catalog source for a cloud template named **`BasicCloudMachine`**.

Assign variables.

```
$ url='https://appliance.domain.com'
$ api_version='2020-08-25'
$ project_id='394a4ccb-22c6-4ef0-8c75-8b77efbefb51'
```

List all available sources in your catalog.

```
$ curl -X GET \
  $url/catalog/api/admin/sources?apiVersion=$api_version \
  -H "Authorization: Bearer $access_token" | jq "."
```

Examine the response to confirm that the name of the catalog source that you plan to create is not listed. The following snippet shows that you cannot create a catalog source with the name **`Catalog Source from Blueprintecho s`**.

```
...
      "id": "753d24a3-e2b0-4d5e-bba6-9e32e5964c69",
      "name": "Catalog Source from Blueprintecho s",
...
```

List all available catalog source types.

```
$ curl -X GET \
  $url/catalog/api/types?apiVersion=$api_version \
  -H "Authorization: Bearer $access_token" | jq "."
```

Examine the response to find the ID for a Automation Assembler Templates source type.

```
...
      "id": "com.vmw.blueprint",
      "name": "Automation Assembler Template",
      "baseUri": "http://catalog-service:8000/catalog/api/provider/blueprint",
      "createdBy": "deploymentservice",
...
```

Assign the source type to the catalog item type variable.

```
$ catalog_type_id='com.vmw.blueprint'
```

Create a catalog source of the Automation Assembler Templates source type.

```
$ curl -X POST \
  $url/catalog/api/admin/sources?apiVersion=$api_version \
  -H "Authorization: Bearer $access_token" \
  -H "Content-Type: application/json" \
  -d '{
    "config":{
    "sourceProjectId":"'$project_id'"
  },
    "typeId":"'$catalog_type_id'",
    "name":"Catalog Source from Automation Assembler Templates"
}' | jq "."
```

A snippet of the response shows the catalog source ID.

```
{
  "id": "753d24a3-e2b0-4d5e-bba6-9e32e5964c69",
  "name": "Catalog Source from Automation Assembler Templates",
  "typeId": "com.vmw.blueprint",
  "createdAt": "2021-11-08T22:02:33.553Z",
...
```

Assign the catalog source ID.

```
$ catalog_source_id='753d24a3-e2b0-4d5e-bba6-9e32e5964c69'
```

List items discovered in your project.

```
$ curl -X GET \
  $url/catalog/api/admin/items?projectId=$project_id&apiVersion=$api_version \
  -H "Authorization: Bearer $access_token" | jq "."
```

A snippet of the response shows the catalog item ID with your cloud template name.

```
{
      "id": "718917c0-1e02-3141-8142-11da5acaed8f",
      "name": "BasicCloudMachine",
      "description": "Basic Cloud Machine cloud template",
      "sourceId": "753d24a3-e2b0-4d5e-bba6-9e32e5964c69",
...
```

Assign the catalog item ID.

```
$ catalog_item_id='718917c0-1e02-3141-8142-11da5acaed8f'
```

What to do next

Before requesting a deployment, you use the `catalog_source_id` to create entitlement for the catalog source. Or you can use the `catalog_item_id` to create entitlement for the cloud template item. See Create a Content Sharing Policy.

Request Deployment

To request a deployment from a catalog item, you make a POST request with a project ID that has a cloud template version released to the project. The request body includes the ID of the catalog item from which you are requesting the deployment, and the version of the released cloud template.

Prerequisites

- Verify that all general prerequisites and prerequisites for both the Automation Service Broker Catalog service and the Automation Service Broker Deployment service have been satisfied. See Chapter 3 Prerequisites for API Use Case Examples.
- Verify that the flavor mapping and image mapping specified in the released cloud template version exist in your cloud account. See Create Flavor Mappings and Create Image Mappings.
- Verify that you have the ID for a project that has the cloud template versioned and released to it. See the prerequisites section of Create and Update a Cloud Template.
- Verify that you have the ID of the catalog item from which you plan to request a deployment. See Create a Catalog Source and List Discovered Items.
- Verify that you have the version of a cloud template released to the project that you want to request for deployment. See Version and Release a Cloud Template to a VMware Aria Automation Service Broker Catalog.
- Verify that you have created an entitlement for your catalog item or create one using Create a Content Sharing Policy.

Procedure

1 Assign the project ID variable.

```
project_id='<your_project_id>'
```

2 Assign the catalog item ID variable.

```
catalog_item_id='<your_catalog_item_id>'
```

3 List the available versions of the catalog item that can be requested.

```
curl -X GET \
  $url/catalog/api/items/$catalog_item_id/versions \
  -H "Authorization: Bearer $access_token" | jq "."
```

4 Examine the response to verify the version of the item that you want has been published to the catalog.

5 Assign the catalog item version.

```
catalog_item_version='<your_catalog_item_version>'
```

6 Assign your deployment name variable.

```
deployment_name='<your_deployment_name>'
```

If your deployment name includes spaces, use double quotes as in the following example.

```
deployment_name="This deployment name includes spaces"
```

a To ensure that the deployment name you plan to use does not already exist, list all deployments.

```
curl -X GET \
  -G --data-urlencode "name=$deployment_name" \
  $url/deployment/api/deployments?apiVersion=$api_version \
  -H "Authorization: Bearer $access_token" | jq "."
```

b Examine the response. If your deployment name appears, create a new name and reassign your deployment name variable.

7 To deploy a cloud template, assign variables for image mapping and flavor mapping.

```
image_mapping='<your_image_mapping_name>'
flavor_mapping='<your_flavor_mapping_name>'
```

The image mapping specifies the OS image for a VM. The flavor mapping specifies the CPU count and RAM of a VM.

8 Request the deployment from a catalog item.

```
curl -X POST \
  $url/catalog/api/items/$catalog_item_id/request?apiVersion=$api_version \
  -H "Authorization: Bearer $access_token"  \
  -H 'Content-Type: application/json' \
  -d '{
    "deploymentName":"'"$deployment_name"'",
    "projectId":"'"$project_id"'",
    "inputs":{
        "count":1,
        "image":"'"$image_mapping"'",
        "flavor":"'"$flavor_mapping"'"
    },
    "version":"'"$catalog_item_version"'"
}' | jq "."
```

The **inputs** field includes values for request time variables such as size, image, or password.

9 To obtain the deployment ID, examine the response.

10 Assign the deployment ID variable.

```
deployment_id='<your_deployment_id>'
```

11 Get the status of the deployment.

```
curl -X GET \
  $url/deployment/api/deployments/$deployment_id?apiVersion=$api_version \
  -H "Authorization: Bearer $access_token" | jq "."
```

Example: Request Deployment of a Cloud Template from a Catalog Item

Request the deployment of a cloud template with catalog item ID **718917c0-1e02-3141-8142-11da5acaed8f**. When requesting the deployment, set image mapping set to **ubuntu** and flavor mapping set to **small**.

Assign variables.

```
$ url='https://appliance.domain.com'
$ api_version='2020-08-25'
$ project_id='394a4ccb-22c6-4ef0-8c75-8b77efbefb51'
$ catalog_item_id='718917c0-1e02-3141-8142-11da5acaed8f'
```

List available versions.

```
$ curl -X GET \
  $url/catalog/api/items/$catalog_item_id/versions \
  -H "Authorization: Bearer $access_token" | jq "."
```

A snippet of the response shows version numbers.

```
...
{
      "id": "v2",
      "description": "Creating a version from the current draft",
      "createdAt": "2021-11-087T19:33:04.445Z"
    },
    {
      "id": "v1",
      "description": "Creating a version from the current draft",
      "createdAt": "2021-11-08T19:25:43.327Z"
    }
...
```

Assign the catalog item version number.

```
$ catalog_item_version='v2'
```

Assign a deployment name and check to ensure that it does not already exist.

```
$ deployment_name="Example Deployment of Cloud Template"
$ curl -X GET \
  -G --data-urlencode "name=$deployment_name" \
  $url/deployment/api/deployments?apiVersion=$api_version \
  -H "Authorization: Bearer $access_token" | jq "."
```

A snippet of the response shows existing deployments. **`Example Deployment of Cloud Template`** is not listed.

```
{
      "id": "c14e787f-60ee-4cce-a5b5-c9440bf181ab",
      "name": "Not Example Deployment",
      "orgId": "c9258a19-fef0-4431-a999-d711e1741c60",
      "catalogItemId": "947b9db2-cf89-3b83-8035-bbcf83bd4c34",
...
```

To deploy a cloud template, you must assign image mapping and flavor mapping variables.

```
$ image_mapping='ubuntu'
$ flavor_mapping='small'
```

Request a deployment from the catalog item.

```
$ curl -X POST \
  $url/catalog/api/items/$catalog_item_id/request?apiVersion=$api_version \
  -H "Authorization: Bearer $access_token"  \
  -H 'Content-Type: application/json' \
  -d '{
    "deploymentName":"'"$deployment_name"'",
    "projectId":"'"$project_id"'",
    "inputs":{
        "count":1,
        "image":"'"$image_mapping"'",
        "flavor":"'"$flavor_mapping"'"
    },
    "version":"'"$catalog_item_version"'"
}' | jq "."
```

The response provides the deployment ID.

```
{
  "deploymentId": "3721d9e2-fce3-48eb-96e5-d8f381354610",
  "deploymentName": "Example Deployment of Cloud Template"
}
```

Assign the deployment ID.

```
$ deployment_id='3721d9e2-fce3-48eb-96e5-d8f381354610'
```

Get the deployment status.

```
$ curl -X GET \
  $url/deployment/api/deployments/$deployment_id?apiVersion=$api_version"
  -H "Authorization: Bearer $access_token" | jq "."
```

A snippet of the response shows the deployment status.

```
},
  "projectId": "394a4ccb-22c6-4ef0-8c75-8b77efbefb51",
```

```
  "status": "CREATE_SUCCESSFUL"
}
```

Create a Lease Policy

To create a lease policy for your deployment, you make a POST request with a project ID that has a cloud template version released to the project.

Creating a lease policy is optional. For example, you can create a lease policy to specify when you want a deployment to expire. You specify the policy with either a soft or hard lease enforcement type.

- If specified with soft enforcement, the policy can be overridden and will have lower priority than policies with hard enforcement.
- If specified with hard enforcement, the policy must be enforced. If strict enforcement is not possible, for example in cases of conflicting policies, the policy can be overridden but Automation Service Broker will report an error.

Prerequisites

- Verify that all general prerequisites and prerequisites for the Automation Service Broker Policies service have been satisfied. See Chapter 3 Prerequisites for API Use Case Examples.
- Verify that you have the ID for a project that has the cloud template versioned and released to it. See the prerequisites section of Create and Update a Cloud Template.

Procedure

1 Assign the project ID variable.

```
project_id='<your_project_id>'
```

2 Assign a lease policy with soft enforcement to your project.

```
curl -X POST \
  $url/policy/api/policies?apiVersion=$api_version \
  -H "Authorization: Bearer $access_token" \
  -H 'Content-Type: application/json' \
  -d '{
    "name":"<your_lease_policy_name>",
    "projectId":"'$project_id'",
    "definition":{
        "leaseGrace":1,
        "leaseTermMax":10,
        "leaseTotalTermMax":100
    },
    "enforcementType":"SOFT",
    "typeId":"com.mycompany.policy.deployment.lease"
}' | jq "."
```

Example: Create a lease policy with soft enforcement

Assign variables.

```
$ url='https://appliance.domain.com'
$ api_version='2020-08-25'
$ project_id='394a4ccb-22c6-4ef0-8c75-8b77efbefb51'
```

Create the soft lease policy named **`Sample Lease`**.

```
$ curl -X POST \
  $url/policy/api/policies?apiVersion=$api_version \
  -H "Authorization: Bearer $access_token" \
  -H 'Content-Type: application/json' \
  -d '{
    "name":"Sample Lease",
    "projectId":"'$project_id'",
    "definition":{
        "leaseGrace":1,
        "leaseTermMax":10,
        "leaseTotalTermMax":100
    },
    "enforcementType":"SOFT",
    "typeId":"com.vmware.policy.deployment.lease"
}' | jq "."
```

The response shows the lease policy.

```
{
  "id": "49893797-208c-4322-8ed5-061467674d54",
  "name": "Sample Lease",
  "typeId": "com.mycompany.policy.deployment.lease",
  "enforcementType": "SOFT",
  "orgId": "c9258a19-fef0-4431-a999-d711e1741c60",
  "projectId": "394a4ccb-22c6-4ef0-8c75-8b77efbefb51",
  "definition": {
    "leaseGrace": 1,
    "leaseTermMax": 10,
    "leaseTotalTermMax": 100
  },
  "createdAt": "2021-11-08T02:29:07.936Z",
  "createdBy": "admin@mycompany.com",
  "lastUpdatedAt": "2021-11-08T02:29:07.936Z",
  "lastUpdatedBy": "admin@mycompany.com"
}
```

Working with Deployments and Resources

10

To work with deployments, you use the Automation Assembler and Automation Service Broker Deployment APIs. You also use the Automation Service Broker Deployment APIs to manage or get information about resources in your deployment.

