

# vRealize Automation API Programming Guide

13 APRIL 2021

vRealize Automation 8.4

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

<https://docs.vmware.com/>

**VMware, Inc.**  
3401 Hillview Ave.  
Palo Alto, CA 94304  
[www.vmware.com](http://www.vmware.com)

Copyright © 2021 VMware, Inc. All rights reserved. [Copyright and trademark information.](#)

# Contents

- 1** What are the vRealize Automation APIs and how do I use them 5
  
- 2** Getting Your Authentication Token 9
  - Get Your Access Token 9
  - Verify User Roles 11
  
- 3** Onboarding a Team 14
  - Adding Cloud Accounts 15
    - Add an AWS Cloud Account 15
    - Add a vSphere Cloud Account 17
    - Add an NSX-T or NSX-V Cloud Account 22
    - Add a VMware Managed Cloud Account 25
    - Add an Azure Cloud Account 28
    - Add a Google Cloud Platform Cloud Account 31
  - Create a Cloud Zone 33
  - Create a Project 34
    - Add Users to Your Project 37
  - Add a Cloud Zone to Your Project 40
  - Create Flavor Mappings 43
  - Create Image Mappings 49
  - Create Network Profiles 53
    - Create a Network Profile with Isolation 56
    - Create a Network Profile with Security Groups 61
  - Creating Storage Profiles 63
    - Create an AWS Storage Profile 64
    - Create a vSphere Storage Profile 67
    - Create a vSphere Storage Profile for a First Class Disk 71
    - Create an Azure Storage Profile 75
  - Creating and Using a First Class Disk 79
    - Create a First Class Disk 79
    - Attach a First Class Disk 82
    - Manage First Class Disk Snapshots 85
  - Specify SCSI disk placement 90
  - Update the Custom Properties of a Machine 95
  
- 4** Querying with the IaaS APIs 97
  - Using Pagination and Count 97
  - Filtering Resources by Region ID 98

[Filtering Operations for Projects](#) 104

[Filtering for Machine Status](#) 106

## **5** Protecting Sensitive Data 108

[How to provision a machine with sensitive data](#) 108

[Properties that Support Encryption](#) 111

## **6** Working with Blueprints/Cloud Templates and Deployments 113

[Create and Update a Blueprint](#) 114

[Version and Release a Blueprint to a vRealize Automation Service Broker Catalog](#) 118

[Remove a Blueprint Version from a vRealize Automation Service Broker Catalog](#) 120

[Test Your Blueprint Deployment](#) 122

[Deploy Your Blueprint](#) 124

[Look up Deployment Details](#) 128

## **7** Requesting a Deployment from a Catalog Item 131

[Create a Catalog Source and List Discovered Items](#) 132

[Create Entitlements](#) 135

[Request Deployment](#) 138

[Create a Lease Policy](#) 142

## **8** Managing Your Deployment 144

[Deploy a Blueprint with Contents Inline](#) 145

[Change the Lease on Your Deployment](#) 148

[Get Deployment Resource IDs](#) 152

[Reconfigure Load Balancer](#) 153

[Add a Disk to a Machine and Power It Off](#) 158

# What are the vRealize Automation APIs and how do I use them

# 1

As a VMware vRealize<sup>®</sup> Automation<sup>™</sup> user or customer, you can perform vRealize Automation Cloud Assembly, vRealize Automation Service Broker, and vRealize Automation Code Stream functions programmatically by using REST API service calls.

## API Services

vRealize Automation includes the following APIs. API documentation is available with the product or from the VMware API Explorer at <https://code.vmware.com/apis/vrealize-automation>. To access all Swagger documents from a single landing page, go to <https://<vRA-HOSTNAME>/automation-ui/api-docs> where *vRA-HOSTNAME* is the FQDN of your vRealize Automation appliance.

Table 1-1. vRealize Automation

Main Service	Service Name and Description
Cloud Assembly	<b>ABX</b> Create or manage actions and their versions. Execute actions and flows.
	<b>Approvals</b> Enforce policies that control required approvals for a deployment or Day 2 action before the request is provisioned.
	<b>Blueprint</b> Create, validate, and provision blueprints. <b>Note</b> Blueprints in the API are VMware Cloud Templates in the product.
	<b>CMX</b> When using Kubernetes with vRealize Automation , deploy and manage Kubernetes clusters and namespaces.

Table 1-1. vRealize Automation (continued)

Main Service	Service Name and Description
	<p><b>Content Gateway</b></p> <p>Connect to your infrastructure as code content in external content sources, such as Source Code Management providers and VMware Marketplace.</p>
	<p><b>Custom Forms</b></p> <p>Define dynamic form rendering and customization behavior in vRealize Automation Service Broker and vRealize Automation Cloud Assembly.</p>
	<p><b>Deployment</b></p> <p>Access deployment objects and platforms or blueprints that have been deployed into the system.</p> <p><b>Note</b> Blueprints in the API are VMware Cloud Templates in the product.</p>
	<p><b>Infrastructure as a Service (IaaS)</b></p> <p>Perform infrastructure setup tasks, including validation, and provisioning of resources in an iterative manner.</p>
	<p><b>Migration Assistant</b></p> <p>Run assessments and access migration services. Supports migration of content for vRealize Automation 7.4 or later, and for NSX-V to NSX-T.</p>
	<p><b>Onboarding</b></p> <p>Define policies and plans to bring existing VMs from any cloud under management.</p>
	<p><b>Projects</b></p> <p>Provide visibility and isolation of provisioned resources for users with a project role.</p>
Service Broker	<p><b>Catalog</b></p> <p>Access vRealize Automation Service Broker catalog items and catalog sources, including content sharing and the request of catalog items.</p>
	<p><b>Policies</b></p> <p>Interact with policies created in vRealize Automation Service Broker.</p>

Table 1-1. vRealize Automation (continued)

Main Service	Service Name and Description
Code Stream	<p><b>Code Stream</b></p> <p>Create and run pipelines for continuous delivery of your applications to production.</p>
User Profile	<p><b>Customization</b></p> <p>Configure branding information.</p>

## API versioning

Incremental changes to the API can affect your API clients. To ensure backward compatibility and protect your clients from the impact of an incremental change, you can use vRealize Automation API versioning.

API versioning allows you to lock the API at a given point in time and control when you upgrade to a new API version. In the following example, the additional *apiVersion* query parameter locks the call to the API version that was in effect on January 30, 2020.

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

It is not necessary to lock your use case to the date provided in the example. To integrate your use case with the API and lock it to a particular date, you can enter any date you choose. Or if you choose to consume the API without versioning, leave the *apiVersion* value unspecified and the latest API version is used by default. Backward compatibility is not preserved, but your use case automatically benefits from improvements or enhancements to the API.

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

## How Developers Use the vRealize 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 can start by performing the steps outlined in the following use cases:

- [Chapter 3 Onboarding a Team Using vRealize Automation Cloud Assembly APIs](#)

- [Chapter 6 Working with Blueprints/Cloud Templates and Deployments Using vRealize Automation Cloud Assembly APIs](#)
- [Chapter 7 Requesting a Deployment from a Catalog Item Using vRealize Automation Service Broker APIs](#)
- [Chapter 8 Managing Your Deployment Using vRealize Automation Cloud Assembly APIs](#)

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.

# Getting Your Authentication Token

## 2

In the REST API, VMware vRealize 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.

This chapter includes the following topics:

- [Get Your Access Token for the vRealize Automation API](#)
- [Verify User Roles](#)

## Get Your Access Token for the vRealize 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.

---

**Note** The process to obtain the access token is different depending upon the vRealize Automation version.

- For vRealize Automation version 8.0, you can use the Identity Service API alone, or you can use both the Identity Service API and IaaS API to obtain the access token. See [Get Your Access Token](#).
- For vRealize Automation versions 8.0.1, 8.1 or later, you must use both the Identity Service API and IaaS API to obtain the access token. Using the token generated by the Identity Service API alone will not work due to a missing internal state.

---

### Prerequisites

- Secure a channel between the web browser and the vRealize 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 vRealize Automation appliance, your user name, and your password.

```
identity_service_url='https://<vRA-HOSTNAME>'
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 \
"$identity_service_url/csp/gateway/am/api/login?access_token" \
-H 'Content-Type: application/json' \
-d '{
  "username": "'"$username"'",
  "password": "'"$password"'",
}' | jq -r .refresh_token`
```

- 3 Verify the API token variable is assigned.

```
echo $api_token
```

The token is a compact string of characters.

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

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

- 5 Verify the access token variable is assigned.

```
echo $access_token
```

The access token is a JSON Web Token. 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 users of both the cloud and on-premises versions of vRealize Automation and is necessary when using tools that are integrated with vRealize 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 vRealize 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 vRealize Automation API](#).

## Procedure

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

```
identity_service_url='http://<vRA-HOSTNAME>'
username='<your_username>'
password='<your_password>'
```

- 2 Get your organization ID.

```
curl -X GET \
"$identity_service_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 \
$identity_service_url/csp/gateway/am/api/loggedin/user/orgs/$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 \
$identity_service_url/csp/gateway/am/api/loggedin/user/orgs/$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 vRealize Automation API](#).

Assign variables.

```
identity_service_url='https://vra-hostname.company.com'
username='user@example.local'
password='example_password'
```

Get your organization ID.

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

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

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

# Onboarding a Team Using vRealize Automation Cloud Assembly APIs

## 3

To onboard a team, you add a project and set up an infrastructure for the team members in vRealize Automation so that new workloads can be provisioned.

Using the vRealize Automation Cloud Assembly Infrastructure as a Service (IaaS) APIs, the onboarding process includes steps to:

- Add a cloud account.
- Create a cloud zone.
- Create a project.
- Assign resources by attaching a cloud zone to the project.
- Add users to the project as either administrators or members.
- Create flavor and image mappings.
- Create network and storage profiles.

If you plan to attach a cloud zone to your project that is within an existing cloud account in vRealize Automation Cloud Assembly, you do not need to add a cloud account and cloud zone before creating the project.

## Prerequisites for Onboarding a Team

All tasks for onboarding a team share the following common prerequisites:

- Verify that you are an organization owner in VMware vRealize Automation with a vRealize Automation Cloud Assembly administrator service role.
- 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'
```

- Verify that the API version variable is assigned to a date as in the following example.

```
api_version='2019-01-15'
```

It is not necessary to lock your use case to the date provided in the example. To integrate your use case with the API and lock it to a particular date, you can enter any date you choose. Or if you choose to consume the API without versioning, leave the *apiVersion* value unspecified and the latest API version is used by default. Backward compatibility is not preserved, but your use case automatically benefits from improvements or enhancements to the API.