These use cases include examples of procedures you can follow to reconfigure your deployment after initial cloud template deployment. The first example deploys a new cloud template with a deployment ID. In subsequent examples, you use the deployment ID to update the deployment or reconfigure the cloud template components.

If you want to onboard a machine that was deployed outside of Automation Assembler, you use the Relocation Service APIs to create an onboarding plan so that you can onboard the machine. Onboarding use cases show how to create an onboarding plan and onboard a machine with or without a cloud template.

Read the following topics next:

- Deploy a Cloud Template with Contents Inline
- Look up Deployment Details
- Get Deployment Resource IDs
- Change the Lease on Your Deployment
- Delete Your Deployment
- Reconfigure Load Balancer
- Add a Disk to a Machine and Power It Off
- Viewing Billable Objects
- Onboarding virtual machines

Deploy a Cloud Template with Contents Inline

To request a deployment of a cloud template with contents inline, you use the Blueprint APIs to make a POST request with a project ID. Before requesting the deployment, you use the Deployment API to ensure that the deployment name does not already exist.

In this example, you deploy a cloud template by providing contents inline instead of providing a cloud template ID. The new cloud template has different content from previously deployed cloud templates including a load balancer component, a virtual machine component, and a network component. For information about deploying a cloud template by providing a cloud template ID, see Deploy Your Cloud Template.

Prerequisites

- Verify that all general prerequisites and prerequisites for the Automation Service Broker Deployment service have been satisfied. See Chapter 3 Prerequisites for API Use Case Examples.
- Verify that you have the ID for a project that is associated with the deployment that you are managing. See Request Deployment.
- Verify that the flavor mapping and image mapping specified in the cloud template to be deployed exist in your cloud account. See Create Flavor Mappings and Create Image Mappings.
- Verify that the cloud zone that you are deploying into is associated with your project. See Add a Cloud Zone to Your Project.
- Verify that a network profile is configured for the cloud account associated with the project. See Create Network Profiles.
- Assign an API version variable for the Blueprint API.

```
api_version_blueprint='2019-09-12'
```

Note The Automation Service Broker Deployment service and the Automation Assembler Blueprint service have different API version values. You set the API version value for the Automation Service Broker Deployment service when you satisfied the general prerequisites.

Procedure

1 Assign the project ID variable.

```
project_id='<your_project_id>'
```

2 Assign your deployment name variable.

```
deployment_name='<your_deployment_name>'
```

If your deployment name includes spaces, use double quotes as in the following example.

```
deployment_name="This deployment name includes spaces"
```

a To ensure that the deployment name you plan to use does not already exist, list all deployments.

```
curl -X GET \
  -G --data-urlencode "name=$deployment_name" \
  $url/deployment/api/deployments?apiVersion=$api_version \
  -H "Authorization: Bearer $access_token" | jq "."
```

b To verify that the deployment name does not already exist, examine the response. If your deployment name appears, create a new name and reassign your deployment name variable.

3 To deploy a cloud template, assign variables for image mapping and flavor mapping.

```
image_mapping='<your_image_mapping_name>'
flavor_mapping='<your_flavor_mapping_name>'
```

The image mapping specifies the OS image for a VM. The flavor mapping specifies the CPU count and RAM of a VM.

4 Request deployment of a cloud template with contents inline.

```
curl -X POST \
    $url/blueprint/api/blueprint-requests?apiVersion=$api_version_blueprint \
    -H "Authorization: Bearer $access_token" \
    -H 'Content-Type: application/json' \
    -d '{
      "deploymentName": "'"$deployment_name"'",
      "description": "requesting deployment with contents inline",
      "projectId": "'"$project_id"'",
      "inputs": {
            "flavor": "'"$flavor_mapping"'",
            "image" : "'"$image_mapping"'"
      },
      "content" : "formatVersion: 1\ninputs:\n  flavor:\n    type: string\n
title: Flavor\n    description: Flavor Mapping Name\n  image:\n    type:
string\n    title: Image\n    description: Image Mapping Name\nresources:\n
cloud-vm:\n    type: Cloud.AWS.EC2.Instance\n    properties:\n      name:
cloudvm\n      flavor: '\''${input.flavor}'\''\n      image: '\''${input.image}'\''\n
networks:\n        - name: '\''${resource.Cloud_Network_1.name}'\''\n
network: '\''${resource.Cloud_Network_1.id}'\''\n  Provider_LoadBalancer_1:\n    type:
Cloud.LoadBalancer\n    properties:\n      name: OC-LB\n      routes:\n
- protocol: HTTP\n          port: '\''80'\''\n          instanceProtocol:
HTTP\n          instancePort: '\''80'\''\n          healthCheckConfiguration:\n
protocol: HTTP\n            port: '\''80'\''\n            urlPath: /
index.html\n            intervalSeconds: 60\n            timeoutSeconds: 5\n
unhealthyThreshold: 5\n            healthyThreshold: 2\n      network: '\''$
```

```
{resource.Cloud_Network_1.name}'\''\n      instances:\n        - '\''${resource[\"cloud-
vm\"].id}'\''\n      internetFacing: false\n  Cloud_Network_1:\n    type:
Cloud.Network\n    properties:\n      name: provider\n      networkType: public\n"
}' | jq "."
```

5 To obtain the cloud template request ID and the deployment ID, examine the response.

6 Assign the cloud template request ID variable.

```
cloud_template_request_id='<your_cloud_template_request_id>'
```

7 Get the status of the request.

```
curl -X GET \
  $url/blueprint/api/blueprint-requests/$cloud_template_request_id?
apiVersion=$api_version_blueprint \
  -H "Authorization: Bearer $access_token" | jq "."
```

Example: Deploy a Cloud Template with Contents Inline

Request a deployment named **`Deployment with Cloud Template Contents Inline`**. When requesting the deployment, set image mapping set to **`ubuntu`** and flavor mapping set to **`small`**.

Assign variables.

```
$ url='https://appliance.domain.com'
$ api_version='2020-08-25'
$ api_version_blueprint='2019-09-12'
$ project_id='394a4ccb-22c6-4ef0-8c75-8b77efbefb51'
$ deployment_name="Deployment with Cloud Template Contents Inline"
```

To request deployment, you must assign image mapping and flavor mapping variables.

```
$ image_mapping='ubuntu'
$ flavor_mapping='small'
```

Request deployment of a cloud template with contents inline.

```
$ curl -X POST \
    $url/blueprint/api/blueprint-requests?apiVersion=$api_version_blueprint \
    -H "Authorization: Bearer $access_token" \
    -H 'Content-Type: application/json' \
    -d '{
      "deploymentName": "'"$deployment_name"'",
      "description": "requesting deployment with contents inline",
      "projectId": "'"$project_id"'",
      "inputs": {
            "flavor": "'"$flavor_mapping"'",
            "image" : "'"$image_mapping"'"
      },
      "content" : "formatVersion: 1\ninputs:\n  flavor:\n    type: string\n
title: Flavor\n    description: Flavor Mapping Name\n  image:\n    type:
string\n    title: Image\n    description: Image Mapping Name\nresources:\n
cloud-vm:\n    type: Cloud.AWS.EC2.Instance\n    properties:\n      name:
```

```
cloudvm\n      flavor: '\''${input.flavor}'\''\n      image: '\''${input.image}'\''\n
networks:\n        - name: '\''${resource.Cloud_Network_1.name}'\''\n
network: '\''${resource.Cloud_Network_1.id}'\''\n  Provider_LoadBalancer_1:\n    type:
Cloud.LoadBalancer\n    properties:\n      name: OC-LB\n      routes:\n        - protocol:
HTTP\n          port: '\''80'\''\n          instanceProtocol: HTTP\n          instancePort:
'\''80'\''\n          healthCheckConfiguration:\n            protocol: HTTP\n
port: '\''80'\''\n            urlPath: /index.html\n            intervalSeconds:
60\n            timeoutSeconds: 5\n            unhealthyThreshold: 5\n
healthyThreshold: 2\n      network: '\''${resource.Cloud_Network_1.name}'\''\n
instances:\n        - '\''${resource[\"cloud-vm\"].id}'\''\n      internetFacing: false\n
Cloud_Network_1:\n    type: Cloud.Network\n    properties:\n      name: provider\n
networkType: public\n"
}' | jq "."
```

A snippet of the response provides the cloud template request ID and the deployment ID.

```
...
  "type": "blueprint-request",
  "id": "bec37f54-3de5-451d-b484-a110c0ed8772",
...
  "projectId": "394a4ccb-22c6-4ef0-8c75-8b77efbefb51",
  "projectName": "example-project",
  "deploymentId": "5551a299-8b67-45e3-909e-a638d11b0d9f",
  "requestTrackerId": "bec37f54-3de5-451d-b484-a110c0ed8772",
  "deploymentName": "Deployment with Cloud Template Contents inline",
...
```

Assign the cloud template request ID.

```
$ cloud_template_request_id='bec37f54-3de5-451d-b484-a110c0ed8772'
```

Get the deployment status.

```
$ curl -X GET \
  $url/blueprint/api/blueprint-requests/$cloud_template_request_id?
apiVersion=$api_version_blueprint \
  -H "Authorization: Bearer $access_token" | jq "."
```

A snippet of the response shows the deployment status.

```
...
  "inputs": {
    "image": "ubuntu",
    "flavor": "small"
  },
  "status": "FINISHED",
...
```

What to do next

You can use the deployment ID to perform other operations on your deployment. Other operations include, changing the lease on your deployment, fetching deployment resources, reconfiguring a load balancer, or adding a disk to a machine and power it off.

Look up Deployment Details

To look up deployment details such as the resources provisioned for each deployment, you use the Deployment APIs to make a GET request that displays all available resources. Then you use the resource ID in the output to make a GET request that returns the details of a particular resource.

Prerequisites

- Verify that all general prerequisites and prerequisites for the Automation Service Broker Deployment service have been satisfied. See Chapter 3 Prerequisites for API Use Case Examples.
- Verify that you have a deployment ID for the deployment that you requested. See Deploy Your Cloud Template.

Procedure

1 Assign the deployment ID variable.

```
deployment_id='<your_deployment_id>'
```

2 Display all the available resources that are provisioned in your deployment.

```
curl -X GET \
  $url/deployment/api/deployments/$deployment_id?expand=resources&apiVersion=$api_version \
  -H "Authorization: Bearer $access_token" | jq "."
```

3 Examine the response to find the ID of the resource for which you want details.

4 Assign the deployment resource ID.

```
deployment_resource_id='<your_deployment_resource_id>'
```

5 Display the details of that resource.

```
curl -X GET \
  $url/deployment/api/deployments/$deployment_id/resources/$deployment_resource_id?
apiVersion=$api_version \
  -H "Authorization: Bearer $access_token" | jq "."
```

6 List the deployment events.

```
curl -X GET \
  $url/deployment/api/deployments/$deployment_id/events?apiVersion=$api_version \
  -H "Authorization: Bearer $access_token" | jq "."
```

In case of deployment failures, listing deployment events can help with debugging.