Any additional prerequisites are specified with the individual tasks.

This chapter includes the following topics:

- [Adding Cloud Accounts](#)
- [Create a Cloud Zone](#)
- [Create a Project](#)
- [Add a Cloud Zone to Your Project](#)
- [Create Flavor Mappings](#)
- [Create Image Mappings](#)
- [Create Network Profiles](#)
- [Creating Storage Profiles](#)
- [Creating and Using a First Class Disk](#)
- [Specify SCSI disk placement using the vRealize Automation API](#)
- [Update the Custom Properties of a Machine](#)

## Adding Cloud Accounts

Using different input variables, you can use the vRealize Automation Cloud Assembly IaaS API to create cloud accounts for AWS, vSphere, Azure, or Google Cloud Platform.

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

### Add an AWS Cloud Account

To create an AWS cloud account with or without cloud zones, you make a POST request. The request body includes the AWS-specific parameters required to create the cloud account.

#### Prerequisites

- Verify that all general prerequisites have been satisfied. See [Prerequisites for Onboarding a Team](#).
- Verify that you have the AWS access key ID and the AWS secret access key for the new cloud account.

## Procedure

- 1 Assign the AWS 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 AWS 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 AWS 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 AWS cloud account you created.

## Example: Create an AWS Cloud Account

Assign the required variables.

```
$ url='https://appliance.domain.com'
$ api_version='2019-01-15'
$ 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": "c8c3c9bfd449475-7f703c5265a63d87-
f8e705d89b2569e1aac66c6d00bf4fc7ef4b1c44100f0e944af31eb8ba3d2a5a-f4226a20b65c4675574bc5fbff6c0",
"updatedAt": "2019-07-04",
"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 vSphere-specific parameters required to create the cloud account.

### Prerequisites

- Verify that all general prerequisites have been satisfied. See [Prerequisites for Onboarding a Team](#).

- Verify that you have the following parameters for the new cloud account:
  - vSphere host name
  - vSphere user name
  - vSphere 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 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 create a vSphere cloud account without default cloud zones.

```
curl -X POST \
  "$url/iaas/api/cloud-accounts?apiVersion=$api_version" \
  -H 'Content-Type: application/json' \
  -H "Authorization: Bearer $access_token" \
  -d '{
    "cloudAccountType": "vsphere",
    "privateKeyId": "'$vsphere_user'",
  }
```

```

"privateKey": "'$vsphere_password'",
"cloudAccountProperties": {
  "hostName": "'$vsphere_host_name'",
  "acceptSelfSignedCertificate": "false",
  "dcId": "'$data_collector_id'",
  "privateKeyId": "'$vsphere_user'",
  "privateKey": "'$vsphere_password'"
},
"regionIds": [
  "<your_region_id>"
],
"createDefaultZones": false,
"tags": [
  {
    "key": "env",
    "value": "prod"
  }
],
"name": "demo-vsphere-account",
"description": "This is a demo vSphere account without default cloud zones"
}' | jq "."

```

- 8 (Optional) Submit a request to create a vSphere 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 $access_token" \
  -d '{
    "cloudAccountType": "vsphere",
    "privateKeyId": "'$vsphere_user'",
    "privateKey": "'$vsphere_password'",
    "cloudAccountProperties": {
      "hostName": "'$vsphere_host_name'",
      "acceptSelfSignedCertificate": "false",
      "dcId": "'$data_collector_id'",
      "privateKeyId": "'$vsphere_user'",
      "privateKey": "'$vsphere_password'"
    },
    "regionIds": [
      "<your_region_id>"
    ],
    "createDefaultZones": true,
    "tags": [
      {
        "key": "env",
        "value": "prod"
      }
    ],
    "name": "demo-vsphere-account-default-cloud-zones",
    "description": "This is a demo vSphere account with default cloud zones"
  }' | jq "."

```

## 9 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 "."
```

- 10 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 without default cloud zones.

Assign the required variables.

```
$ url='https://appliance.domain.com'
$ api_version='2019-01-15'
```

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": "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'
```

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

```

$ curl -X POST \
  "$url/iaas/api/cloud-accounts?apiVersion=$api_version" \
  -H 'Content-Type: application/json' \
  -H "Authorization: Bearer $access_token" \
  -d '{
    "cloudAccountType": "vsphere",
    "privateKeyId": "'$vsphere_user'",
    "privateKey": "'$vsphere_password'",
    "cloudAccountProperties": {
      "hostName": "'$vsphere_host_name'",
      "acceptSelfSignedCertificate": "false",
      "dcId": "'$data_collector_id'",
      "privateKeyId": "'$vsphere_user'",
      "privateKey": "'$vsphere_password'"
    },
    "regionIds": [
      "'$vsphere_region_id'"
    ],
    "createDefaultZones": false,
    "tags": [
      {
        "key": "env",
        "value": "prod"
      }
    ],
    "name": "demo-vsphere-account",
    "description": "This is a demo vSphere account without default cloud zones"
  }' | jq "."

```

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

```
...
  "tags": [],
  "name": "demo-vsphere-account",
  "id": "515684ccebafde75-7f703c5265a63d87-
e78aab87e9c8d5cd4cd1da1a285403f0f4e77a5240720d093e147b830b172542-23b5c527d7083675572f5099a8da0",
  "updatedAt": "2019-08-02",
  "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 `ccloud-accounts` API call, you can use a `ccloud-accounts-nsx-t` or `ccloud-accounts-nsx-v` API call to list NSX-T or NSX-V cloud accounts respectively.

### Prerequisites

- Verify that all general prerequisites have been satisfied. See [Prerequisites for Onboarding a Team](#).
- 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](#) .

```
vsphere_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.

### 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='2019-01-15'
```

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_ccloud_account_id='515684ccebafde75-7f703c5265a63d87- ... -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": "2019-08-19",
  "organizationId": "8327d53f-91ea-420a-8613-ba8f3149db95",
  "orgId": "8327d53f-91ea-420a-8613-ba8f3149db95",
  ...
```

## Add a VMware Managed Cloud Account

To create a VMware Managed Cloud (VMC) account, you make a POST request. The request body includes the VMC-specific parameters required to create the cloud account.

### Prerequisites

- Verify that all general prerequisites have been satisfied. See [Prerequisites for Onboarding a Team](#).

- Verify that you have the following parameters for the new cloud account:
  - VMC API token
  - VMC SDDC name
  - VMC vCenter private IP
  - VMC NSX Manager IP
  - VMC vCenter user name
  - VMC vCenter password
  - VMC 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 VMC account variables.

```
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 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 "."

```

- 5 To obtain the VMC account ID, examine the response.
- 6 Assign the VMC account ID variable.

```
vmc_cloud_id='<your_vmc_cloud_id>'
```

- 7 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 "."
```

- 8 Examine the response and verify that the name and ID of the VMC account you created is listed.

## Example: Create a VMC 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='2019-01-15'
```

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 VMC account variables.

```
$ 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 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 "."
```

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

```
...
  "tags": [],
  "name": "demo-vmc-account",
  "description": "VMC cloud account",
  "id": "e2188ee5e1da675590a4d60dfaf2",
  "updatedAt": "2019-08-21",
  "organizationId": "6ab4d969-1f2a-4db8-a712-78d34239d1d2",
  "orgId": "6ab4d969-1f2a-4db8-a712-78d34239d1d2",
  ...
```

## Add an Azure Cloud Account

To create an Azure cloud account, you make a POST request. The request body includes Azure-specific parameters 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 an Azure cloud account with fewer input parameters.

## Prerequisites

- Verify that all general prerequisites have been satisfied. See [Prerequisites for Onboarding a Team](#).
- Verify that you have the following parameters for the new cloud account:
  - Azure subscription ID
  - Azure tenant ID
  - Azure client application ID
  - Azure client application secret key

## Procedure

- 1 Assign the 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 an 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 an 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 Azure cloud account you created is listed.

### Example: Create an Azure Cloud Account

Assign the required variables.

```

$ url='https://appliance.domain.com'
$ api_version='2019-01-15'
$ 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": "2019-07-04",
"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 GCP-specific parameters required to create the cloud account.

### Prerequisites

- Verify that all general prerequisites have been satisfied. See [Prerequisites for Onboarding a Team](#).
- 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='2019-01-15'
$ gcp_project_id='Example-e2e'
$ gcp_private_key_id='defg3c20c85abcde6a95b44222c4c1d68554b87e'
$ gcp_private_key='-----BEGIN PRIVATE KEY-----\nMIIEvQIBADANBgkqhkiG9w0BAQEFAA
...
3izE4KDeebLh7SkwFbUt7lFW25UL20\nKAY7FRTPvb0+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": "2019-07-04",
"organizationId": "8327d53f-91ea-420a-8613-ba8f3149db95",
...
```

## 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 have been satisfied. See [Prerequisites for Onboarding a Team](#).
- Verify that you have the cloud account ID for the cloud account you added. See [Add an AWS 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

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='2019-01-15'
$ 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": "2018-08-31",
...
```

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": "2018-08-31",
...
```

## Create a Project

As a vRealize Automation Cloud Assembly administrator, you can use the Projects API to create a project. Then you add members and cloud zones to the project so that project members can deploy blueprints to the associated zones.

Before creating a project, you get a list of projects to verify that the project you plan to create does not exist. Then you create the project with vRealize Automation Cloud Assembly users assigned to project roles.

## Prerequisites

- Verify that all general prerequisites have been satisfied. See [Prerequisites for Onboarding a Team](#).
- Verify that the project roles that you plan to assign have sufficient permissions to perform project-related tasks.

---

**Note** A vRealize Automation Cloud Assembly 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 vRealize 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" : "This is an example project",
  "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 in the project.
- The type parameter is optional. It assigns the administrator, member, or viewer to a user or group role. If unspecified, the value defaults to user.

- 5 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/project-service/api/projects?apiVersion=$api_version&"$filter"'=name%20eq%20'$project_name'" | jq "."
```

- 6 Examine the response and record the ID of your newly created project.

## 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 assigned to a group role.

```
$ 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" : "admin1@mycompany.com", "type" : "group"}
  ],
  "members" : [
    {"email" : "member1@mycompany.com", "type" : "user"}
  ],
  "viewers" : [
    {"email" : "viewer1@mycompany.com", "type" : "user"}
  ] }' | jq "."
```

A snippet of the response from your request shows the administrators, members, and viewers related to the project and the project ID.

```
{
  "administrators": [
    {
      "email": "admin1@mycompany.com",
      "type": "group"
    }
  ],
  "members": [
    {
      "email": "member1@mycompany.com",
      "type": "user"
    }
  ],
  "viewers": [
    {
```

```

    "email": "viewer1@mycompany.com",
    "type": "user"
  }
],
"sharedResources": true,
"name": "Example-project",
"description": "This is an example project",
"id": "5944aacb-91de-4541-bb9e-ef2a5403f81b",
"orgId": "8327d53f-91ea-420a-8613-ba8f3149db95",
...

```

### What to do next

Add a cloud zone, plus an additional administrator and user to the project. See [Add a Cloud Zone to Your Project](#).

## Add Users to Your Project

As a vRealize Automation Cloud Assembly 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 have been satisfied. See [Prerequisites for Onboarding a Team](#).
- Verify that you have the project administrator role in your project and you have the project ID. See [Create a Project](#).
- 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.

- 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 \
"$url/project-service/api/projects/$project_id?apiVersion=$api_version" \
-H 'Content-Type: application/json' \
-H "Authorization: Bearer $access_token" \
-d '{
  "administrators" : [
    {"email" : "<your_new_administrator_email>", ["type" : <"user" | "group">]},
    {"email" : "<existing_administrator>", ["type" : <"user" | "group">]}
  ]
}' | jq "."
```

- 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 \
"$url/project-service/api/projects/$project_id?apiVersion=$api_version" \
-H 'Content-Type: application/json' \
-H "Authorization: Bearer $access_token" \
-d '{
  "members" : [
    {"email" : "<your_new_member_email>", ["type" : <"user" | "group">]},
    {"email" : "<existing_member>", ["type" : <"user" | "group">]}
  ]
}' | jq "."
```

## Example: Add an Administrator and Member to Your Project

For the new project **Example-project**, add another administrator and user at **mycompany.com**. Assign the new administrator to a group role.

```
$ url='https://appliance.domain.com'
$ api_version='2019-01-15'
$ 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/project-service/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": "group"
  }
],
"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 \
  "$url/project-service/api/projects/$project_id?apiVersion=$api_version" \
  -H 'Content-Type: application/json' \
  -H "Authorization: Bearer $access_token" \
  -d '{
    "administrators" : [
      {"email" : "newadministrator@mycompany.com", "type" : "group"},
      {"email" : "admin1@mycompany.com", "type" : "group"}
    ]
  }' | jq "."

```

Add the member. Include the existing member **member1@mycompany.com** in the PATCH request.

```

$ curl -X PATCH \
  "$url/project-service/api/projects/$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 lists the project with its administrators, members, and viewer.

```

{
  "administrators": [
    {
      "email": "newadministrator@mycompany.com",
      "type": "group"
    },
    {

```

```

    "email": "admin1@mycompany.com",
    "type": "group"
  }
],
"members": [
  {
    "email": "newmember@mycompany.com",
    "type": "user"
  },
  {
    "email": "member1@mycompany.com",
    "type": "user"
  }
],
"viewers": [
  {
    "email": "viewer1@mycompany.com",
    "type": "user"
  }
],
"sharedResources": true,
"name": "Example-project",
"description": "This is an example project",
"id": "5944aacb-91de-4541-bb9e-ef2a5403f81b",
"orgId": "8327d53f-91ea-420a-8613-ba8f3149db95"
}
}

```

## Add a Cloud Zone to Your Project

As a vRealize Automation Cloud Assembly 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 have been satisfied. See [Prerequisites for Onboarding a Team](#).
- Verify that you have added a project and you have the project ID. See [Create a 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 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='2019-01-15'
$ 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 blueprint, 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 have been satisfied. See [Prerequisites for Onboarding a Team](#).

- 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='2019-01-15'
$ cloud_account_id='c8c3c9bfdb449475-7f703c5265a63d87- ... -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":
    "c8c3c9bfdb449475-7f703c5265a63d87-5fa34c478df36b060e1ca3551254c4033013bf3283908e4661cd1c6fb2f8b9a
    e-ce5aad01092b47558644f6b6615d",
  "id": "37d6c1acf4a8275586468873c739",
  "updatedAt": "2019-04-11",
  ...
```

- 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": "2019-07-29",
  ...
```

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='2019-01-15'
$ cloud_account_id='c8c3c9bfd449475-7f703c5265a63d87- ... -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": "c8c3c9bfd449475-7f703c5265a63d87-d06bf79904ce5096492a2a2fc557fb0457d7d3c5b5e7ae20b29957788812bb3d-d5a5e16bdc3eec7557245925e1b08",
  "id": "2aaf79b789eee8755724592b06d39",
  "updatedAt": "2018-07-31",
  ...
```

- 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": "2018-08-08",
...

```

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='2019-01-15'
$ cloud_account_id='c8c3c9bfdb449475-7f703c5265a63d87- ... -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":
"c8c3c9bfdb449475-7f703c5265a63d87-5fa34c478df36b060e1ca3551254c4033013bf3283908e4661cd1c6fb2f8b9a
e-ce5aad01092b47558644f6b6615d",
"id": "37d6c1acf4a8275586468873c739",
"updatedAt": "2019-04-11",
...

```

- 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": "2019-07-29",
  ...
```

## 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 blueprint, 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.

### Prerequisites

- Verify that all general prerequisites have been satisfied. See [Prerequisites for Onboarding a Team](#).
- 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. This example specifies an AWS 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 images.

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

- 6 To select fabric image that you want, examine the response.
- 7 Create an 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": "<your_image_profile>",
    "description": "Example image profile",
    "imageMapping": {
      "ubuntu": {
        "id": "<image_id_from_response>"
      }
    }
  },
```

```

    "ubuntu16": {
      "id": "<image_id_from_response>"
    },
    "ubuntu1604": {
      "name": "https://cloud-images.ubuntu.com/releases/16.04/release-20190605/ubuntu-16.04-
server-cloudimg-amd64.ova"
    }
  },
  "regionId": "'$region_id'"
}' | jq "."

```

- 8 (Optional) For a vSphere region ID, create an image profile with an image mapping that specifies the OVA/OVF links instead of and image ID.

```

curl -X POST \
  $url/iaas/api/image-profiles?apiVersion=$api_version \
  -H 'Content-Type: application/json' \
  -H "Authorization: Bearer $access_token" \
  -d '{
    "name": "image-profile",
    "description": "vSphere Compute Images",
    "imageMapping": {
      "ubuntu": {
        "name": "https://cloud-images.ubuntu.com/releases/16.04/release-20190605/ubuntu-16.04-
server-cloudimg-amd64.ova"
      }
    },
    "regionId": "'$region_id'"
  }' | jq "."

```

- 9 To obtain the image profile ID, examine the response.
- 10 Assign the image profile ID variable.

```
image_profile_id='<your_image_profile_id>'
```

- 11 (Optional) Look up the image profile you created with your image profile ID.

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

The response shows the name and ID for the image profile you created.

---

**Note** Using the external region ID and the cloud account ID, you can also filter for the image profile with a query that does not require the image profile ID. See [Filtering Resources by Region ID](#).

---

## Example: Create image mapping

Assign the required variables including a cloud account ID.

```
$ url='https://appliance.domain.com'
$ api_version='2019-01-15'
$ cloud_account_id='c8c3c9bfdb449475-7f703c5265a63d87- ... -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":
  "c8c3c9bfdb449475-7f703c5265a63d87-5fa34c478df36b060e1ca3551254c4033013bf3283908e4661cd1c6fb2f8b9ae-
ce5aad01092b47558644f6b6615d",
  "id": "37d6c1acf4a8275586468873c739",
  "updatedAt": "2019-04-11",
  ...
```

Assign the region ID.

```
$ region_id='37d6c1acf4a8275586468873c739'
```

List all fabric images based on the name *ubuntuserver*.

```
curl -X GET -H 'Content-Type: application/json' -H "Authorization: Bearer $access_token" "$url/
iaas/api/fabric-images/?apiVersion=$api_version&" '$filter'='name%20eq%20'*ubuntuserver*latest'%20or
%20externalId%20eq%20ami-a8d2d7ce" | jq "
```

A snippet of the response shows a fabric image ID. Select the IDs that you want.

```
...
  "osFamily": "Linux",
  "externalRegionId": "francecentral",
  "isPrivate": false,
  "externalId": "Canonical:UbuntuServer:16.04-LTS:latest",
  "name": "Canonical:UbuntuServer:16.04-LTS:latest",
  "description": "Canonical:UbuntuServer:16.04-LTS:latest",
  "id": "fa13cb9367b63e755734f761858d8",
  "updatedAt": "2018-08-13",
  ...
```

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 $access_token" \
-d '{
  "name": "example-image-profile",
  "description": "Example image profile",
  "imageMapping": {
    "ubuntu": {
      "id": "fa13cb9367b63e755734f761858d8"
    },
    "ubuntu16": {
      "id": "fa13cb9367b63e755734f761858d8"
    }
  },
  "regionId": "'$region_id'"
}' | jq "
```

A snippet of the response shows the image profile ID.

```
...
  "externalRegionId": "us-east-1",
  "name": "example-image-profile",
  "description": "Example image profile",
  "id": "c70c0677a8e3787558edac19c9b10",
  "updatedAt": "2019-07-30",
  ...
```

## Create Network Profiles

To create a network profile, you make a POST request with a region ID associated with a cloud account.

A vRealize Automation Cloud Assembly 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.

The networks in this example are used for provisioning to existing or public networks.

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 have been satisfied. See [Prerequisites for Onboarding a Team](#).
- 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='2019-01-15'
$ cloud_account_id='c8c3c9bfdb449475-7f703c5265a63d87- ... -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":
  "c8c3c9bfdb449475-7f703c5265a63d87-5fa34c478df36b060e1ca3551254c4033013bf3283908e4661cd1c6fb2f8b9ae-
ce5aad01092b47558644f6b6615d",
  "id": "37d6c1acf4a8275586468873c739",
  "updatedAt": "2019-04-11",
  ...
```

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",
      "c8c3c9bfdb449475-7f703c5265a63d87-5fa34c478df36b060e1ca3551254c4033013bf3283908e4661cd1c6fb2f8b9ae-
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": "2019-08-01",
  ...

```

## Create a Network Profile with Isolation

To create a network profile with isolation, you make a POST request and specify the isolation type.

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 enabled by either using a subnet or security groups. 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 or using security groups.

## Prerequisites

- Verify that all general prerequisites have been satisfied. See [Prerequisites for Onboarding a Team](#).
- 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 [Step 2 in Create Network Profiles](#).
- Verify that you have the IDs for the networks you want to include in the profile. See [Step 5 in Create Network Profiles](#).