Example: Look up the details of a provisioned resource in your deployment

Display the resources provisioned in your deployment.

Assign variables.

```
$ url='https://appliance.domain.com'
$ api_version='2020-08-25'
$ deployment_id='15454178-63fc-42ea-b4ad-7ed8a5cdb128'
```

Look up deployment details.

```
$ curl -X GET \
  $url/deployment/api/deployments/$deployment_id?expand=resources&apiVersion=$api_version \
  -H "Authorization: Bearer $access_token" | jq "."
```

A snippet of the response shows the details for each resource details including a deployment resource ID.

```
...
    "resources": [
    {
      "id": "3994a33e-bd93-4eea-87f1-f99ff17717ce",
      "name": "BasicCloudMachine[0]",
...
```

Assign the deployment resource ID variable for the `BasicCloudMachine` resource.

```
$ deployment_resource_id='3994a33e-bd93-4eea-87f1-f99ff17717ce'
```

Display the details of that resource.

```
$ curl -X GET \
  $url/deployment/api/deployments/$deployment_id/resources/$deployment_resource_id?
apiVersion=$api_version \
  -H "Authorization: Bearer $access_token" | jq "."
```

A snippet of the response shows the details of the single resource.

```
{
  "id": "3994a33e-bd93-4eea-87f1-f99ff17717ce",
  "name": "BasicCloudMachine[0]",
  "type": "Cloud.Machine",
  "dependsOn": [],
  "createdAt": "2021-11-08T17:56:09.463Z",
  "properties": {
    "id": "/resources/compute/3114189206b1763d",
    "name": "BasicCloudMachine",
...
    "service": "EC2",
    "storage": {
      "disks": [
```

```
        {
          "name": "boot-disk",
          "type": "HDD",
...
     "networks": [
      {
        "name": "BasicCloudMachine_nic",
        "address": "172.16.1.98",
        "assignment": "dynamic"
      }
        ],
...
    "__ext:ComputeReservationTaskState:STARTED:SELECTED": "true",
    "__ext:ComputeAllocationTaskState:STARTED:START_COMPUTE_ALLOCATION": "true"
  },
  "state": "OK"
}
```

List events from the deployment.

```
$ curl -X GET \
  $url/deployment/api/deployments/$deployment_id/events?apiVersion=$api_version \
  -H "Authorization: Bearer $access_token" | jq "."
```

A snippet of the response shows successful events.

```
...
      "totalTasks": 3,
      "status": "SUCCESSFUL",
      "inputs": {
        "count": 2,
        "image": "ubuntu",
        "flavor": "small"
      }
...
```

Get Deployment Resource IDs

To get IDs of the resources in your deployment, you use the Deployment APIs to make a GET request.

To perform operations on the load balancer or virtual machine in your deployment, you need the IDs of those resources.

Prerequisites

- Verify that all general prerequisites and prerequisites for the Automation Service Broker Deployment service have been satisfied. See Chapter 3 Prerequisites for API Use Case Examples.
- Verify that you have the ID of the deployment you want to reconfigure. See Deploy a Cloud Template with Contents Inline.

Procedure

1 Assign your deployment ID variable.

```
deployment_id='<your_deployment_id>'
```

2 List the resources in your deployment.

```
curl -X GET \
  $url/deployment/api/deployments/$deployment_id/resources?apiVersion=$api_version \
  -H "Authorization: Bearer $access_token" | jq "."
```

3 Examine the response.

- Find the resource named `Provider_LoadBalancer_1` and copy the value to assign to the load balancer ID.
- Find the resource named `cloud-vm` and copy the value to assign to the virtual machine ID.

4 Assign variables for the resources.

```
load_balancer_id='<your_load_balancer_id>'
virtual_machine_id='<your_virtual_machine_id>'
```

Example: Get Deployment Resource IDs

Get the resource IDs for your deployment with ID **`5551a299-8b67-45e3-909e-a638d11b0d9f`**.

Assign variables.

```
$ url='https://appliance.domain.com'
$ api_version='2020-08-25'
$ deployment_id='5551a299-8b67-45e3-909e-a638d11b0d9f'
```

List the resources in your deployment.

```
$ curl -X GET \
  $url/deployment/api/deployments/$deployment_id/resources?apiVersion=$api_version \
  -H "Authorization: Bearer $access_token" | jq "."
```

A snippet of the response shows the ID for the resource `Provider_LoadBalancer_1` and the ID for the resource `cloud-vm`.

```
...
{
      "id": "d5b4569d-2234-4fc4-a594-45e6b0251588",
      "name": "Provider_LoadBalancer_1",
      "type": "Cloud.LoadBalancer",
...
{
      "id": "42f49781-1490-4a08-ae21-8baf383a72ac",
      "name": "cloud-vm",
```

```
      "type": "Cloud.AWS.EC2.Instance",
...
```

Assign the load balancer ID and virtual machine ID variables.

```
$ load_balancer_id='d5b4569d-2234-4fc4-a594-45e6b0251588'
$ virtual_machine_id='42f49781-1490-4a08-ae21-8baf383a72ac'
```

What to do next

Use the load balancer ID to reconfigure your load balancer. Use the virtual machine ID to add a disk to the VM and power off the VM.

Change the Lease on Your Deployment

To change the lease on your deployment, you use the Deployment APIs to make a POST request with a new lease expiration date.

The lease on your deployment is set to never expire by default. The following procedure shows how to add a lease expiration date. It also includes an optional step that shows how to reset the lease on your deployment back to never expire.

Prerequisites

- Verify that all general prerequisites and prerequisites for the Automation Service Broker Deployment service have been satisfied. See Chapter 3 Prerequisites for API Use Case Examples.
- Verify that you have the ID of the deployment you want to reconfigure. See Deploy a Cloud Template with Contents Inline.

Procedure

1 Assign your deployment ID variable.

```
deployment_id='<your_deployment_id>'
```

2 Get a list of actions available for your deployment.

```
curl -X GET \
  $url/deployment/api/deployments/$deployment_id/actions?apiVersion=$api_version \
  -H "Authorization: Bearer $access_token" | jq "."
```

3 Examine the response.

- Confirm that you see the action `"name": "ChangeLease"`.
- `"valid":true` indicates that the action is valid for the deployment.

4 Assign the action ID variable for the action `"name": "ChangeLease"`.

```
action_id='Deployment.ChangeLease'
```

5 List the deployment actions for the action ID .

```
curl -X GET \
  $url/deployment/api/deployments/$deployment_id/actions/$action_id?
apiVersion=$api_version \
  -H "Authorization: Bearer $access_token" | jq "."
```

6 Examine the response. The schema field shows the format of the input for an action on the deployment.

7 To change the lease on the deployment, assign the lease expiry date using the format specified in the schema as in the following example.

```
lease_expiry_date=2021-05-15T23:11:00Z
```

8 Change the lease expiration date.

```
curl -X POST \
  $url/deployment/api/deployments/$deployment_id/requests \
  -H "Authorization: Bearer $access_token" \
  -H 'Content-Type: application/json' \
  -d '{
    "actionId":"Deployment.ChangeLease",
    "inputs": {
        "Lease Expiration Date": "'"$lease_expiry_date"'"
    }
}' | jq "."
```

9 Examine the response and assign the request ID.

```
request_id='<your_request_id>'
```

10 Check the status of the request.

```
curl -X GET \
  $url/deployment/api/requests/$request_id?apiVersion=$api_version \
  -H "Authorization: Bearer $access_token" | jq "."
```

If the request is successful, the response shows `"status":"SUCCESSFUL"`.

11 (Optional) To change the lease on the deployment back to never expire, use the same **`action_id='Deployment.ChangeLease'`** but leave **`inputs`** empty.

```
curl -X POST \
  $url/deployment/api/deployments/$deployment_id/requests \
  -H "Authorization: Bearer $access_token" \
  -H 'Content-Type: application/json' \
  -d '{
    "actionId":"Deployment.ChangeLease",
    "inputs": {}
}' | jq "."
```

Use the request ID from the response to check the status of the request. The lease is changed when the response shows `"status":"SUCCESSFUL"`.

Example: Change the Lease on Your Deployment

Change the lease on your deployment with ID **`5551a299-8b67-45e3-909e-a638d11b0d9f`**.

Assign variables.

```
$ url='https://appliance.domain.com'
$ api_version='2020-08-25'
$ deployment_id='5551a299-8b67-45e3-909e-a638d11b0d9f'
```

List the actions available for your deployment.

```
$ curl -X GET \
  $url/deployment/api/deployments/$deployment_id/actions?apiVersion=$api_version \
  -H "Authorization: Bearer $access_token" | jq "."
```

A snippet of the response shows `Deployment.ChangeLease`.

```
...
{
    "id": "Deployment.ChangeLease",
    "name": "ChangeLease",
    "displayName": "Change Lease",
    "description": "Set a deployment's expiration date",
    "valid": true,
    "actionType": "RESOURCE_ACTION"
  }
...
```

Assign the action ID variable.

```
$ action_id='Deployment.ChangeLease'
```

To get the schema for your action, list the deployment actions by ID.

```
$ curl -X GET \
  $url/deployment/api/deployments/$deployment_id/actions/$action_id?apiVersion=$api_version \
  -H "Authorization: Bearer $access_token" | jq "."
```

A snippet of the response provides the schema for the lease expiration date.

```
...
      "properties": {
      "Deployment expires in": {
        "type": "string",
        "readOnly": true,
        "default": "9d 21h 27m"
      },
      "Lease Expiration Date": {
        "type": "string",
```

```
        "title": "Lease Expiration Date",
        "description": "The lease can be extended by up to 90d 0h 0m",
        "format": "date-time",
        "formatMinimum": "2021-02-12T21:47:00Z",
        "formatMaximum": "2021-05-20T19:15:00Z",
        "default": "2021-02-22T19:15:00Z"
...
```

Assign the lease expiry date variable in the propert format.

```
$ lease_expiry_date=2021-05-15T23:11:00Z
```

Submit a request to change the lease expiration.

```
$ curl -X POST \
  $url/deployment/api/deployments/$deployment_id/requests \
  -H "Authorization: Bearer $access_token" \
  -H 'Content-Type: application/json' \
  -d '{
    "actionId":"Deployment.ChangeLease",
    "inputs": {
        "Lease Expiration Date": "'"$lease_expiry_date"'"
    }
}' | jq "."
```

A snippet of the response shows request ID.

```
...
  "id": "6b9b5534-0d84-4f07-9941-5c3cc26f7e3b",
  "name": "Change Lease",
  "deploymentId": "5551a299-8b67-45e3-909e-a638d11b0d9f",
...
```

Assign the request ID variable.

```
$ request_id='6b9b5534-0d84-4f07-9941-5c3cc26f7e3b'
```

Check the status of the request.

```
$ curl -X GET \
  $url/deployment/api/requests/$request_id?apiVersion=$api_version \
  -H "Authorization: Bearer $access_token" | jq "."
```

A snippet of the response shows that the request was successful.

```
...
  "actionId": "Deployment.ChangeLease",
  "completedTasks": 1,
  "totalTasks": 1,
  "status": "SUCCESSFUL"
}
```

Delete Your Deployment

To delete a deployment, you use the Automation Service Broker Deployment API and if needed, the Automation Assembler IaaS API. You make a DELETE request to remove the deployment and clean up the associated resources from the cloud provider.

If using the Automation Service Broker Deployment service fails to delete the deployment, you can use the Automation Assembler IaaS service to force the deletion. The following procedure shows how to delete a deployment when both API services are used.

Prerequisites

- Verify that all general prerequisites and prerequisites for the Automation Service Broker Deployment service have been satisfied. See Chapter 3 Prerequisites for API Use Case Examples.
- Verify that you have the Automation Assembler administrator service role.
- Verify that you know the deployment that you want to delete.
- Assign an API version variable for the IaaS API.

```
api_version_iaas='2021-07-15'
```

Note The Automation Service Broker Deployment service and the Automation Assembler Infrastructure as a Service (IaaS) service have different API version values. You set the API version value for the Automation Service Broker Deployment service when you satisfied the general prerequisites.