## Procedure

- 1 Assign the region ID variable.

```
region_id='<your_region_id>'
```

- 2 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 ""
```

- 3 Examine the response to find the IDs for the network domain that you want to include in your network profile.
- 4 Create a network profile that supports isolation using a subnet and 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 $access_token" \
-d '{
  "name": "<your-network-profile-with-isolation-by-subnet>",
  "description": "Example Network Profile",
  "regionId": "'$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.

## 5 (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_id>",
    "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 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 three types of network profiles with isolation

The following example includes the requests used to create a network that supports isolation using a subnet, using an external subnet, and using security groups.

Assign the required variables including a cloud account ID and a region ID.

```
$ url='https://appliance.domain.com'
$ api_version='2019-01-15'
$ cloud_account_id='c8c3c9bfdb449475-7f703c5265a63d87- ... -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": "2018-09-11",
  "updatedAt": "2019-04-11",
  "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/c8c3c9bfdb449475-7f703c5265a63d87-5fa34c478df36b060e1ca3551254c4033013bf3283908e4661cd1c6fb2f8b9ae-ce5aad01092b47558644f6b6615d"
      ]
    },
    "self": {
      "href": "/iaas/api/network-domains/233df662ec3b4875575931653ef00"
    }
  }
...

```

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": "2019-08-01",
  ...
```

You can also create a network profile named

**example-network-profile-with-isolation-by-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": "2019-08-01",
  ...
```

You can also create a network profile named

**example-network-profile-with-isolation-by-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": "2019-08-01",
...

```

## 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 have been satisfied. See [Prerequisites for Onboarding a Team](#).
- 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 [Step 2 in Create Network Profiles](#).
- Verify that you have the IDs for the networks you want to include in the profile. See [Step 5 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='2019-01-15'
$ cloud_account_id='c8c3c9bfdb449475-7f703c5265a63d87- ... -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": "2019-08-01",
  "updatedAt": "2019-08-01",
  "organizationId": "8327d53f-91ea-420a-8613-ba8f3149db95",
  "orgId": "8327d53f-91ea-420a-8613-ba8f3149db95",
  "_links": {
    "cloud-accounts": {
      "hrefs": [
        "/iaas/api/cloud-accounts/"
      ]
    }
  }
}
```

```

c8c3c9b9fdb449475-7f703c5265a63d87-5fa34c478df36b060e1ca3551254c4033013bf3283908e4661cd1c6fb2f8b9ae-
ce5aad01092b47558644f6b6615d"
    ]
  },
  "self": {
    "href": "/iaas/api/security-groups/bdab0d4c28af6e7558f061f772518"
  }
  ...

```

With the IDs of fabric networks that you want to include in your profile 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": "2019-08-02",
...

```

## Creating Storage Profiles

Using different input variables, you can use the vRealize Automation Cloud Assembly IaaS API to create storage profiles for AWS, vSphere, or Azure cloud accounts.

A vRealize Automation Cloud Assembly 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 AWS 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 AWS fabric volume type.

As an alternative to using the `storage-profiles` API call to create an AWS 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 AWS storage profile using the `storage-profiles` API call.

### Prerequisites

- Verify that all general prerequisites have been satisfied. See [Prerequisites for Onboarding a Team](#).
- 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 AWS 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 AWS 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 AWS storage profile

Assign the required variables including the cloud account ID for an AWS cloud account.

```
$ url='https://appliance.domain.com'
$ api_version='2019-01-15'
$ aws_cloud_account_id='c8c3c9bfd449475-7f703c5265a63d87- ... -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":
  "c8c3c9bfd449475-7f703c5265a63d87-5fa34c478df36b060e1ca3551254c4033013bf3283908e4661cd1c6fb2f8b9ae-
ce5aad01092b47558644f6b6615d",
  "id": "37d6c1acf4a8275586468873c739",
  "updatedAt": "2019-04-11",
  ...
```

Assign the AWS region ID.

```
$ aws_region_id='37d6c1acf4a8275586468873c739'
```

List all AWS 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": "2019-08-02",
"updatedAt": "2019-08-02",
...

```

## 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 have been satisfied. See [Prerequisites for Onboarding a Team](#).

- 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'%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.

```
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",
  "diskMode": "independent-persistent",
  "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='2019-01-15'
$ vsphere_cloud_account_id='c8c3c9bfdb449475-7f703c5265a63d87- ... -ce5aad01092b47558644f6b6615d'
```

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%20cloudAccountId%20eq%20'$vsphere_cloud_account_id'" | jq ""
```

A snippet of the response shows the region ID.

```
...
  "externalRegionId": "Datacenter:datacenter-10",
  "cloudAccountId":
  "c8c3c9bfdb449475-7f703c5265a63d87-809fe6fef311fdd63aa6dac546574aa898213265e988e34cc851db19b8c05b96-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": "2018-08-07",
    "updatedAt": "2018-08-10",
    ...

```

- 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 $access_token" \
  -d '{
    "name": "vsphere-storage-profile",
    "description": "Example vSphere storage profile",
    "defaultItem": true,
    "supportsEncryption": true,
    "tags": [ { "key" : "env", "value": "dev" } ],
    "diskProperties": {
      "provisioningType": "thin",
      "diskMode": "independent-persistent",
      "sharesLevel": "low",
      "shares": "500",
      "limitIops": "500"
    },
    "diskTargetProperties": {
      "storagePolicyId": "'$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": "2019-08-04",
  "updatedAt": "2019-08-04",
  ...

```

## 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 have been satisfied. See [Prerequisites for Onboarding a Team](#).
- 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='2019-01-15'
$ 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%20cloudAccountId%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": "2018-08-07",
  "updatedAt": "2018-08-10",
  ...
```

- 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": "2020-07-10",
  "updatedAt": "2020-07-10",
  ...
}
```

#### What to do next

Use the tags to create a First Class Disk. See [Create a First Class Disk](#).

## Create an Azure Storage Profile

To create an Azure storage profile, you make a POST request with a region ID. The request body includes an Azure fabric storage account ID.

As an alternative to using the `storage-profiles` API call to create an 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 an Azure storage profile using the `storage-profiles` API call.

#### Prerequisites

- Verify that all general prerequisites have been satisfied. See [Prerequisites for Onboarding a Team](#).
- 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 an 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 an Azure storage profile

Assign the required variables including the cloud account ID for an Azure cloud account.

```
$ url='https://appliance.domain.com'
$ api_version='2019-01-15'
$ azure_cloud_account_id='c8c3c9b9fdb449475-7f703c5265a63d87- ... -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":
"c8c3c9b9fdb449475-7f703c5265a63d87-5fa34c478df36b060e1ca3551254c4033013bf3283908e4661cd1c6fb2f8b9ae-
```

```
ce5aad01092b47558644f6b6615d",
  "id": "37d6c1acf4a8275586468873c739",
  "updatedAt": "2019-04-11",
  ...
```

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": "2019-08-02",
  "updatedAt": "2019-08-02",
  ...
```

## 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 have been satisfied. See [Prerequisites for Onboarding a Team](#).
- Verify that you have added a project and you have the project ID. See [Create a Project](#).
- 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='2019-01-15'
$ 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 have been satisfied. See [Prerequisites for Onboarding a Team](#).
- 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='2019-01-15'
$ 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": "2020-06-04",
  "updatedAt": "2020-06-05",
  ...
}

```

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 have been satisfied. See [Prerequisites for Onboarding a Team](#).
- 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='2019-01-15'
$ 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": "16cfd8b8-559c-49ff-8162-0a4c57079c81",
    "createdAt": "2020-07-15",
    "updatedAt": "2020-07-15",
    "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": "2020-07-15",
  "updatedAt": "2020-07-15",
  "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": "2020-07-15",
  "updatedAt": "2020-07-15",
  "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": "2020-07-15",
  "updatedAt": "2020-07-15",
  "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"
    }
  }
}
```

## Specify SCSI disk placement using the vRealize 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 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 have been satisfied. See [Prerequisites for Onboarding a Team](#).

## 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": [
    "e0f23c91d5ecca75-7f703c5265a63d87-7e3d8d60a55d1306cc791422547ead9153c3bdf1c802400819ad45a341cba1f3-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": "2020-07-02",
  "updatedAt": "2020-07-02",
  "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/e0f23c91d5ecca75-7f703c5265a63d87-7e3d8d60a55d1306cc791422547ead9153c3bdf1c802400819ad45a341cba1f3-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-
e78aab87e9c8d5cd4cd1da1a285403f0f4e77a5240720d093e147b830b172542-6b4f9990d36ee87558f04e6e8e0ca"
  ],
  "provisioningStatus": "READY",
  "customProperties": {
    "diskKind": "Unmanaged"
  },
  "externalId": "74213f6d-ee11-4549-996a-772c0621f7d1",
  "name": "Hard disk 1",
  "id": "02c39b60-32f5-4a7e-9317-88bf8a3fe20c",

```

```

"createdAt": "2020-12-11",
"updatedAt": "2020-12-11",
"organizationId": "f670fdfc-66d6-4689-9793-d524e7066d1e",
"orgId": "f670fdfc-66d6-4689-9793-d524e7066d1e",
"_links": {
  "cloud-accounts": {
    "hrefs": [
      "/iaas/api/cloud-accounts/e0f23c91d5ecca75-7f703c5265a63d87-
e78aab87e9c8d5cd4cd1da1a285403f0f4e77a5240720d093e147b830b172542-6b4f9990d36ee87558f04e6e8e0ca"
    ]
  },
  "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 blueprint 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. As a reminder, a blueprint in the API is a cloud template in the product.

## 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 have been satisfied. See [Prerequisites for Onboarding a Team](#).
- 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='2019-01-15'
```

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",
    ...
  },
  ...
```

# Querying with the IaaS APIs

# 4

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

This chapter includes the following topics:

- [Using Pagination and Count](#)
- [Filtering Resources by Region ID](#)
- [Filtering Operations for Projects](#)
- [Filtering for Machine Status](#)

## 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 specified, 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	<pre>curl -X GET -H 'Content-Type: application/json' -H "Authorization: Bearer \$access_token" "\$url/ iaas/api/cloud-accounts?apiVersion=\$api_version&amp; \$filter=cloudAccountType%20eq%20%27vsphere%27&amp; \$top=20&amp;\$skip=0"   jq "."</pre>
List the second set of 20 cloud accounts, or 21–40	<pre>curl -X GET -H 'Content-Type: application/json' -H "Authorization: Bearer \$access_token" "\$url/ iaas/api/cloud-accounts?apiVersion=\$api_version&amp; \$filter=cloudAccountType%20eq%20%27vsphere%27&amp; \$top=20&amp;\$skip=20"   jq "."</pre>
List the third set of 20 cloud accounts, or 41–45	<pre>curl -X GET -H 'Content-Type: application/json' -H "Authorization: Bearer \$access_token" "\$url/ iaas/api/cloud-accounts?apiVersion=\$api_version&amp; \$filter=cloudAccountType%20eq%20%27vsphere%27&amp; \$top=20&amp;\$skip=40"   jq "."</pre>

## 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 vRealize 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 have been satisfied. See [Prerequisites for Onboarding a Team](#).