Procedure

1 List all deployments.

```
curl -X GET "$url/deployment/api/deployments?apiVersion=$api_version" -H "Authorization:
Bearer $access_token" | jq "."
```

2 Examine the response to find the name and ID of the deployment that you want to delete.

3 Assign your deployment ID variable.

```
deployment_id='<your_deployment_id>'
```

4 Get a list of actions for the deployment.

```
curl -X GET \
  "$url/deployment/api/deployments/$deployment_id/actions?apiVersion=$api_version" \
  -H "Authorization: Bearer $access_token" | jq "."
```

5 Examine the response.

- Confirm that you see the action `"id": "Deployment.Delete"`.

- `"valid":true` indicates that the action is valid for the deployment.

6 Delete the deployment.

```
curl -X DELETE \
  "$url/deployment/api/deployments/$deployment_id?apiVersion=$api_version" \
  -H "Authorization: Bearer $access_token" | jq "."
```

7 Examine the response for the ID of the deletion request.

8 Assign the ID to the request ID variable.

```
request_id='<delete_request_id>'
```

9 Get the status of the deployment request.

```
curl -X GET \
  "$url/deployment/api/requests/$request_id?apiVersion=$api_version" \
  -H "Authorization: Bearer $access_token" | jq "."
```

10 Examine the response.

- If the response shows `"status": "SUCCESSFUL"`, then the delete request succeeded.
- If the response shows `"status": "FAILED"` then the delete request may have failed due to a dependency such as an unreleased IP in an AD account. If you are certain that you want to remove the deployment and all related resources, you can choose to ignore the deletion failures and submit a delete request that uses the Automation Assembler IaaS service.

11 (Optional) If you want to force deletion, use **`forceDelete=true`** in the Automation Assembler IaaS service request to delete the deployment.

```
curl -X DELETE \
  "$url/iaas/api/deployments/$deployment_id?forceDelete=true&apiVersion=$api_version_iaas"
\
  -H "Authorization: Bearer $access_token" | jq "."
```

12 (Optional) Repeat steps 7, 8, and 9.

13 (Optional) Examine the response to verify that `"status": "SUCCESSFUL"` appears.

Example: Delete Your Deployment

This example shows how to delete a deployment with the Automation Service Broker Deployment service and when that deletion fails, force the deletion with the Automation Assembler IaaS service.

Assign variables.

```
$ url='https://appliance.domain.com'
$ api_version='2020-08-25'
$ api_version_iaas='2021-07-15'
```

List all deployments.

```
curl -X GET "$url/deployment/api/deployments?apiVersion=$api_version" -H "Authorization:
Bearer $access_token" | jq "."
```

Examine the response to find the deployment that you want to delete.

```
    {
      "id": "164f2d4e-1755-491e-b0a0-583f0ed4ae3e",
      "name": "example_deployment",
      "description": "",
      ...
    },
```

Assign your deployment variable.

```
deployment_id='164f2d4e-1755-491e-b0a0-583f0ed4ae3e'
```

List the actions available for your deployment.

```
$ curl -X GET \
  "$url/deployment/api/deployments/$deployment_id/actions?apiVersion=$api_version" \
  -H "Authorization: Bearer $access_token" | jq "."
```

A snippet of the response shows the action `"id": "Deployment.Delete"` with `"valid": true` so you can delete the deployment.

```
...
{
    "id": "Deployment.Delete",
    "name": "Delete",
    "displayName": "Delete",
    "description": "Delete a deployment",
    "valid": true,
    "actionType": "RESOURCE_ACTION"
  }
...
```

Delete the deployment.

```
curl -X DELETE \
  "$url/deployment/api/deployments/$deployment_id?apiVersion=$api_version" \
  -H "Authorization: Bearer $access_token" | jq "."
```

A snippet of the response shows the ID which is the request ID.

```
...
  "id": "d9541db3-2806-42aa-bde0-fb870d114833",
  "name": "Delete",
  "requestedBy": "user@mycompany.com",
  "actionId": "Deployment.Delete",
  "deploymentId": "164f2d4e-1755-491e-b0a0-583f0ed4ae3e",
  "resourceIds": [
```

```
    ...
  ],
  "status": "PENDING",
...
```

Assign the request ID variable.

```
$ request_id='d9541db3-2806-42aa-bde0-fb870d114833'
```

Check the status of the request.

```
$ curl -X GET \
  "$url/deployment/api/requests/$request_id?apiVersion=$api_version" \
  -H "Authorization: Bearer $access_token" | jq "."
```

A snippet of the response shows that the request failed.

```
{
  "id": "d9541db3-2806-42aa-bde0-fb870d114833",
  "name": "Delete",
  "requestedBy": "user@mycompany.com",
  "actionId": "Deployment.Delete",
  "deploymentId": "164f2d4e-1755-491e-b0a0-583f0ed4ae3e",
  "resourceIds": [
    ...
  ],
  "status": "FAILED",
...
```

If you are certain that you want to delete the deployment, force the deletion using the Automation Assembler IaaS service.

```
curl -X DELETE \
  "$url/iaas/api/deployments/$deployment_id?forceDelete=true&apiVersion=$api_version_iaas" \
  -H "Authorization: Bearer $access_token" | jq "."
```

Examine the response to get the new request ID.

```
{
  "progress": 0,
  "status": "INPROGRESS",
  "name": "Delete",
  "id": "a1b674a2-09aa-4b14-9459-35dddddb9bbc1",
  "selfLink": "/iaas/api/request-tracker/a1b674a2-09aa-4b14-9459-35dddddb9bbc1"
...
```

Assign the new request ID.

```
$ new_request_id='a1b674a2-09aa-4b14-9459-35dddddb9bbc1'
```

Use the Automation Service Broker Deployment service to check the status of the request.

```
$ curl -X GET \
  "$url/deployment/api/requests/$new_request_id?apiVersion=$api_version" \
  -H "Authorization: Bearer $access_token" | jq "."
```

Examine the response to verify that the action is in progress.

```
{
  "id": "a1b674a2-09aa-4b14-9459-35dddddb9bbc1",
  "name": "Delete",
  "requestedBy": "user@mycompany.com",
  "actionId": "Deployment.Delete",
  "deploymentId": "164f2d4e-1755-491e-b0a0-583f0ed4ae3e",
  "resourceIds": [
  ...
  ],
  "status": "INPROGRESS",
...
```

Continue to check the status of the request.

```
$ curl -X GET \
  "$url/deployment/api/requests/$new_request_id?apiVersion=$api_version" \
  -H "Authorization: Bearer $access_token" | jq "."
```

The response shows when the deployment deletion request is successful.

```
{
  "id": "a1b674a2-09aa-4b14-9459-35dddddb9bbc1",
  "name": "Delete",
  "requestedBy": "user@mycompany.com",
  "actionId": "Deployment.Delete",
  "deploymentId": "164f2d4e-1755-491e-b0a0-583f0ed4ae3e",
  "resourceIds": [
  ...
  ],
  "status": "SUCCESSFUL",
...
```

Reconfigure Load Balancer

To reconfigure the load balancer in your deployment, you use the Deployment APIs to make a POST request with the ID of the load balancer to update.

Prerequisites

- Verify that all general prerequisites and prerequisites for the Automation Service Broker Deployment service have been satisfied. See Chapter 3 Prerequisites for API Use Case Examples.

- Verify that you have the ID of the deployment you want to reconfigure. See Deploy a Cloud Template with Contents Inline.
- Verify that you have the ID of the load balancer in your deployment. See Get Deployment Resource IDs.

Procedure

1 Assign your deployment ID variable.

```
deployment_id='<your_deployment_id>'
```

2 Assign your load balancer ID variable.

```
load_balancer_id='<your_load_balancer_id>'
```

3 Get a list of actions available for the load balancer in your deployment.

```
curl -X GET \
  $url/deployment/api/deployments/$deployment_id/resources/$load_balancer_id/actions?
apiVersion=$api_version \
  -H "Authorization: Bearer $access_token" | jq "."
```

4 Examine the response.

- Confirm that you see the action `"name": "LoadBalancer.Reconfigure"`.
- `"valid":true` indicates that the action is valid for the deployment resource.

5 Assign the action ID variable for the reconfigure action `"name": "LoadBalancer.Reconfigure"`.

```
reconfigure_action_id='Cloud.LoadBalancer.LoadBalancer.Reconfigure'
```

6 List the resource actions for the action ID .

```
curl -X GET \
  $url/deployment/api/deployments/$deployment_id/resources/$load_balancer_id/actions/
$reconfigure_action_id?apiVersion=$api_version \
  -H "Authorization: Bearer $access_token" | jq "."
```

7 Examine the response. The schema field shows the format of the input for an action on the load balancer resource.

8 Reconfigure the load balancer with input inline.

```
curl -X POST \
  $url/deployment/api/deployments/$deployment_id/resources/$load_balancer_id/requests \
  -H "Authorization: Bearer $access_token" \
  -H 'Content-Type: application/json' \
  -d '{
  "actionId": "Cloud.LoadBalancer.LoadBalancer.Reconfigure",
  "inputs": {
        "routes": [
```

```
          {
            "port": "81",
            "protocol": "TCP",
            "instancePort": "81",
            "instanceProtocol": "TCP",
            "healthCheckConfiguration": {
              "port": "81",
              "urlPath": "/index.html",
              "protocol": "HTTP",
              "timeoutSeconds": 5,
              "intervalSeconds": 60,
              "healthyThreshold": 2,
              "unhealthyThreshold": 5
            }
          }
        ]
  }
}' | jq "."
```

9 Examine the response and assign the request ID.

```
request_id='<your_request_id>'
```

10 Check the status of the request.

```
curl -X GET \
  $url/deployment/api/requests/$request_id?apiVersion=$api_version \
  -H "Authorization: Bearer $access_token" | jq "."
```

If the request is successful, the response shows `"status":"SUCCESSFUL"`.

Example: Reconfigure the Load Balancer in Your Deployment

For your deployment with ID **5551a299-8b67-45e3-909e-a638d11b0d9f**, reconfigure the load balancer with resource ID **d5b4569d-2234-4fc4-a594-45e6b0251588**.

Assign variables.

```
$ url='https://appliance.domain.com'
$ api_version='2020-08-25'
$ deployment_id='5551a299-8b67-45e3-909e-a638d11b0d9f'
```

Assign the load balancer ID.

```
$ load_balancer_id='d5b4569d-2234-4fc4-a594-45e6b0251588'
```

List the actions available for the load balancer resource.

```
$ curl -X GET \
  $url/deployment/api/deployments/$deployment_id/resources/$load_balancer_id/actions?
apiVersion=$api_version \
  -H "Authorization: Bearer $access_token" | jq "."
```

A snippet of the response shows `LoadBalancer.Reconfigure` action.

```
...
{
    "id": "Cloud.LoadBalancer.LoadBalancer.Reconfigure",
    "name": "LoadBalancer.Reconfigure",
    "displayName": "Reconfigure",
    "description": "Reconfigure Load Balancer",
    "valid": true,
    "actionType": "RESOURCE_ACTION"
  }
...
```

Assign the action ID variable.

```
$ reconfigure_action_id='Cloud.LoadBalancer.LoadBalancer.Reconfigure'
```

To get the schema for your action, list the resource actions by ID.

```
$ curl -X GET \
  $url/deployment/api/deployments/$deployment_id/resources/$load_balancer_id/actions/
$reconfigure_action_id?apiVersion=$api_version" \
  -H "Authorization: Bearer $access_token" | jq "."
```

A snippet of the response provides the schema for the load balancing action.

```
...
  "schema": {
    "type": "object",
    "title": "Reconfigure Load Balancer",
    "description": "Request schema for updating routes of load balancer resource",
    "properties": {
      "routes": {
        "type": "array",
        "items": {
          "type": "object",
          "properties": {
            "protocol": {
              "type": "string",
              "title": "Protocol",
              "description": "The communication protocol for an incoming request to the load
balancer. HTTP, HTTPS, or TCP.",
              "enum": [
                "HTTP",
                "HTTPS",
                "TCP"
              ]
            },
            "port": {
              "type": "string",
              "title": "Port",
              "description": "The listening port for an incoming request to the load
balancer.",
              "pattern": "^\\d+$"
```

```
            },
            "instanceProtocol": {
              "type": "string",
              "title": "Instance protocol",
              "description": "The communication protocol used between the load balancer and
the machines in the pool. HTTP, HTTPS, or TCP.",
              "enum": [
                "HTTP",
                "HTTPS",
                "TCP"
              ]
...
```