- 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":
    "e0f23c91d5ecca75-7f703c5265a63d87-7e3d8d60a55d1306cc791422547ead9153c3bdf1c802400819ad45a341cba1f
3-c39814fe67b8247557cab2652647d",
    "updatedAt": "2020-06-27",
    "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='e0f23c91d5ecca75-7f703c5265a63d87-7e3d8d60a55d1306cc791422547ead9153c3bdf1c80240
0819ad45a341cba1f3-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":
```

```
"e0f23c91d5ecca75-7f703c5265a63d87-7e3d8d60a55d1306cc791422547ead9153c3bdf1c802400819ad45a341cba1f
3-c39814fe67b8247557cab2652647d",
  "id": "277e3cd9fe87527557cab268cae5a",
  "updatedAt": "2019-07-31",
  ...
```

- 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_account_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 have been satisfied. See [Prerequisites for Onboarding a Team](#).

- 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&"$filter="'cloudAccountType%20eq%20'vmc'%20and%20name%20eq%20'$cloud_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-f47bdfef43c"
      ]
    },
    "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": "2020-04-22",
    "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_cloud_account_id'"
```

## Resource Query Examples

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

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

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

Resource	Query
Storage Profiles AWS	<code>\$url/iaas/api/storage-profiles-aws? \$filter="externalRegionId%20eq %20'\$external_region_id'%20and%20cloudAccountId %20eq%20'\$cloud_account_id'"</code>
Storage Profiles vSphere	<code>\$url/iaas/api/storage-profiles-vsphere? \$filter="externalRegionId%20eq %20'\$external_region_id'%20and%20cloudAccountId %20eq%20'\$cloud_account_id'"</code>

## 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	<code>\$url/project-service/api/projects?\$filter=name %20eq%20'00-TestProject'</code>
not equal	<code>\$url/project-service/api/projects?\$filter=name %20ne%20'00-TestProject'</code>
logical and	<code>\$url/project-service/api/projects?\$filter=name %20eq%20'00-TestProject'%20and%20sharedResources %20eq%20true</code>
logical or	<code>\$url/project-service/api/projects?\$filter=name %20eq%20'00-TestProject'%20or%20sharedResources %20eq%20false</code>
logical negation	<code>\$url/project-service/api/projects?\$filter=not %20startswith(name,'00')</code>
starts with	<code>\$url/project-service/api/projects? \$filter=startswith(name,'00')</code>
substring of	<code>\$url/project-service/api/projects? \$filter=substringof('00', name)</code>

Operation	Query
ends with	<code>\$url/project-service/api/projects? \$filter=endswith(name, 'ct')</code>
length	<code>\$url/project-service/api/projects? \$filter=length(name)%20eq%205</code>
index of	<code>\$url/project-service/api/projects? \$filter=indexof(name, '00')%20eq%200</code>
substring from	<code>\$url/project-service/api/projects? \$filter=substring(name, 1)%20eq%20'0-Te'</code>
substring from to	<code>\$url/project-service/api/projects? \$filter=substring(name, 1, 2)%20eq%20'0-'</code>
to lower	<code>\$url/project-service/api/projects? \$filter=tolower(name)%20eq%20'00-testproject'</code>
to upper	<code>\$url/project-service/api/projects? \$filter=toupper(name)%20eq%20'00-TESTPROJECT'</code>
concatenate	<code>\$url/project-service/api/projects? \$filter=concat(concat(name, ','), description) %20eq%20'test project, test project description'</code>
trim	<code>\$url/project-service/api/projects? \$filter=trim(name)%20eq%20'00-TestProject'</code>

## 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	<code>curl -X GET \$url/project-service/api/projects? \$top=20 -H 'Content-Type: application/json' -H "Authorization: Bearer \$access_token"   jq ""</code>
Skip the first ten items in the collection	<code>curl -X GET \$url/project-service/api/projects? \$skip=10 -H 'Content-Type: application/json' -H "Authorization: Bearer \$access_token"   jq ""</code>

If you want to...	Use this request
List the items in ascending order with <b>asc</b> or descending order with <b>desc</b> . If the type of order is not specified, the list defaults to ascending order.	<p>For ascending order:</p> <pre>curl -X GET \$url/project-service/api/projects? \$skip=0\$orderBy=name%20asc -H 'Content-Type: application/json' -H "Authorization: Bearer \$access_token"   jq ". "</pre> <p>For descending order:</p> <pre>curl -X GET \$url/project-service/api/projects? \$skip=0\$orderBy=name%20desc -H 'Content-Type: application/json' -H "Authorization: Bearer \$access_token"   jq ". "</pre>
List the total number of items in the collection.	<pre>curl -X GET \$url/project-service/api/projects? \$count=true -H 'Content-Type: application/json' -H "Authorization: Bearer \$access_token"   jq ". "</pre>

## 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 vRealize Automation .

If a machine has a discovered status, you can bring it into vRealize 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 vRealize 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	<pre>curl -X GET "\$url/iaas/api/machines?apiVersion=\$api_version&amp;"\$filter="deploymentId%20eq%20'" -H 'Content-Type: application/json' -H "Authorization: Bearer \$access_token"   jq "."</pre>
List all discovered machines	<pre>curl -X GET "\$url/iaas/api/cloud-accounts?apiVersion=\$api_version&amp;"\$filter="deploymentId%20ne%20'" -H 'Content-Type: application/json' -H "Authorization: Bearer \$access_token"   jq "."</pre>

# Protecting Sensitive Data

# 5

By using the vRealize Automation Cloud Assembly 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. vRealize 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.

This chapter includes the following topics:

- [How to provision a machine with sensitive data](#)
- [Properties that Support Encryption](#)

## 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 vRealize Automation Cloud Assembly, 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 have been satisfied. See [Prerequisites for Onboarding a Team](#).
- 3 Create a project with the cloud zone that you created using the vRealize Automation Cloud Assembly 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 \
  "$url/project-service/api/projects?apiVersion=$api_version"
-H 'Content-Type: application/json'
-H "Authorization: Bearer $access_token"
-d '{
  "name" : "example-project",
  "properties": {
    "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",
...
```

- 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..."
}
...

```

vRealize 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 vRealize 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 <hr/> <b>Note</b> Data encryption is only supported for deployed machines. It is not supported for discovered machines.	<ul style="list-style-type: none"> <li>■ customProperties: value</li> <li>■ nics: customProperties: value</li> <li>■ Sensitive parts of bootConfig: content</li> <li>■ remoteAccess: password</li> </ul>	<pre>{   "name": "machine1",   "customProperties": {     "username": "guest",     "password":       "((sensitive:mypass))"   } }</pre>
Create/update/list image profiles	Sensitive parts of imageMapping: value: cloudConfig	<pre>{   "imageMapping" : {      "ubuntu": {       "id": "{{awsUbuntuId}}",       "cloudConfig": "#cloud- config\nchpasswd:\n list:\nuser1:       ((sensitive:password1))"     }   },    "name" : "aws-image-profile",   "regionId": "{{awsRegionId}}" }</pre>
Create/update/list projects	customProperties: value	<pre>{   "name": "project1",   "customProperties": {     "vidm-password":       "((sensitive:mypass))"   } }</pre>
Create/list load balancers	<ul style="list-style-type: none"> <li>■ customProperties: value</li> <li>■ nics: customProperties: value</li> </ul>	<pre>{   "name": "load-balancer1",   "nics": {     "customProperties": {       "dhcp-server-password":         "((sensitive:mypass))"     }   } }</pre>

Endpoint	You can apply encryption to:	Example Input
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>

# Working with Blueprints/Cloud Templates and Deployments Using vRealize Automation Cloud Assembly APIs

## 6

To create and update blueprints, version blueprints, and deploy blueprints, you use the vRealize Automation Cloud Assembly Blueprint APIs. Blueprints is the term used in the API. In the product, blueprints are renamed to VMware Cloud Templates.

### Prerequisites for Working with Blueprints

All tasks for working with blueprints share the following common prerequisites:

- Verify that you are at least an organization member in VMware vRealize Automation with a vRealize Automation Cloud Assembly administrator service role.
- 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'
```

- Verify that the API version variable is assigned to a date as in the following example.

```
api_version='2019-09-12'
```

It is not necessary to lock your use case to the date provided in the example. To integrate your use case with the API and lock it to a particular date, you can enter any date you choose. Or if you choose to consume the API without versioning, leave the *apiVersion* value unspecified and the latest API version is used by default. Backward compatibility is not preserved, but your use case automatically benefits from improvements or enhancements to the API.

Any additional prerequisites are specified with the individual tasks.

This chapter includes the following topics:

- [Create and Update a Blueprint](#)
- [Version and Release a Blueprint to a vRealize Automation Service Broker Catalog](#)
- [Remove a Blueprint Version from a vRealize Automation Service Broker Catalog](#)

- [Test Your Blueprint Deployment](#)
- [Deploy Your Blueprint](#)
- [Look up Deployment Details](#)

## Create and Update a Blueprint

To create a blueprint, you make a POST request. The request body includes the name of the new blueprint and the project ID of an existing project. To update a blueprint, you make a PUT request that changes one of the properties of the blueprint.

Before creating a blueprint, you get a list of blueprints to verify that the blueprint you plan to create does not already exist. After a blueprint is created, you can update it to change the blueprint definition.

### Prerequisites

- Verify that all general prerequisites have been satisfied. See [Prerequisites for Working with Blueprints](#).
- Verify that you have a project ID for the project that includes cloud zones configured to support the resource requirements of your blueprint. See [Create a Project](#).

### Procedure

- 1 Assign the project ID variable.

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

- 2 Assign your blueprint name variable.

```
blueprint_name='<your_blueprint_name>'
```

*your\_blueprint\_name* is a name that you choose.

- 3 Get a list of blueprints.