Submit a request to reconfigure the load balancer with new route properties.

```
$ curl -X POST \
  $url/deployment/api/deployments/$deployment_id/resources/$load_balancer_id/requests \
  -H "Authorization: Bearer $access_token" \
  -H 'Content-Type: application/json' \
  -d '{
  "actionId": "Cloud.LoadBalancer.LoadBalancer.Reconfigure",
  "inputs": {
        "routes": [
          {
            "port": "81",
            "protocol": "TCP",
            "instancePort": "81",
            "instanceProtocol": "TCP",
            "healthCheckConfiguration": {
              "port": "81",
              "urlPath": "/index.html",
              "protocol": "HTTP",
              "timeoutSeconds": 5,
              "intervalSeconds": 60,
              "healthyThreshold": 2,
              "unhealthyThreshold": 5
            }
          }
        ]
  }
}' | jq "."
```

A snippet of the response shows request ID.

```
...
  "id": "7342a348-65e0-4376-9472-94be56b928a9",
  "name": "Reconfigure",
  "deploymentId": "13c04d0a-fd81-4bcc-99b1-ac499fb1821d",
...
```

Assign the request ID variable.

```
$ request_id='342a348-65e0-4376-9472-94be56b928a9'
```

Check the status of the request.

```
$ curl -X GET \
  $url/deployment/api/requests/$request_id?apiVersion=$api_version" \
  -H "Authorization: Bearer $access_token" | jq "."
```

A snippet of the response shows that the request was successful.

```
...
  "actionId": "Cloud.LoadBalancer.LoadBalancer.Reconfigure",
  "completedTasks": 1,
  "totalTasks": 1,
  "status": "SUCCESSFUL",
}
```

Add a Disk to a Machine and Power It Off

To add a disk to a machine in your deployment, you use the Deployment APIs to make a POST request with the ID of the virtual machine to update. To power off the machine, you make a POST request and specify the action to perform.

Prerequisites

- Verify that all general prerequisites and prerequisites for the Automation Service Broker Deployment service have been satisfied. See Chapter 3 Prerequisites for API Use Case Examples.
- Verify that you have the ID of the deployment you want to reconfigure. See Deploy a Cloud Template with Contents Inline.
- Verify that you have the ID of the virtual machine in your deployment. See Get Deployment Resource IDs.

Procedure

1 Assign your deployment ID variable.

```
deployment_id='<your_deployment_id>'
```

2 Assign your virtual machine ID variable.

```
virtual_machine_id='<your_virtual_machine_id>'
```

3 Get a list of actions available for the virtual machine in your deployment.

```
curl -X GET \
  $url/deployment/api/deployments/$deployment_id/resources/$virtual_machine_id/actions?
apiVersion=$api_version \
  -H "Authorization: Bearer $access_token" | jq "."
```

4 Examine the response.

- Confirm that you see the action `Add.Disk` with `"valid":true`. Copy the value to assign to the add disk action ID.
- Confirm that you see the action `PowerOff` with `"valid":true`. Copy the value to assign to the power off action ID.

`"valid":true` indicates that each action is valid for the deployment resource.

5 Assign variables for the resources.

```
add_disk_action_id='<your_add_disk_id>'
poweroff_machine_action_id='<your_poweroff_action_id>'
```

6 List the resource actions for the add disk action ID.

```
curl -X GET \
  $url/deployment/api/deployments/$deployment_id/actions/$reconfigure_action_id?
apiVersion=$api_version \
  -H "Authorization: Bearer $access_token" | jq "."
```

7 Examine the response. The schema field shows the format of the input for an action on the virtual machine resource.

8 Attach a disk of size 1 GB to the machine.

```
curl -X POST \
  $url/deployment/api/deployments/$deployment_id/resources/$virtual_machine_id/requests \
  -H "Authorization: Bearer $access_token" \
  -H 'Content-Type: application/json' \
  -d '{
    "actionId":"Cloud.AWS.EC2.Instance.Add.Disk",
    "inputs":{
        "name":"disk1",
        "capacityGb":1,
        "type":"Cloud.Volume"
    }
}' | jq "."
```

9 Examine the response and assign the request ID.

```
request_id='<your_request_id>'
```

10 Check the status of the request.

```
curl -X GET \
  $url/deployment/api/requests/$request_id?apiVersion=$api_version \
  -H "Authorization: Bearer $access_token" | jq "."
```

If the request is successful, the response shows `"status":"SUCCESSFUL"`.

11 List the resource actions for the power off action ID.

```
curl -X GET \
  $url/deployment/api/deployments/$deployment_id/resources/$virtual_machine_id/actions/
$poweroff_machine_action_id?apiVersion=$api_version \
  -H "Authorization: Bearer $access_token" | jq "."
```

12 Examine the response. No schema field indicates that no **`inputs`** field is required for this action on the virtual machine resource.

13 Power off the machine.

```
curl -X POST \
  $url/deployment/api/deployments/$deployment_id/resources/$virtual_machine_id/requests \
  -H "Authorization: Bearer $access_token" \
  -H 'Content-Type: application/json' \
  -d '{
  "actionId":"Cloud.AWS.EC2.Instance.PowerOff"
}' | jq "."
```

14 Examine the response and assign the request ID.

```
request_id='<your_request_id>'
```

15 Check the status of the request.

```
curl -X GET \
  $url/deployment/api/requests/$request_id?apiVersion=$api_version \
  -H "Authorization: Bearer $access_token" | jq "."
```

If the request is successful, the response shows `"status":"SUCCESSFUL"`.

Example: Add a Disk and Power Off Your Virtual Machine

For your deployment with ID **`5551a299-8b67-45e3-909e-a638d11b0d9f`**, reconfigure the virtual machine with resource ID **`42f49781-1490-4a08-ae21-8baf383a72ac`** by adding a disk and powering it off.

Assign variables.

```
$ url='https://appliance.domain.com'
$ api_version='2020-08-25'
$ deployment_id='5551a299-8b67-45e3-909e-a638d11b0d9f'
```

Assign the virtual machine ID.

```
$ virtual_machine_id='42f49781-1490-4a08-ae21-8baf383a72ac'
```

List the actions available for the virtual machine resource.

```
$ curl -X GET \
  $url/deployment/api/deployments/$deployment_id/resources/$virtual_machine_id/actions?
```

```
apiVersion=$api_version \
  -H "Authorization: Bearer $access_token" | jq "."
```

A snippet of the response shows the actions to `Add.Disk` and `PowerOff`.

```
...
  {
    "id": "Cloud.AWS.EC2.Instance.Add.Disk",
    "name": "Add.Disk",
    "displayName": "Add Disk",
    "description": "Add a disk to the machine",
    "valid": true,
    "actionType": "RESOURCE_ACTION"
  },
...
  {
    "id": "Cloud.AWS.EC2.Instance.PowerOff",
    "name": "PowerOff",
    "displayName": "Power Off",
    "description": "Power off a machine",
    "valid": true,
    "actionType": "RESOURCE_ACTION"
  },
...
```

Assign the action ID variables to add a disk and power off the virtual machine.

```
$ add_disk_action_id='Cloud.AWS.EC2.Instance.Add.Disk'
$ power_off_action_id='Cloud.AWS.EC2.Instance.PowerOff'
```

Get the add disk action for the virtual machine resource.

```
$ curl -X GET \
  $url/deployment/api/deployments/$deployment_id/resources/$virtual_machine_id/actions/
$add_disk_action_id?apiVersion=$api_version \
  -H "Authorization: Bearer $access_token" | jq "."
```

A snippet of the response provides the schema to add a disk.

```
...
    "properties": {
      "name": {
        "type": "string",
        "title": "Name",
        "description": "Disk Name",
        "minLength": 1
      },
      "capacityGb": {
        "type": "integer",
        "title": "Size(GB)",
        "description": "Disk Capacity in GB",
        "minimum": 1
      },
      "type": {
```

```
        "type": "string",
        "title": "Type",
        "description": "Disk Resource Type.",
        "readOnly": true,
        "default": "Cloud.Volume"
      },
...
```

Follow the schema and submit a request to add a 1 GB disk to the virtual machine.

```
$ curl -X POST \
  $url/deployment/api/deployments/$deployment_id/resources/$virtual_machine_id/requests \
  -H "Authorization: Bearer $access_token" \
  -H 'Content-Type: application/json' \
  -d '{
    "actionId":"Cloud.AWS.EC2.Instance.Add.Disk",
    "inputs":{
        "name":"disk1",
        "capacityGb":1,
        "type":"Cloud.Volume"
    }
}' | jq "."
```

A snippet of the response shows request ID.

```
...
  "id": "17dec8d9-2e2a-4c29-9067-ce41c37be7a3",
  "name": "Add Disk",
  "deploymentId": "5551a299-8b67-45e3-909e-a638d11b0d9f",
...
```

Assign the request ID variable.

```
$ request_id='17dec8d9-2e2a-4c29-9067-ce41c37be7a3'
```

Check the status of the request.

```
$ curl -X GET \
  $url/deployment/api/requests/$request_id?apiVersion=$api_version \
  -H "Authorization: Bearer $access_token" | jq "."
```

A snippet of the response shows that the request was successful.

```
...
  "actionId": "Cloud.AWS.EC2.Instance.Add.Disk",
  "completedTasks": 3,
  "totalTasks": 3,
  "status": "SUCCESSFUL",
}
```

Get the power off action for the virtual machine resource.

```
$ curl -X GET \
  $url/deployment/api/deployments/$deployment_id/resources/$virtual_machine_id/actions/
```

```
$poweroff_machine_action_id?apiVersion=$api_version \
  -H "Authorization: Bearer $access_token" | jq "."
```

The complete response shows that there is no schema for the power off action.

```
{
  "id": "Cloud.AWS.EC2.Instance.PowerOff",
  "name": "PowerOff",
  "displayName": "Power Off",
  "description": "Power off a machine",
  "dependents": [
    "Provider_LoadBalancer_1"
  ],
  "valid": true,
  "actionType": "RESOURCE_ACTION"
}
```

Power off the virtual machine.

```
$ curl -X POST \
  $url/deployment/api/deployments/$deployment_id/resources/$virtual_machine_id/requests \
  -H "Authorization: Bearer $access_token" \
  -H 'Content-Type: application/json' \
  -d '{
  "actionId":"Cloud.AWS.EC2.Instance.PowerOff"
}' | jq "."
```

A snippet of the response shows request ID.

```
...
  "id": "ab7d3aec-f850-4b0e-9c1c-47378c182a00",
  "name": "Power Off",
  "deploymentId": "5551a299-8b67-45e3-909e-a638d11b0d9f",
...
```

Assign the request ID variable.

```
$ request_id='ab7d3aec-f850-4b0e-9c1c-47378c182a00'
```

Check the status of the request.

```
$ curl -X GET \
  $url/deployment/api/requests/$request_id?apiVersion=$api_version \
  -H "Authorization: Bearer $access_token" | jq "."
```

A snippet of the response shows that the request was successful.

```
...
  "actionId": "Cloud.AWS.EC2.Instance.PowerOff",
  "completedTasks": 1,
  "totalTasks": 1,
  "status": "SUCCESSFUL",
```

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

Viewing Billable Objects

As an Automation Assembler or Automation Service Broker administrator, you can use the Automation Service Broker Deployment APIs to view billable objects that are used in your organization.

Prerequisites for viewing billable objects

- Verify that all general prerequisites and prerequisites for the Automation Service Broker Deployment service have been satisfied. See Chapter 3 Prerequisites for API Use Case Examples.

How to get a summary of billable objects

The following request retrieves a categorized summary or total count of VMware Aria Automation billable objects in your organization.

```
curl -X GET \
$url/deployment/api/billing-metrics?apiVersion=$api_version \
-H 'Accept: application/json' \
-H "Authorization: Bearer $access_token"  | jq "."
```

The following sample response shows all billable objects.

```
{
  "billingMetrics": [
    {
      "id": "vSphereManagedVMCount",
      "displayName": "vSphere Managed VM Count",
      "value": 19
    },
    {
      "id": "vSphereCpuCount",
      "displayName": "vSphere CPU Count",
      "value": 20
    },
    {
      "id": "vSphereCpuCoreCount",
      "displayName": "vSphere CPU Core Count",
      "value": 52
    },
    {
      "id": "vmcManagedVMCount",
      "displayName": "VMC Managed VM Count",
      "value": 0
    },
    {
      "id": "vmcCpuCount",
```

```
      "displayName": "VMC CPU Count",
      "value": 0
    },
    {
      "id": "vmcCpuCoreCount",
      "displayName": "VMC CPU Core Count",
      "value": 0
    },
    {
      "id": "publicCloudManagedVMCount",
      "displayName": "Public Cloud Managed VM Count",
      "value": 58
    }
  ]
}
```

How to get information about billable objects

By applying the filter `billable=true`, the following request limits the amount of output from the call to include only the billable objects that are in your organization.

```
curl -X GET \
"$url/deployment/api/resources?apiVersion=$api_version&billable=true" \
-H 'Accept: application/json' \
-H "Authorization: Bearer $access_token" \
 | jq "."
```

The following sample response snippet provides the details about one of the billable objects.

```
{
  "content": [
    {
      "id": "9bc16a26-bd81-4660-9aa6-6eadfe72456d",
      "name": "Cloud_Machine_1-mcm1157065-230404373562",
      "type": "Cloud.AWS.EC2.Instance",
      "properties": {
        "hostName": "",
        "resourceId": "9bc16a26-bd81-4660-9aa6-6eadfe72456d",
        "externalLink": "https://us-east-1.console.aws.amazon.com/ec2/v2/home?region=us-
east-1#Instances:instanceId=i-0c7ec71b0aa26e080;sort=instanceState",
        "project": "7a90a6fb-f78d-4b91-9b3c-d88d3dd897fd",
        "storage": {
          "disks": [
            {
              "iops": "100",
              "name": "Cloud_Machine_1-mcm1157065-230404373562-boot-disk",
              "type": "SSD",
              "service": "ebs",
              "regionId": "us-east-1",
              "bootOrder": 1,
              "encrypted": false,
              "capacityGb": 30,
              "deviceName": "/dev/sda1",
              "deviceType": "ebs",
```

```
            "persistent": false,
            "snapshotId": "snap-00bbf435c29dcdf5c",
            "volumeType": "gp2",
            "endpointType": "aws",
            "resourceLink": "/resources/disks/3525c26f-30e9-4070-bb1d-a53430594f84",
            "SourceTaskLink": "/provisioning/resource-enumeration-tasks",
            "existingResource": "false"
          }
        ]
      },
      "networks": [
        {
          "id": "/resources/network-interfaces/25ff21a4-8f79-4d77-861f-03148e532823",
          "name": "public-subnet-us-east-1c",
          "tags": [],
          "address": "172.31.84.7",
          "assignment": "dynamic",
          "deviceIndex": 0,
          "mac_address": "12:f0:45:3a:bd:1f",
          "resourceName": "public-subnet-us-east-1c",
          "securityGroupNames": [
            "photon-model-sg"
          ]
        }
      ],
      "powerState": "OFF",
      "zone": "us-east-1c",
      "environmentName": "Amazon Web Services",
      "providerId": "i-0c7ec71b0aa26e080",
      "osType": "WINDOWS",
      "id": "/resources/compute/9bc16a26-bd81-4660-9aa6-6eadfe72456d",
      "cpuCount": 1,
      "isSimulate": "false",
      "image": "small",
      "totalMemoryMB": 1024,
      "componentType": "Cloud.AWS.EC2.Instance",
      "awsVpcId": "vpc-adf7f2d5",
      "imageId": "ami-0bad85cc4b10fde5e",
      "endpointType": "aws",
      "address": "",
      "endpointId": "f2a31ae2-6256-4d2e-b72e-4687d57ce246",
      "externalId": "i-0c7ec71b0aa26e080",
      "resourceName": "Cloud_Machine_1-mcm1157065-230404373562",
      "tags": [
        {
          "key": "ll"
        },
        {
          "key": "sdf"
        }
      ],
      "rootDeviceType": "ebs",
      "primaryMAC": "12:f0:45:3a:bd:1f",
      "flavor": "aws-flavor",
      "service": "ec2",
```

```
        "name": "Cloud_Machine_1",
        "flavorRef": "t2.micro",
        "accounts": [
          "aws-vaidehi",
          "aws-skarwa"
        ],
        "region": "us-east-1",
        "flavorMappingName": "aws-flavor",
        "account": "aws-vaidehi"
      },
      "createdAt": "2023-04-20T17:12:54.348099Z",
      "syncStatus": "SUCCESS",
      "origin": "DEPLOYED",
      "deploymentId": "f588a723-65f4-402f-b548-a13cdd1b74d0",
      "projectId": "7a90a6fb-f78d-4b91-9b3c-d88d3dd897fd",
      "orgId": "79395484-98e2-450e-b1ab-095d6dae71ef",
      "billable": true
    },
...
```


Onboarding virtual machines

To onboard machines that are not yet managed by an Automation Assembler project, you use the Onboarding APIs.

The onboarding process begins with creating an onboarding plan that locates the machines to be onboarded and creates a deployment for those machines when the plan runs. When you add deployments to the plan, you have the option of adding the deployment with a cloud template that allows all day 2 actions on the deployment.

Onboard machines as a single deployment

As an administrator, you can use the Onboarding APIs to onboard unmanaged machines as a single Automation Assembler deployment so that you can manage the machines. To onboard machines, you:

- Create an onboarding plan.
- Query for unmanaged machines.
- Create a bulk deployment that adds the unmanaged machines.
- Run the plan.

When you run the plan you can check the plan progress to see if the plan completes successfully and onboards the machines.

The following procedure shows how to onboard machines and create a deployment in VMware Aria Automation without using a cloud template. This is the fastest way to onboard machines. However, to run the **Update** day 2 action on the deployment after onboarding, you must create the deployment with a cloud template. See How do I add a cloud template to my onboarding plan.

Prerequisites

- Verify that all general prerequisites and prerequisites for the Onboarding service have been satisfied. See Chapter 3 Prerequisites for API Use Case Examples.
- Verify that you have the ID for the cloud account with cloud zones where the machines to be onboarded are located. See Adding Cloud Accounts and Create a Cloud Zone .
- Verify that you have the ID for a project with at least one user and with access to the cloud zones. See Create a Project to use in Automation Assembler

Procedure

1 Assign your project ID variable, for example:

```
project_id='4c4f8a47-d746-43f4-b88c-ea94ac8bd573'
```

2 Assign your cloud ID, for example:

```
cloud_account_id='0cbc8179-d202-4a30-9460-c25d2653a677'
```

3 Create your onboarding plan without placements.

```
curl -X POST \
    $url/relocation/onboarding/plan \
    -H "Authorization: Bearer $access_token" \
    -H 'Content-Type: application/json' \
    -d '{
      "name": "'<your_plan_name>'",
      "projectId": "'"$project_id"'",
      "endpointIds": [
          "'"$cloud_account_id"'"
      ],
      "usePlacements": false
  }' | jq "."
```

Note To create an onboarding plan with placements, set **`"usePlacements": true`**. With placement turned on, the onboarded workload considers resource limits defined in your project cloud zones or resource quota plicy and takes longer to run.

Examine the response to verify the status, the project ID, the cloud ID, and to get the documentSelfLink, for example:

```
{
    "status": "OK",
    "nextRefreshTimeMicros": 0,
    "refreshIntervalMicros": 0,
    "name": "Plan-782",
    "projectId": "4c4f8a47-d746-43f4-b88c-ea94ac8bd573",
    "endpointIds": [
        "0cbc8179-d202-4a30-9460-c25d2653a677"
    ]
```

```
    ...
    "documentSelfLink": "/relocation/onboarding/plan/d2010bcd-34f6-4fac-9d67-f7f4c6ec686a"
    ...
}
```

The value for the `documentSelfLink` is used for the `planLink` in subsequent steps.

4 Use the `planLink` to discover the compute resources representing unmanaged machines.

```
curl -X POST \
    $url/relocation/api/wo/query-unmanaged-machine \
    -H "Authorization: Bearer $access_token" \
    -H 'Content-Type: application/json' \
    -d '{
      "planLink": "/relocation/onboarding/plan/d2010bcd-34f6-4fac-9d67-f7f4c6ec686a"
      "expandFields": [
          "id",
          "documentSelfLink",
          "name",
          "powerState",
          "address",
          "creationTimeMicros",
          "expandedTags",
          "tagLinks",
          "endpointLinks"
      ],
      "optionExcludePlanMachines": true
  }' | jq "."
```

Examine the response to get the snippet for unmanaged machines with **`"powerState": "ON"`**, for example:

```
...
"/resources/compute/8bff887d-56fd-3752-bb99-d3f32a35abde": {
    "address": "10.10.10.17",
    "creationTimeMicros": 1717409843457000,
    "documentSelfLink": "/resources/compute/dd646fdc-e6db-3b16-a72e-d1c905f7a9c8",
    "endpointLinks": "/resources/endpoints/aa6a5445-e63a-4659-a277-00073a2645c8",
    "expandedTags": [],
    "id": "5017d984-aa61-e00e-dfa6-a986403c01b8",
    "name": "Machine-76",
    "powerState": "ON",
    "tagLinks": []
    }
"/resources/compute/9bc16a26-bd81-4660-9aa6-6eadfe72456d": {
    "address": "10.10.2.243",
    "creationTimeMicros": 1717409843457000,
    "documentSelfLink": "/resources/compute/8615bbfe-ea90-420a-a1cf-8048ba4271d4",
    "endpointLinks": "/resources/endpoints/2d57f1c4-fbe0-46ce-a061-3a5f1396f37b",
    "expandedTags": [],
    "id": "a6f324e4-101c-33f6-ac51-d9aaaa020123",
```

```
    "name": "Machine-25",
    "powerState": "ON",
    "tagLinks": []
    }...
```

5 Use the `planLink`, `documentSelfLink`, and `name` to create a single deployment for the unmanaged machines.

```
curl -X POST \
    $url/relocation/onboarding/task/create-deployment-bulk \
    -H "Authorization: Bearer $access_token" \
    -H 'Content-Type: application/json' \
    -d '{
      "planLink": "/relocation/onboarding/plan/d2010bcd-34f6-4fac-9d67-f7f4c6ec686a",
      "deployments": [
          {
              "name": "'<your_deployment_name>'",
              "resources": [
                  {
                      "link": "/resources/compute/dd646fdc-e6db-3b16-a72e-d1c905f7a9c8",
                      "name": "Machine-76"
                  },
                  {
                      "link": "/resources/compute/8615bbfe-ea90-420a-a1cf-8048ba4271d4",
                      "name": "Machine-25"
                  }
              ]
          }
      ]
  }' | jq "."
```

The response shows that the discovered machines have been assigned to the deployment.

```
{
    "planLink": "/relocation/onboarding/plan/d2010bcd-34f6-4fac-9d67-f7f4c6ec686a",
    "deployments": [
        {
            "planLink": "/relocation/onboarding/plan/d2010bcd-34f6-4fac-9d67-f7f4c6ec686a",
            "name": "Deployment-562",
            "resources": [
               {
                     "link": "/resources/compute/dd646fdc-e6db-3b16-a72e-d1c905f7a9c8",
                     "name": "Machine-76"
               },
               {
                   "link": "/resources/compute/8615bbfe-ea90-420a-a1cf-8048ba4271d4",
                   "name": "Machine-25"
               }
            ],
            "tenantLink": "/core/tenants/dc96003b28356c75",
            "documentVersion": 0,
            "documentUpdateTimeMicros": 0,
            "documentExpirationTimeMicros": 0
        }
```

```
    ],
    "tenantLink": "/core/tenants/dc96003b28356c75",
    "documentVersion": 0,
    "documentUpdateTimeMicros": 0,
    "documentExpirationTimeMicros": 0
}
```