```
curl -X GET $url/blueprint/api/blueprints -H "Authorization: Bearer $access_token" | jq "."
```

- 4 To verify that the blueprint you plan to create is not already listed, examine the response.

- 5 Validate the blueprint before creating it.

```
curl -X POST \
  $url/blueprint/api/blueprint-validation?apiVersion=$api_version \
  -H 'Content-Type: application/json' \
  -H "Authorization: Bearer $access_token" \
  -d '{
    "name" : ""$blueprint_name"",
    "description" : "Basic Cloud Machine blueprint",
    "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 blueprint.

```
curl -X POST \
  $url/blueprint/api/blueprints?apiVersion=$api_version \
  -H 'Content-Type: application/json' \
  -H "Authorization: Bearer $access_token" \
  -d '{
    "name" : ""$blueprint_name"",
    "description" : "Basic Cloud Machine blueprint",
    "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 blueprint.
- 9 Assign the blueprint ID variable.

```
blueprint_id='<your_blueprint_id>'
```

*your\_blueprint\_id* is the ID of the new blueprint that you created.

- 10 To verify that the blueprint has been created, get a list of blueprints and filter for *your\_blueprint\_name*.

```
curl -X GET $url/blueprint/api/blueprints?name=$blueprint_name -H "Authorization: Bearer
$access_token" | jq "
```

- 11 (Optional) To update the blueprint, use a PUT request and specify *your\_blueprint\_id*.

```
curl -X PUT \
  $url/blueprint/api/blueprints/$blueprint_id?apiVersion=$api_version \
  -H 'Content-Type: application/json' \
  -H "Authorization: Bearer $access_token" \
  -d '{
    "name" : ""$blueprint_name"",
    "description" : "Basic Cloud Machine blueprint",
    "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 "
```



```

    "name" : ""$blueprint_name"",
    "description" : "Basic Cloud Machine blueprint",
    "content" : "formatVersion: 1\ninputs:\n  flavor:\n    type: string\n    title: Flavor\n  description: Flavor Mapping Name\n  image:\n    type: string\n    title: Image\n  description: Image Mapping Name\n  count:\n    type: integer\n    minimum: 1\n    default: 1\n    maximum: 2\n  title: Number of Instances\nresources:\n  BasicCloudMachine:\n    type: Cloud.Machine\nproperties:\n  name: BasicCloudMachine\n  flavor: ''${input.flavor}''\n  image: ''${input.image}''\n  count: ''${input.count}''",
    "projectId" : ""$project_id"",
    "requestScopeOrg": false
  } | jq ".

```

The response from your request to create the blueprint shows the blueprint 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 blueprint ID variable.

```
$ blueprint_id='1f170637-81a3-4257-b1cd-b2219ee8034c'
```

Update the blueprint to set a tag on the machine properties in the blueprint.

```

$ curl -X PUT \
  $url/blueprint/api/blueprints/$blueprint_id?apiVersion=$api_version \
  -H 'Content-Type: application/json' \
  -H "Authorization: Bearer $access_token" \
  -d '{
    "name" : ""$blueprint_name"",
    "description" : "Basic Cloud Machine blueprint",
    "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\nproperties:\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 blueprint ID to version and release the blueprint to a catalog.

## Version and Release a Blueprint to a vRealize Automation Service Broker Catalog

After creating a blueprint, you version and release your blueprint using a POST request. The request body includes the ID of an existing blueprint and the number of the version to release.

By versioning and releasing the blueprint, you mark a blueprint version as ready to be consumed by vRealize Automation Service Broker. To show the released blueprint in vRealize Automation Service Broker, you must have a catalog source.

### Prerequisites

- Verify that all general prerequisites have been satisfied. See [Prerequisites for Working with Blueprints](#).
- Verify that you have the blueprint ID for the blueprint you want to version and release. See [Create and Update a Blueprint](#).
- Verify that you know the version of the blueprint that you want to create and release to the catalog.

### Procedure

- 1 Assign the blueprint ID variable.

```
blueprint_id='<your_blueprint_id>'
```

- 2 Assign a blueprint version variable.

```
blueprint_version='<your_blueprint_version>'
```

*your\_blueprint\_version* is the version that you want to create.

- 3 To version the blueprint without releasing it, submit the request with "release": false .

```
curl -X POST \
  $url/blueprint/api/blueprints/$blueprint_id/versions?apiVersion=$api_version \
  -H "Authorization: Bearer $access_token" \
  -H 'Content-Type: application/json' \
  -d '{
    "changeLog": "Creating a version ""$blueprint_version""",
    "description": "Creating a version from the current draft",
    "release": false,
    "version": ""$blueprint_version""
  }' | jq "
```

#### 4 Release the blueprint.

```
curl -X POST \
  $url/blueprint/api/blueprints/$blueprint_id/versions/$blueprint_version/actions/release?
  apiVersion=$api_version" \
  -H "Authorization: Bearer $access_token" \
  -H 'Content-Type: application/json' | jq "."
```

### Example: Version and Release a Blueprint

Release version 5 of your blueprint with ID **1f170637-81a3-4257-b1cd-b2219ee8034c**.

Assign variables.

```
$ url='https://appliance.domain.com'
$ api_version='2019-09-12'
$ blueprint_id='1f170637-81a3-4257-b1cd-b2219ee8034c'
$ blueprint_version='v5'
```

Version the blueprint without releasing it.

```
$ curl -X POST \
  $url/blueprint/api/blueprints/$blueprint_id/versions?apiVersion=$api_version \
  -H "Authorization: Bearer $access_token" \
  -H 'Content-Type: application/json' \
  -d '{
  "changeLog": "Creating a version ""$blueprint_version"",
  "description": "Creating a version from the current draft",
  "release": false,
  "version": ""'$blueprint_version'""
}' | jq "."
```

Release the blueprint.

```
$ curl -X POST \
  $url/blueprint/api/blueprints/$blueprint_id/versions/$blueprint_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 blueprint version with a RELEASED status.

```
...
"blueprintId": "1f170637-81a3-4257-b1cd-b2219ee8034c",
"name": "MyExampleBlueprint",
"description": "Basic Cloud Machine blueprint",
"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\nproperties:\n  name: BasicCloudMachine\n  flavor: '${input.flavor}'\n  image: '${input.image}'\n  count: '${input.count}'\n  tags: [\n    {\n      \"key\": \"env
```



- 4 To see the change in the vRealize Automation Service Broker UI, select the **Content & Policies** tab.
  - a Select **Content Sources**.
  - b Select the name of the content source with your blueprint.
  - c On the Content Source Details screen that appears, click **Save and Import**.

## Results

The blueprint version you specified is removed from the vRealize Automation Service Broker catalog. Any other released blueprint versions remain listed.

## Example: Remove a Blueprint Version

Remove version 5 of your blueprint with ID **fa6b42d5-ac46-451d-8917-b2f7e527b785**.

Assign variables.

```
$ url='https://appliance.domain.com'
$ api_version='2019-09-12'
$ blueprint_id='fa6b42d5-ac46-451d-8917-b2f7e527b785'
$ blueprint_version='v5'
```

Remove the blueprint version.

```
$ curl -X POST \
  $url/blueprint/api/blueprints/$blueprint_id/versions/$blueprint_version/action/unrelease?apiVersion=
  $api_version \
  -H "Authorization: Bearer $access_token" \
  -H 'Content-Type: application/json' | jq "."
```

A snippet of the response shows the blueprint version with a VERSIONED status.

```
...
"blueprintId": "1f170637-81a3-4257-b1cd-b2219ee8034c",
"name": "MyExampleBlueprint",
"description": "Basic Cloud Machine blueprint",
"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\nproperties:\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 Blueprint Deployment

To test the deployment of a blueprint, you use the Blueprint APIs to make a POST request with the blueprint ID as input.

Before deploying a blueprint, you can test the syntax and placement of your blueprint to ensure deployment viability. If errors are reported in the test, you must fix the errors and test again before deploying the blueprint.

### Prerequisites

- Verify that all general prerequisites have been satisfied. See [Prerequisites for Working with Blueprints](#).
- Verify that the flavor mapping and image mapping specified in the released vRealize Automation blueprint version exist in your cloud account. See [Create Flavor Mappings](#) and [Create Image Mappings](#).
- Verify that you have the ID for the blueprint you want to deploy. See [Create and Update a Blueprint](#).

### Procedure

- 1 Assign the blueprint ID variable.

```
blueprint_id='<your_blueprint_id>'
```

- 2 Assign image mapping and flavor mapping variables for the blueprint 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 blueprint deployment.

```
curl -X POST \
  $url/blueprint/api/blueprint-requests?apiVersion=$api_version \
  -H "Authorization: Bearer $access_token" \
  -H 'Content-Type: application/json' \
  -d '{
    "simulate":true,
    "blueprintId": ""$blueprint_id"",
    "inputs": {
      "count": 2,
      "image":"""$image_mapping""",
      "flavor":"""$flavor_mapping""
    }
  }' | jq "."
```

- 4 Examine the response and assign the blueprint request ID.

```
blueprint_request_id='<your_blueprint_request_id>'
```

- 5 Get the status of the test request.

```
curl -X GET \
  $url/blueprint/api/blueprint-requests/$blueprint_request_id?apiVersion=$api_version \
  -H "Authorization: Bearer $access_token" | jq "."
```

## Example: Test a Deployment

For a blueprint 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'
$ blueprint_id='1f170637-81a3-4257-b1cd-b2219ee8034c'
$ image_mapping='ubuntu'
$ flavor_mapping='small'
```

Test the blueprint deployment.

```
$ curl -X POST \
  $url/blueprint/api/blueprint-requests?apiVersion=$api_version \
  -H "Authorization: Bearer $access_token" \
  -H 'Content-Type: application/json' \
  -d '{
    "simulate":true,
    "blueprintId": ""$blueprint_id"",
    "inputs": {
      "count": 2,
      "image":""$image_mapping"",
      "flavor":""$flavor_mapping""
    }
  }' | jq "."
```

A snippet of the response shows the blueprint request ID.

```
{
  "id": "5c33355e-fc52-4a30-97c3-3752cf9b644e",
  "createdAt": "2019-10-11T00:11:55.544Z",
  ...
  "blueprintId": "1f170637-81a3-4257-b1cd-b2219ee8034c",
  ...
}
```

Assign the blueprint request ID variable.

```
$ blueprint_request_id='5c33355e-fc52-4a30-97c3-3752cf9b644e'
```

Request the status of the deployment.

```
$ curl -X GET \
  $url/blueprint/api/blueprint-requests/$blueprint_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 blueprint.