6 Use the `planLink` to run the plan and onboard the machines.

```
curl -X POST \
    $url/relocation/api/wo/execute-plan \
    -H "Authorization: Bearer $access_token" \
    -H 'Content-Type: application/json' \
    -d '{
      "planLink": "/relocation/onboarding/plan/d2010bcd-34f6-4fac-9d67-f7f4c6ec686a"
  }' | jq "."
```

7 Use the `planLink` to the check the run.

```
curl -X GET \
    $url/relocation/onboarding/plan/d2010bcd-34f6-4fac-9d67-f7f4c6ec686a \
    -H "Authorization: Bearer $access_token" \
    -H 'Content-Type: application/json'
 | jq "."
```

The status in the response changes as the plan moves through `executing` or `updating`. If the run fails, the status shows `"status": "error"`.

```
{
    "status": "OK",
    "nextRefreshTimeMicros": 0,
    "refreshIntervalMicros": 0,
    "name": "Plan-782",
    "projectId": "4c4f8a47-d746-43f4-b88c-ea94ac8bd573",
    "endpointIds": [
        "0cbc8179-d202-4a30-9460-c25d2653a677"
    ],
    "createdBy": "example_admin@example_company.com",
    "enableExtensibilityEvents": true,
    "organizationId": "174adb59-8132-46f0-9cd8-2b2361e9cb2c",
    "customProperties": {},
    "usePlacements": false,
    "isQuick": false,
    "tenantLink": "/core/tenants/dc96003b28356c75",
    "documentVersion": 0,
    "documentKind":
"com:vmware:relocation:services:onboarding:plan:OnboardingPlanService:OnboardingPlanState",
    "documentSelfLink": "/relocation/onboarding/plan/d2010bcd-34f6-4fac-9d67-f7f4c6ec686a",
    "documentUpdateTimeMicros": 1718190629415000,
    "documentUpdateAction": "POST",
    "documentExpirationTimeMicros": 0,
    "documentAuthPrincipalLink": "/core/authz/users/example_admin@example_company.com"
}
```

How do I add a cloud template to my onboarding plan

When creating an onboarding plan, you can attach an existing VMware cloud template (VCT) to the deployment. The cloud template allows you to iteratively update the deployment and avoid the need to delete onboarded resources and create new ones while gradually scaling out your deployment.

With a cloud template attached to your deployment, you can run all day 2 actions including the **Update** day 2 action provided that the YAML of the selected cloud template accepts user input.

Note You can choose any cloud template but depending on the cloud template details, an **Update** day 2 action might delete and re-create original resources.

By attaching a cloud template with resource mapping, you can create an onboarding deployment that maps VMs to template resources in the same way as the deployment that was originally provisioned in VMware Aria Automation using the same cloud template. The only restriction is that the onboarding deployment must have the same number of machines as the selected cloud template and the admin must specify the VM to template resource mapping.

A deployment that uses a cloud template with resource mapping includes the **Update** day 2 action if the selected template accepts user input and the update only applies changes to existing resources. It will not re-create the original resources unless there are major changes between the current version of cloud template and original deployment. Onboarded deployments can also be iteratively updated without deletion and re-creation of the onboarded machines.

Prerequisites for adding deployments with a cloud template

- Verify that all general prerequisites and prerequisites for the Onboarding service have been satisfied. See Chapter 3 Prerequisites for API Use Case Examples.
- Verify that you have the `planLink` for an existing onboarding plan. For example, to list onboarding plans use **`GET /relocation/api/onboarding/plan`**. See Onboard machines as a single deployment.
- Verify that you have the IDs and names of the resources that you are planning to onboard. For example, to list unmanaged machines, use **`POST /relocation/api/wo/query-unmanaged-machine`**. See Onboard machines as a single deployment.
- Verify that the VMware cloud template that you plan to attach exists. For example, to list cloud templates use **`GET /blueprint/api/blueprints`**. See Create and Update a Cloud Template.
- If adding a cloud template with mapping, verify that you have the correct template component name to link to each machine. For example:
  - To get template component names, use **`GET /blueprint/api/blueprints/<blueprint_id>`** where *blueprint_id* is the ID for your cloud template. In the response, the blueprint **`content`** field lists the machine component names.

- To list unmanaged machines, use **`POST /relocation/api/wo/query-unmanaged-machine`**.

Example payloads include the following input:

- planLink for existing onboarding plan

```
"planLink": "/relocation/onboarding/plan/d2010bcd-34f6-4fac-9d67-f7f4c6ec686a"
```

- Template to attach if onboarding with template but without mapping

```
"name": "singleMachines"
"link": "/blueprint/api/blueprints/b79612a9-6e2f-4b48-a775-88eb7676dc2c"
```

- Template to attach if onboarding with template and with mapping

```
      "template": {
        "name": "vc01",
        "link": "/blueprint/api/blueprints/74bbcdae-b945-4a95-ac86-8bed0a75e2c2",
        "components": {
          "/resources/compute/bf07d26a-38cd-392e-b3d6-0004cf36168a":
"Cloud_vSphere_Machine_1",
          "/resources/compute/cc628c21-5531-414d-a09e-5f35916ce689":
"Cloud_vSphere_Machine_2"
        }
```

How do I add a deployment with a cloud template

The following example adds a deployment with the template **`singleMachine`** to bring the VMs **`East-centos-small-000011`** and **`est`** under management.

```
curl -X POST \
    $url/relocation/onboarding/task/create-deployment-bulk \
    -H "Authorization: Bearer $access_token" \
    -H 'Content-Type: application/json' \
    -d '{
       "deployments": [
         {
           "resources": [
             {
               "link": "/resources/compute/b82870f1-575b-308b-ac2e-874fad85703b",
               "name": "East-centos-small-000011",
               "tagLinks": []
             },
             {
               "link": "/resources/compute/e12fefb6-2d0a-3ee1-bb7f-1ceceb3d1ca6",
               "name": "est",
               "tagLinks": []
             }
           ]
           "template": {
             "name": "singleMachine",
```

```
              "link": "/blueprint/api/blueprints/b79612a9-6e2f-4b48-a775-88eb7676dc2c"
            }
          }
        ],
        "planLink": "/relocation/onboarding/plan/d2010bcd-34f6-4fac-9d67-f7f4c6ec686a"
}' | jq "."
```

How do I add a deployment with a cloud template and mapping

The following example adds a deployment with the template **vc01** to bring:

- VMs **VM-20** and **VM-21** under management with:
  - **VM-20** mapped to **Cloud_vSphere_Machine_1** in the template.
  - **VM-21** mapped to **Cloud_vSphere_Machine_2** in the template.
- VMs **VM-0014** and **VM-0015** under management with:
  - **VM-0014** mapped to **Cloud_vSphere_Machine_1** in the template.
  - **VM-0015** mapped to **Cloud_vSphere_Machine_2** in the template.

```
curl -X POST \
    $url/relocation/onboarding/task/create-deployment-bulk \
    -H "Authorization: Bearer $access_token" \
    -H 'Content-Type: application/json' \
    -d '{
       "deployments": [
         {
       "template": {
         "name": "vc01",
         "link": "/blueprint/api/blueprints/74bbcdae-b945-4a95-ac86-8bed0a75e2c2",
         "components": {
           "/resources/compute/bf07d26a-38cd-392e-b3d6-0004cf36168a":
"Cloud_vSphere_Machine_1",
           "/resources/compute/cc628c21-5531-414d-a09e-5f35916ce689":
"Cloud_vSphere_Machine_2"
         }
       },
       "resources": [
         {
           "link": "/resources/compute/bf07d26a-38cd-392e-b3d6-0004cf36168a",
           "name": "VM-20",
           "tagLinks": []
         },
         {
           "link": "/resources/compute/cc628c21-5531-414d-a09e-5f35916ce689",
           "name": "VM-21",
           "tagLinks": []
         }
       ]
     },
     {
       "template": {
         "name": "vc01",
```

```
          "link": "/blueprint/api/blueprints/74bbcdae-b945-4a95-ac86-8bed0a75e2c2",
          "components": {
            "/resources/compute/2ce64191-867e-3bb2-9017-6f5aa5d3a3c2":
"Cloud_vSphere_Machine_1",
            "/resources/compute/c4f46940-e4e1-45fb-bd30-58794398a50c":
"Cloud_vSphere_Machine_2"
          }
        },
        "resources": [
          {
            "link": "/resources/compute/2ce64191-867e-3bb2-9017-6f5aa5d3a3c2",
            "name": "VM-0014",
            "tagLinks": []
          },
          {
            "link": "/resources/compute/c4f46940-e4e1-45fb-bd30-58794398a50c",
            "name": "VM-0015",
            "tagLinks": []
          }
        ]
      }
    ],
    "planLink": "/relocation/onboarding/plan/d2010bcd-34f6-4fac-9d67-f7f4c6ec686a"
}' | jq "."
```

Working with Pipelines

11

As a Automation Pipelines administrator, you can use the Automation Pipelines APIs to model your software release process. The following API examples show how to add an endpoint to use in a pipeline task, run the pipeline, and verify results.

Read the following topics next:

- Create an Endpoint
- Create and Enable a Pipeline
- Run and Monitor your Pipeline

Create an Endpoint

To create an endpoint, you make a POST request with the endpoint properties. Then you use the ID of the endpoint created to validate it.

The following procedure shows how to create a Jenkins endpoint to use in your pipeline. To create a Jenkins endpoint, you must provide the URL of the Jenkins server and the admin password. Before using it in a pipeline, you validate the endpoint to verify that it can connect to the Jenkins server.

Prerequisites

- Verify that all general prerequisites and prerequisites for the Automation Pipelines service have been satisfied. See Chapter 3 Prerequisites for API Use Case Examples.
- Verify that you have the ID for the project that you want to use for your endpoint. See Create a Project with the Project Service API.

Procedure

1 Assign the project ID variable.

```
project_id='<your_project_id>'
```

2 Create a Jenkins endpoint.

```
curl -X POST \
  $url/codestream/api/endpoints?apiVersion=$api_version \
  -H 'Content-Type: application/json' \
```

```
  -H "Authorization: Bearer $access_token"  \
  -d '{
    "name": "<your_endpoint_name>",
    "description": "",
    "isRestricted": false,
    "properties": {
        "url": "<your_Jenkins_server_URL>",
        "username": "admin",
        "password": "<admin_passwd_for_Jenkins_server>",
        "folderPath": "",
        "pollInterval": 2,
        "retryCount": 2,
        "retryWaitSeconds": 3
    },
    "type": "jenkins",
    "project": "'$project_id'"
}' | jq "."
```

3 Examine the response and assign the endpoint ID variable.

```
endpoint_id='<your_endpoint_id>'
```

4 To verify that the endpoint can connect to the Jenkins server, validate the endpoint.

```
$ curl -X POST \
  $url/codestream/api/endpoint-validation?apiVersion=$api_version \
  -H 'Content-Type: application/json' \
  -H "Authorization: Bearer $access_token"  \
  -d '{
    "project": "'$project_id'",
    "kind": "ENDPOINT",
    "id": "'$endpoint_id'",
    "name": "<your_endpoint_name>",
    "type": "jenkins",
    "properties": {
        "url": "<your_Jenkins_server_URL>",
        "username": "admin",
        "password": "<admin_passwd_for_Jenkins_server>",
        "folderPath": "",
        "pollInterval": 2,
        "retryCount": 2,
        "retryWaitSeconds": 3
    }
} | jq "."
```

Examine the response to verify that the endpoint is valid.

Example: Create a Jenkins endpoint

Create a Jenkins endpoint named **`jenkins-example`**.