## Deploy Your Blueprint

To request the deployment of a blueprint, you use the Blueprint APIs to make a POST request with the blueprint ID as input.

The prerequisites of this task call for verifying that you have the blueprint ID of the blueprint you want to deploy. In addition, this procedure includes an optional step to deploy a blueprint without a blueprint ID by providing contents inline instead.

### Prerequisites

- Verify that all general prerequisites have been satisfied. See [Prerequisites for Working with Blueprints](#).
- Verify that you have a project ID for the project that includes cloud zones configured to support the resource requirements of your blueprint. See [Create a Project](#).
- Verify that the flavor mapping and image mapping specified in the released vRealize Automation blueprint version exist in your cloud account. See [Create Flavor Mappings](#) and [Create Image Mappings](#).
- Verify that you have the ID for the blueprint you want to deploy. See [Create and Update a Blueprint](#).
- Verify that you have tested your blueprint deployment to ensure deployment viability. See [Test Your Blueprint Deployment](#).

## Procedure

- 1 Assign the blueprint ID variable.

```
blueprint_id='<your_blueprint_id>'
```

- 2 To deploy a blueprint, 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 blueprint.

```
curl -X POST \
  $url/blueprint/api/blueprint-requests?apiVersion=$api_version \
  -H "Authorization: Bearer $access_token" \
  -H 'Content-Type: application/json' \
  -d '{
    "description": "requesting deployment from blueprint",
    "blueprintId": "'"$blueprint_id"'",
    "inputs": {
      "count": 2,
      "image": "'"$image_mapping"'",
      "flavor": "'"$flavor_mapping"'",
    }
  }' | jq ". "
```

- 4 Examine the response to get the blueprint request ID and the deployment ID.
- 5 Assign the blueprint request ID and the deployment ID.

```
blueprint_request_id='<your_blueprint_request_id>'
deployment_id='<your_deployment_id>'
```

- 6 (Optional) If you do not have a blueprint ID, you can also request a blueprint deployment with contents inline.

- a Validate the blueprint before creating it.

```
curl -X POST \
  $url/blueprint/api/blueprint-validation?apiVersion=$api_version \
  -H 'Content-Type: application/json' \
  -H "Authorization: Bearer $access_token" \
  -d '{
    "name" : ""$blueprint_id"",
    "description" : "Basic Cloud Machine blueprint",
    "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 blueprint deployment.

```
curl -X POST \
  $url/blueprint/api/blueprint-requests \
  -H "Authorization: Bearer $access_token" \
  -H 'Content-Type: application/json' \
  -d '{
    "description": "requesting deployment from inline blueprint",
    "projectId": ""$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}'\''"
```

- 7 Look up the status of the blueprint deployment.

```
curl -X GET \
  $url/api/blueprint-requests/$blueprint_request_id?apiVersion=$api_version \
  -H "Authorization: Bearer $access_token" | jq "
```

## Example: Deploy a Blueprint

For a blueprint 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'
$ blueprint_id='1f170637-81a3-4257-b1cd-b2219ee8034c'
$ image_mapping='ubuntu'
$ flavor_mapping='small'
```

Request the deployment of a blueprint.

```
$ curl -X POST \
  $url/blueprint/api/blueprint-requests?apiVersion=$api_version \
  -H "Authorization: Bearer $access_token" \
  -H 'Content-Type: application/json' \
  -d '{
    "description": "requesting deployment from blueprint",
    "blueprintId": ""$blueprint_id"",
    "inputs": {
      "count": 2,
      "image":""$image_mapping"",
      "flavor":""$flavor_mapping""
    }
  }' | jq ""
```

A snippet of the response shows the blueprint request ID, the deployment ID, and the blueprint 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 blueprint request ID variable.

```
$ blueprint_request_id='889f95a8-79a3-4b2f-b19e-32d1536dd69a'
```

Request the status of the deployment.

```
$ curl -X GET \
  $url/blueprint/api/blueprint-requests/$blueprint_request_id?apiVersion=$api_version \
  -H "Authorization: Bearer $access_token" | jq ""
```

A snippet of the response shows the blueprint request ID and the blueprint 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.

## 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 have been satisfied. See [Prerequisites for Working with Blueprints](#).
- Verify that you have a deployment ID for the deployment that you requested. See [Deploy Your Blueprint](#).

### 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?expandResources=true&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.

**Results****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='2019-01-15'
$ deployment_id='15454178-63fc-42ea-b4ad-7ed8a5cdb128'
```

Look up deployment details.

```
$ curl -X GET \
  $url/deployment/api/deployments/$deployment_id?expandResources=true&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": "2019-08-09T17:56:09.463Z",
  "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"
  }
  ...
```

# Requesting a Deployment from a Catalog Item Using vRealize Automation Service Broker APIs

## 7

To create catalog items and request deployments, you use the vRealize Automation Service Broker Catalog APIs.

This use case includes the procedures required to request a blueprint. You associate a project that you previously created with the catalog source, create a catalog source, and create entitlements, 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.

## Prerequisites for Requesting a Deployment from a Catalog Item

All tasks for requesting a deployment from a catalog item share the following common prerequisites:

- Verify that you are at least an organization member in VMware vRealize Automation with both a vRealize Automation Cloud Assembly administrator service role and a vRealize Automation Service Broker administrator service role.
- 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'
```

- Verify that the API version variable is assigned to a date as in the following example.

```
api_version='2019-01-15'
```

It is not necessary to lock your use case to the date provided in the example. To integrate your use case with the API and lock it to a particular date, you can enter any date you choose. Or if you choose to consume the API without versioning, leave the *apiVersion* value unspecified and the latest API version is used by default. Backward compatibility is not preserved, but your use case automatically benefits from improvements or enhancements to the API.

- Verify that you have the ID for a project that has the blueprint versioned and released to it. See [Version and Release a Blueprint to a vRealize Automation Service Broker Catalog](#).

- Verify that you have the name of the blueprint that you want to request for deployment. See [Create and Update a Blueprint](#).
- Verify that you have the version of a blueprint released to the project that you want to request for deployment. See [Version and Release a Blueprint to a vRealize Automation Service Broker Catalog](#).

Any additional prerequisites are specified with the individual tasks.

This chapter includes the following topics:

- [Create a Catalog Source and List Discovered Items](#)
- [Create Entitlements](#)
- [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 blueprint version released to the project.

Because you are requesting the deployment of a blueprint from the catalog, this example lists the steps required to create a Cloud Assembly Blueprint source type.

### Prerequisites

- Verify that all general prerequisites have been satisfied. See [Prerequisites for Onboarding a Team](#).

### Procedure

- 1 Assign the project ID variable.

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

- 2 List all available sources in your catalog.

```
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 available 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.

- Assign the catalog type ID variable for the Cloud Assembly Blueprint source type.

```
catalog_type_id='com.vmw.blueprint'
```

- Create a catalog source for your blueprint.

```
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 ""
```

- To obtain the catalog source ID, examine the response.
- 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 ""
```

- To obtain the catalog item ID, examine the response.
- (Optional) Assign the blueprint name variable.

```
blueprint_name='<your_blueprint_name_that_was_released_to_catalog>'
```

- (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=$blueprint_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 blueprint named **BasicCloudMachine**.

Assign variables.

```
$ url='https://appliance.domain.com'
$ api_version='2019-01-15'
$ project_id='394a4ccb-22c6-4ef0-8c75-8b77efbfb51'
```

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 Blueprintechno s**.

```
...
  "id": "753d24a3-e2b0-4d5e-bba6-9e32e5964c69",
  "name": "Catalog Source from Blueprintechno 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 catalog source named **Cloud Assembly Blueprint**.

```
...
  "id": "com.vmw.blueprint",
  "name": "Cloud Assembly Blueprint",
  "baseUri": "http://catalog-service:8000/catalog/api/provider/blueprint",
  "createdBy": "deploymentsservice",
  ...
```

Assign the ID of **Cloud Assembly Blueprint** to the catalog type variable.

```
$ catalog_type_id='com.vmw.blueprint'
```

Create a catalog source.

```
$ 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 Blueprints"
  }' | jq ""
```

A snippet of the response shows the catalog source ID.

```
{
  "id": "753d24a3-e2b0-4d5e-bba6-9e32e5964c69",
  "name": "Catalog Source from Blueprints",
```

```
"typeId": "com.vmw.blueprint",
"createdAt": "2019-08-05T22:02:33.553Z",
...
```

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 blueprint name.

```
{
  "id": "718917c0-1e02-3141-8142-11da5acaed8f",
  "name": "BasicCloudMachine",
  "description": "Basic Cloud Machine blueprint",
  "sourceId": "753d24a3-e2b0-4d5e-bba6-9e32e5964c69",
  ...
}
```

Assign the catalog item ID.

```
$ catalog_item_id='718917c0-1e02-3141-8142-11da5acaed8f'
```

### What to do next

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 blueprint item. See [Create Entitlements](#).

## Create Entitlements

To create entitlements, you make a POST request with a project ID that has a blueprint version released to the project. Request bodies also include the IDs of a created catalog source or of a catalog item.

Entitlements are tied to projects. Depending upon the entitlements you want to display in your project, you can entitle all the items discovered in a catalog source or individual catalog items. This example includes the steps to create an entitlement for both a catalog source and a catalog item. The step to create an entitlement for a catalog item is optional, because creating entitlement for the catalog source that includes the item also creates entitlement for the catalog item.

### Prerequisites

- Verify that all general prerequisites have been satisfied. See [Prerequisites for Onboarding a Team](#).

- Verify that you have IDs of the catalog source or catalog item for which you plan to create entitlements. See [Create a Catalog Source and List Discovered Items](#).

## Procedure

- 1 Assign the project ID variable.

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

- 2 Assign the catalog source and catalog item IDs.

```
catalog_source_id='<your_catalog_source_id>'
catalog_item_id='<your_catalog_item_id>'
```

- 3 To see what has been entitled for your project so far, list all available entitlements.

```
curl -X GET \
  $url/catalog/api/admin/entitlements?projectId=$project_id&apiVersion=$api_version" \
  -H "Authorization: Bearer $access_token" | jq "."
```

- 4 Examine the response to see if an entitlement for your catalog item exists.
- 5 Create an entitlement for a catalog source.

```
curl -X POST \
  $url/catalog/api/admin/entitlements?apiVersion=$api_version \
  -H "Authorization: Bearer $access_token" \
  -H "Content-Type: application/json" \
  -d '{
    "projectId":"$project_id",
    "definition":{
      "type":"CatalogSourceIdentifier",
      "id":"$catalog_source_id"
    }
  }' | jq "."
```

- 6 To confirm that the entitlement for a catalog source has been defined, examine the response.
- 7 (Optional) Create an entitlement for a catalog item.

```
curl -X POST \
  $url/catalog/api/admin/entitlements?apiVersion=$api_version \
  -H "Authorization: Bearer $access_token" \
  -H "Content-Type: application/json" \
  -d '{
    "projectId":"$project_id",
    "definition":{
      "type":"CatalogItemIdentifier",
      "id":"$catalog_item_id"
    }
  }' | jq "."
```

- 8 (Optional) To confirm that the entitlement for a catalog item has been defined, examine the response.

## Example: Create entitlements for a catalog source and catalog item

In this example, the catalog source ID is for a catalog of blueprints. The catalog item is for a single blueprint.

Assign variables.

```
$ url='https://appliance.domain.com'
$ api_version='2019-01-15'
$ project_id='394a4ccb-22c6-4ef0-8c75-8b77efbefb51'
$ catalog_source_id='753d24a3-e2b0-4d5e-bba6-9e32e5964c69'
$ catalog_item_id='718917c0-1e02-3141-8142-11da5acaed8f'
```

Create entitlement for a catalog source.

```
$ curl -X POST \
  $url/catalog/api/admin/entitlements?apiVersion=$api_version \
  -H "Authorization: Bearer $access_token" \
  -H "Content-Type: application/json" \
  -d '{
    "projectId": "$project_id",
    "definition":{
      "type": "CatalogSourceIdentifier",
      "id": "$catalog_source_id"
    }
  }' | jq "."
```

The response shows the entitlement for a catalog source.

```
{
  "id": "6f432b9e-9085-4508-93ce-67f6fb5eaf68",
  "projectId": "394a4ccb-22c6-4ef0-8c75-8b77efbefb51",
  "definition": {
    "type": "CatalogSourceIdentifier",
    "id": "753d24a3-e2b0-4d5e-bba6-9e32e5964c69",
    "name": "Catalog Source from Blueprintechnology",
    "sourceType": "com.vmw.blueprint",
    "numItems": 2
  }
}
```

Or create entitlement for a discovered item.

```
$ curl -X POST \
  $url/catalog/api/admin/entitlements?apiVersion=$api_version \
  -H "Authorization: Bearer $access_token" \
  -H "Content-Type: application/json" \
  -d '{
    "projectId": "$project_id",
    "definition":{
      "type": "CatalogItemIdentifier",
      "id": "$catalog_item_id"
    }
  }' | jq "."
```

The response shows the entitlement for a catalog item.

```
{
  "id": "d32ff5f7-7913-4264-9796-b28a6d385082",
  "projectId": "394a4ccb-22c6-4ef0-8c75-8b77efbefb51",
  "definition": {
    "type": "CatalogItemIdentifier",
    "id": "718917c0-1e02-3141-8142-11da5acaed8f",
    "name": "BasicCloudMachine",
    "description": "Basic Cloud Machine blueprint",
    "sourceName": "Catalog Source from Blueprintechnologies",
    "sourceType": "com.vmw.blueprint"
  }
}
```

### What to do next

With the entitlement created for either the item as part of the catalog source or just the catalog item, you can request a deployment from the catalog item. See [Request Deployment](#).

## Request Deployment

To request a deployment from a catalog item, you make a POST request with a project ID that has a blueprint 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 blueprint.

### Prerequisites

- Verify that all general prerequisites have been satisfied. See [Prerequisites for Onboarding a Team](#).
- Verify that the flavor mapping and image mapping specified in the released vRealize Automation blueprint version exist in your cloud account. See [Create Flavor Mappings](#) and [Create Image Mappings](#).
- 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 created an entitlement for your catalog item. See [Create Entitlements](#).

### 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 that you want has been published to the catalog.
- 5 Assign the blueprint version.

```
blueprint_version='<your_blueprint_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 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.

- 7 To deploy a blueprint, 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 Blueprint from a Catalog Item

Request the deployment of a blueprint 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='2019-01-15'
$ 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": "2019-08-07T19:33:04.445Z"
},
{
  "id": "v1",
  "description": "Creating a version from the current draft",
  "createdAt": "2019-08-07T19:25:43.327Z"
}
...

```

Assign a blueprint version number.

```
$ blueprint_version='v2'
```

Assign a deployment name and check to ensure that it does not already exist.

```
$ deployment_name="Example Deployment"
$ 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** 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 blueprint, 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"
}
```

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 blueprint 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 vRealize Automation Service Broker will report an error.

### Prerequisites

- Verify that all general prerequisites have been satisfied. See [Prerequisites for Onboarding a Team](#).

### 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='2019-01-15'
$ 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": "2019-08-07T02:29:07.936Z",
  "createdBy": "admin@mycompany.com",
  "lastUpdatedAt": "2019-08-07T02:29:07.936Z",
  "lastUpdatedBy": "admin@mycompany.com"
}

```

# Managing Your Deployment Using vRealize Automation Cloud Assembly APIs



To manage your deployment after initial deployment, you use the vRealize Automation Cloud Assembly Deployment APIs.

This use case includes examples of procedures you can follow to reconfigure your deployment after initial blueprint deployment. The first example deploys a new blueprint with a deployment ID, and in subsequent examples, you use the deployment ID to update the deployment or reconfigure the blueprint components.

## Prerequisites for Managing Your Deployment

All tasks for managing your deployment share the following common prerequisites:

- Verify that you are at least an organization member in VMware vRealize Automation with a vRealize Automation Cloud Assembly administrator service role.
- 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'
```

- Verify that the API version variable is assigned to a date as in the following example.

```
api_version='2019-01-15'
```

It is not necessary to lock your use case to the date provided in the example. To integrate your use case with the API and lock it to a particular date, you can enter any date you choose. Or if you choose to consume the API without versioning, leave the *apiVersion* value unspecified and the latest API version is used by default. Backward compatibility is not preserved, but your use case automatically benefits from improvements or enhancements to the API.

- 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 blueprint 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](#).

Any additional prerequisites are specified with the individual tasks.

This chapter includes the following topics:

- [Deploy a Blueprint with Contents Inline](#)
- [Change the Lease on Your Deployment](#)
- [Get Deployment Resource IDs](#)
- [Reconfigure Load Balancer](#)
- [Add a Disk to a Machine and Power It Off](#)

## Deploy a Blueprint with Contents Inline

To request a deployment of a blueprint with contents inline, you use the Blueprint APIs to make a POST request with a project ID.

In this example, you deploy a blueprint by providing contents inline instead of providing a blueprint ID. The new blueprint has different content from previously deployed blueprints including a load balancer component, a virtual machine component, and a network component. For information about deploying a blueprint by providing a blueprint ID, see [Deploy Your Blueprint](#).

### Prerequisites

- Verify that all general prerequisites have been satisfied. See [Prerequisites for Onboarding a Team](#).

### 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 blueprint, 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 the deployment of a blueprint with contents inline.

```
curl -X POST \
  $url/blueprint/api/blueprint-requests?apiVersion=$api_version \
  -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\n\n"].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 blueprint request ID and the deployment ID, examine the response.

6 Assign the blueprint request ID variable.

```
blueprint_request_id='<your_blueprint_request_id>'
```

7 Get the status of the request.

```
curl -X GET \
  $url/blueprint/api/blueprint-requests/$blueprint_request_id?apiVersion=$api_version" \
  -H "Authorization: Bearer $access_token" | jq "."
```

## Example: Deploy a Blueprint with Contents Inline

Request a deployment named **Deployment with Blueprint 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='2019-09-12'
$ project_id='394a4ccb-22c6-4ef0-8c75-8b77efbefb51'
$ deployment_name="Deployment with Blueprint Contents Inline"
```

To deploy a blueprint, you must assign image mapping and flavor mapping variables.

```
$ image_mapping='ubuntu'
$ flavor_mapping='small'
```

Request a deployment of an inline blueprint.

```
$ curl -X POST \
  $url/blueprint/api/blueprint-requests?apiVersion=$api_version \
  -H "Authorization: Bearer $access_token" \
  -H "Content-Type: application/json" \
  -d '{
    "deploymentName": ""$deployment_name"",
    "description": "requesting blueprint 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
\n"].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 blueprint request ID.

```

...
"type": "blueprint-request",
"id": "bec37f54-3de5-451d-b484-a110c0ed8772",
"selfLink": "/blueprint/api/blueprint-requests/bec37f54-3de5-451d-b484-a110c0ed8772",
"createdAt": "2019-08-12T19:15:59.025Z",
...

```

Assign the blueprint request ID.

```
$ blueprint_request_id='bec37f54-3de5-451d-b484-a110c0ed8772'
```

Get the deployment status.

```
$ curl -X GET \
  $url/blueprint/api/blueprint-requests/$blueprint_request_id?apiVersion=$api_version \
  -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.

## 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.

## Prerequisites

- Verify that all general prerequisites have been satisfied. See [Prerequisites for Onboarding a Team](#).
- Verify that you have the ID of the deployment you want to reconfigure. See [Deploy a Blueprint 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.

```
lease_expiry_date=2019-11-15T23:11:00Z
```

- 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".

## 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='2019-01-15'
$ 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": "2019-08-12T21:47:00Z",
      "formatMaximum": "2019-11-20T19:15:00Z",
      "default": "2019-08-22T19:15:00Z"
    }
  }
...

```

Assign the lease expiry date variable in the property format.

```
$ lease_expiry_date=2019-11-15T23:11:00Z
```

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"
}
```

## 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 have been satisfied. See [Prerequisites for Onboarding a Team](#).
- Verify that you have the ID of the deployment you want to reconfigure. See [Deploy a Blueprint 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='2019-01-15'
$ 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 `cCloud-vm`.

```
...
{
  "id": "d5b4569d-2234-4fc4-a594-45e6b0251588",
  "name": "Provider_LoadBalancer_1",
  "type": "Cloud.LoadBalancer",
  ...
}
{
  "id": "42f49781-1490-4a08-ae21-8baf383a72ac",
  "name": "cCloud-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.

## 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 have been satisfied. See [Prerequisites for Onboarding a Team](#).
- Verify that you have the ID of the deployment you want to reconfigure. See [Deploy a Blueprint 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='2019-01-15'
$ 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 have been satisfied. See [Prerequisites for Onboarding a Team](#).
- Verify that you have the ID of the deployment you want to reconfigure. See [Deploy a Blueprint 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='2019-01-15'
$ 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": {}  
}
```