Assign variables.

```
$ url='https://appliance.domain.com'
$ api_version='2019-10-17'
$ project_id='MyProject1'
```

Create the Jenkins endpoint.

```
$ curl -X POST \
  $url/codestream/api/endpoints?apiVersion=$api_version \
  -H 'Content-Type: application/json' \
  -H "Authorization: Bearer $access_token"  \
  -d '{
    "name": "jenkins-example",
    "description": "",
    "isRestricted": false,
    "properties": {
        "url": "http://example-jenkins-server.mycompany.com:8080",
        "username": "admin",
        "password": "1146Examplea3eJenkin5d004Pa55word",
        "folderPath": "",
        "pollInterval": 2,
        "retryCount": 2,
        "retryWaitSeconds": 3
    },
    "type": "jenkins",
    "project": "'$project_id'"
}' | jq "."
```

The response from your request shows the endpoint ID.

```
{
    "project": "MyProject1",
    "kind": "ENDPOINT",
    "id": "85723b0b-a819-435e-8d71-f8f834cdbaa2",
    "name": "jenkins-example",
    "description": "",
    "updatedBy": "adminuser@mycompany.com",
    "createdAt": "2022-08-03T08:57:10.033+0000",
    "updatedAt": "2022-11-04T11:14:49.315+0000",
    "_link": "/codestream/api/endpoints/85723b0b-a819-435e-8d71-f8f834cdbaa2",
...
}
```

Assign the endpoint ID variable.

```
$ endpoint_id='85723b0b-a819-435e-8d71-f8f834cdbaa2'
```

Validate the endpoint.

```
$ curl -X POST \
  $url/codestream/api/endpoint-validation?apiVersion=$api_version \
  -H 'Content-Type: application/json' \
  -H "Authorization: Bearer $access_token"  \
```

```
  -d '{
    "project": "'$project_id'",
    "kind": "ENDPOINT",
    "id": "'$endpoint_id'",
    "name": "jenkins-example",
    "type": "jenkins",
    "properties": {
        "url": "http://example-jenkins-server.mycompany.com:8080",
        "username": "admin",
        "password": "1146Examplea3eJenkin5d004Pa55word",
        "folderPath": "",
        "pollInterval": 2,
        "retryCount": 2,
        "retryWaitSeconds": 3
    }
} | jq "."
```

The status message in the response verifies that the endpoint is valid.

```
{
    "output": {},
    "status": "COMPLETED",
    "statusMessage": "Valid Jenkins Endpoint",
    "duration": 0
}
```

What to do next

Create a pipeline with a Jenkins task that uses the endpoint.

Create and Enable a Pipeline

To create a pipeline, you make a POST request and provide the endpoint ID. Then you use the ID of the pipeline created in a PATCH request to enable it.

The following procedure shows how to create a pipeline with a Jenkins task using the Jenkins endpoint that you created. Then you enable the pipeline to run.

Prerequisites

- Verify that all general prerequisites and prerequisites for the Automation Pipelines service have been satisfied. See Chapter 3 Prerequisites for API Use Case Examples.
- Verify that you have the ID for the project that you used for your endpoint. See Create a Project with the Project Service API.
- Verify that you know the name of the Jenkins endpoint that you created. See Create an Endpoint.

Procedure

1 Assign the project ID variable.

```
project_id='<your_project_id>'
```

2 Assign the endpoint name variable.

```
endpoint_name="'<your_endpoint_name>'"
```

3 Create the pipeline with a Jenkins task.

- For job, provide the job on the Jenkins server that your pipeline will run.
- For parameters, provide the parameters that will be passed to the job.

```
curl -X POST \
  $url/codestream/api/pipelines?apiVersion=$api_version \
  -H 'Content-Type: application/json' \
  -H "Authorization: Bearer $access_token"  \
  -d '{
    "project": "'$project_id'",
    "kind": "PIPELINE",
    "name": "'<your_pipeline_name>'",
    "concurrency": 10,
    "stageOrder": ["Stage0"],
    "stages": {
        "Stage0": {
            "taskOrder": ["Task0"],
            "tags": [],
            "tasks": {
                "Task0": {
                    "type": "Jenkins",
                    "ignoreFailure": false,
                    "preCondition": "",
                    "input": {
                        "job": "'<your_job_name>",
                        "jobFolder": "",
                        "parameters": {
                            "'<your_parameters>'": ""
                        }
                    },
                    "endpoints": {
                        "jenkinsServer": "'$endpoint_name'"
                    },
                    "tags": [],
                    "_configured": true
                }
            }
        }
    }
}' | jq "."
```

A snippet of the response shows the pipeline ID with the pipeline disabled. Assign a variable for the pipeline ID.

```
pipeline_id='<your_pipeline_id>'
```

4 Enable the pipeline.

```
curl -X PATCH \
  $url/codestream/api/pipelines/$pipeline_id?apiVersion=$api_version \
  -H 'Content-Type: application/json' \
  -H "Authorization: Bearer $access_token"  \
  -d '{"state": "ENABLED"}' | jq "."
```

The response shows the pipeline state changed to `"state": "ENABLED"`.

Example: Create and Enable a Pipeline

Using the endpoint named **`jenkins-example`** that you created, create a pipeline named **`jenkinspipeline`** with a Jenkins task.

Assign variables.

```
$ url='https://appliance.domain.com'
$ api_version='2019-10-17'
$ project_id='MyProject1'
$ endpoint_name='jenkins-example'
```

Create the pipeline with a Jenkins task.

```
$ curl -X POST \
  $url/codestream/api/pipelines?apiVersion=$api_version \
  -H 'Content-Type: application/json' \
  -H "Authorization: Bearer $access_token"  \
  -d '{
    "project": "'$project_id'",
    "kind": "PIPELINE",
    "name": "jenkinspipeline",
    "concurrency": 10,
    "stageOrder": ["Stage0"],
    "stages": {
        "Stage0": {
            "taskOrder": ["Task0"],
            "tags": [],
            "tasks": {
                "Task0": {
                    "type": "Jenkins",
                    "ignoreFailure": false,
                    "preCondition": "",
                    "input": {
                        "job": "Build-DemoApp",
                        "jobFolder": "",
                        "parameters": {
                            "vRCSTestExecutionId": ""
                        }
```

```
                    },
                    "endpoints": {
                        "jenkinsServer": "'$endpoint_name'"
                    },
                    "tags": [],
                    "_configured": true
                }
            }
        }
    }
}' | jq "."
```

A snippet of the response shows the pipeline ID and shows the state of the pipeline as disabled.

```
{
    "project": "MyProject1",
    "kind": "PIPELINE",
    "id": "2677aa61-578a-4465-a653-a3c787fed3be",
    "name": "jenkinspipeline",
    "createdBy": "adminuser@mycompany.com",
    "updatedBy": "adminuser@mycompany.com",
    "createdAt": "2022-11-04T11:20:09.905+0000",
    "updatedAt": "2022-11-04T11:23:40.971+0000",
    "_link": "/codestream/api/pipelines/2677aa61-578a-4465-a653-a3c787fed3be",
       ...
    "rollbacks": [],
    "tags": [],
    "state": "DISABLED"
}
```

Assign a variable for the pipeline ID.

```
$ pipeline_id="2677aa61-578a-4465-a653-a3c787fed3be"
```

Enable the pipeline.

```
$ curl -X PATCH \
  $url/codestream/api/pipelines/$pipeline_id?apiVersion=$api_version \
  -H 'Content-Type: application/json' \
  -H "Authorization: Bearer $access_token"  \
  -d '{"state": "ENABLED"}' | jq "."
```

A snippet of the response shows the state of the pipeline is enabled.

```
{
    "project": "MyProject1",
    "kind": "PIPELINE",
    "id": "2677aa61-578a-4465-a653-a3c787fed3be",
    "name": "jenkinspipeline",
    "createdBy": "adminuser@mycompany.com",
    "updatedBy": "adminuser@mycompany.com",
    "createdAt": "2022-11-04T11:20:09.905+0000",
    "updatedAt": "2022-11-04T11:24:50.693+0000",
    "_link": "/codestream/api/pipelines/2677aa61-578a-4465-a653-a3c787fed3be",
```

```
        ...
    "rollbacks": [],
    "tags": [],
    "state": "ENABLED"
}
```

What to do next

Run your pipeline.

Run and Monitor your Pipeline

To run your pipeline and verify that it completes successfully, you make a POST request with the pipeline ID and monitor the run.

Prerequisites

- Verify that all general prerequisites and prerequisites for the Automation Pipelines service have been satisfied. See Chapter 3 Prerequisites for API Use Case Examples.
- Verify that you have the ID for the pipeline that you created. See Create and Enable a Pipeline.

Procedure

1 Assign the pipeline ID variable.

```
pipeline_id='<your_pipeline_id>'
```

2 Run your pipeline.

```
curl -X POST \
  $url/codestream/api/pipelines/$pipeline_id/executions?apiVersion=$api_version \
  -H 'Content-Type: application/json' \
  -H "Authorization: Bearer $access_token"  \
  -d '{
    "comments":"",
    "input":{}
}' | jq "."
```

3 Examine the response and assign the execution ID variable.

```
execution_id='<your_execution_id>'
```

4 Monitor the run.

```
$ curl -X GET $url/codestream/api/executions/$execution_id?apiVersion=$api_version -H
'Content-Type: application/json' -H "Authorization: Bearer $access_token"  | jq "."
```

Continue to monitor the pipeline activity until the response shows:

```
...
"status": "COMPLETED",
"statusMessage": "Execution Completed."
...
```

Example: Run and Monitor your Pipeline

Using the ID for the pipeline named **`jenkinspipeline`** that you created, run and monitor the pipeline.

Assign variables.

```
$ url='https://appliance.domain.com'
$ api_version='2019-10-17'
$ pipeline_id='2677aa61-578a-4465-a653-a3c787fed3be'
```

Run your pipeline.

```
$ curl -X POST \
  $url/codestream/api/pipelines/$pipeline_id/executions?apiVersion=$api_version \
  -H 'Content-Type: application/json' \
  -H "Authorization: Bearer $access_token"  \
  -d '{
    "comments":"",
    "input":{}
}' | jq "."
```

The response from your request shows the execution ID.

```
{
    "executionId": "6fc8e571-418e-4b07-b869-e661fdbdf29f",
    "executionLink": "/codestream/api/executions/6fc8e571-418e-4b07-b869-e661fdbdf29f",
    "executionIndex": 0
}
```

Assign the execution ID variable.

```
$ execution_id='6fc8e571-418e-4b07-b869-e661fdbdf29f'
```

Monitor the run.

```
$ curl -X GET $url/codestream/api/executions/$execution_id?apiVersion=$api_version -H
'Content-Type: application/json' -H "Authorization: Bearer $access_token"  | jq "."
```

Continue to monitor the pipeline activity until the response shows the task and run completed successfully.

```
{
    "project": "MyProject1",
    "id": "6fc8e571-418e-4b07-b869-e661fdbdf29f",
    "name": "jenkinspipeline",
```

```
    "updatedBy": "adminuser@mycompany.com",
    "updatedAt": "2022-11-04 11:26:25.283",
    "_link": "/codestream/api/executions/6fc8e571-418e-4b07-b869-e661fdbdf29f",
       ...
    "stageOrder": ["Stage0"],
    "stages": {
        "Stage0": {
            "status": "COMPLETED",
            "statusMessage": "COMPLETED",
            "taskOrder": ["Task0"],
            "tasks": {
                "Task0": {
                    "type": "Jenkins",
                    "name": "Task0",
                    "id": "6fc8e571-418e-4b07-b869-e661fdbdf29f~0.0.0",
                                  ...
                    },
                    "status": "COMPLETED",
                    "statusMessage": "Jenkins task completed successfully.",
                    "_durationInMicros": 35915000,
                    "_startTime": 1667561149181000,
                    "_endTime": 1667561185160000
                }
            },
            "_durationInMicros": 36000000,
            "_startTime": 1667561149160000,
            "_endTime": 1667561185223000,
            "notifications": []
        }
    },
    "status": "COMPLETED",
    "statusMessage": "Execution Completed.",
...
```