Using and Managing VMware Cloud Assembly

9 September 2021
VMware Cloud Assembly
VMware vRealize Automation Cloud
You can find the most up-to-date technical documentation on the VMware website at:

https://docs.vmware.com/
# Contents

1. **What is Cloud Assembly**  
   How does Cloud Assembly work

2. **Tutorials**
   - Setting up and testing vSphere infrastructure and deployments
   - Configuring and provisioning a production workload
   - Using tags to manage vSphere resources
   - Adding a cloud template to the Service Broker catalog with a custom request form
   - Onboarding and managing vSphere resources
   - Multi-cloud infrastructure and deployments
     - Part 1: Configure the example infrastructure
     - Part 2: Create the example project
     - Part 3: Design and deploy the example cloud template
   - Configuring VMware Cloud on AWS
     - Configure a basic VMware Cloud on AWS workflow
     - Configure an isolated network in VMware Cloud on AWS
   - Configuring an external IPAM integration for Infoblox
     - Add required extensible attributes in the Infoblox application before deploying the download package
     - Download and deploy an external IPAM provider package
     - Create a running environment for an IPAM integration point
     - Add an external IPAM integration for Infoblox
     - Configure a network and network profile to use external IPAM for an existing network
     - Define and deploy a cloud template that uses an external IPAM provider range assignment
     - Using Infoblox-specific properties for IPAM integrations in cloud templates
     - Control network data collection by using Infoblox filters

3. **Setting up Cloud Assembly for your organization**
   - What are the vRealize Automation Cloud user roles
   - Organization and service user roles
   - Custom user roles
   - Use cases: How can user roles help me control access
   - Adding cloud accounts
   - Credentials required for working with cloud accounts
   - Setting up cloud-based vendor accounts
   - Setting up cloud accounts that require a cloud proxy
   - Integrating with other applications
Setting up cloud-based integrations 218
Setting up integrations that use a cloud proxy 228
Configure CloudHealth Integration in Cloud Assembly 280
Integrating with vRealize Operations Cloud 281
What are onboarding plans 290
Onboard selected machines as a single deployment 291
Advanced configuration 293
What can I do with NSX-T mapping to multiple vCenters 293
What happens if I remove an NSX cloud account association 294
How do I use the IPAM SDK to create a provider-specific external IPAM integration package 295
Using vRealize Automation Cloud with Azure VMware Solution 295
Using vRealize Automation with Google Cloud VMware Engine 296
Using vRealize Automation with Oracle Cloud VMware Solution 297
Using vRealize Automation with VMware Cloud on Dell EMC 298

4 Building your resource infrastructure 299
How to add cloud zones 299
Learn more about cloud zones 300
How to add flavor mappings 303
Learn more about flavor mappings 303
How to add image mappings 304
Learn more about image mappings 304
How to add network profiles 309
Learn more about network profiles 309
Using network settings 316
Using security group settings 320
Using load balancer settings 322
How do I configure a network profile to support an on-demand network for an external IPAM integration 323
How do I configure a network profile to support an existing network for an external IPAM integration 326
How to add storage profiles 326
Learn more about storage profiles 326
How do I use pricing cards 329
How to create pricing cards for vSphere and VMC 332
How to view pricing for AWS and Azure 336
How to use tags 338
Creating a tagging strategy 341
Using capability tags in Cloud Assembly 343
Using constraint tags in Cloud Assembly 344
Standard tags 346
How Cloud Assembly processes tags 347
How do I set up a simple tagging structure 347
How to work with resources 349
  Compute resources 349
  Network resources 349
  Security resources 353
  Storage resources 354
  Machine resources 355
  Volume resources 355
  Learn more about resources 356
Configuring Multi-provider tenant resources with vRealize Automation Cloud 376
  How do I create a Virtual Private Zone for vRealize Automation Cloud 377
  Manage Virtual Private Zone configuration for vRealize Automation Cloud tenants 380
  Create global image and flavor mapping for vRealize Automation Cloud tenants 381
  Configure tenant specific image and flavor mappings for vRealize Automation Cloud 384
  Create extensibility subscriptions for providers or tenants 385
  Working with legacy Virtual Private Zones in newer versions of vRealize Automation Cloud 387

5 Adding and managing projects 388
  How do I add a project for my development team 388
  Learn more about projects 390
    Using project tags and custom properties 391
    Using project-level placement policies 392
    What are the project costs 398
  How do projects work at deployment time 398
  How do I enable Active Directory groups for projects 400

6 Designing your deployments 401
  Getting started with designs 403
  Code completion help 406
  Bindings and dependencies 408
  Template versioning 410
  User input in requests 412
    vRealize Orchestrator actions as inputs 418
  Property groups 422
    Input property groups 423
    Constant property groups 433
    Learn more about property groups 436
  Resource flags for requests 437
  Expressions 439
    Expression syntax 443
7 Managing deployments  603
   How do I monitor deployments  607
   What can I do if a Cloud Assembly deployment fails  608
   How do I manage the life cycle of a completed deployment  611
   What actions can I run on deployments  615
What is Cloud Assembly

You use VMware Cloud Assembly to connect to your public and private cloud providers so that you can deploy machines, applications, and services that you create to those resources. You and your teams develop cloud-templates-as-code in an environment that supports an iterative workflow, from development to testing to production. At provisioning time, you can deploy across a range of cloud vendors. The service is a managed VMware SaaS and NaaS-based framework.

An overview of VMware Cloud Assembly includes the following basic functions.

- The Infrastructure tab is where you add and organize your cloud vendor resources and users. This tab also provides information about deployed cloud templates.

- The Design tab is your development home. You use the canvas and the YAML editor to develop and then deploy your machines and applications.

- The Marketplace tab provides VMware Solution Exchange cloud templates and images that help you build your template library and access supporting OVA or OVFs.

- The Extensibility tab is where you can extend and automate your application life cycles. You can subscribe to events that are used to trigger extensibility actions or vRealize Orchestrator workflows.

- The Deployments tab shows the current status of your provisioned resources. You can access details and history that you use to manage your deployments.
This chapter includes the following topics:

- How does Cloud Assembly work

How does Cloud Assembly work

Cloud Assembly is a cloud template development and deployment service. You and your teams use the service to deploy machines, applications, and services to your cloud vendor resources.

As a Cloud Assembly administrator, generally referred to as a cloud administrator, you set up the provisioning infrastructure and create the projects that group users and resources.

- Add your cloud vendor accounts. See Adding cloud accounts to Cloud Assembly.
- Determine which regions or datastores are the cloud zones that you want your developers deploying to. See Learn more about Cloud Assembly cloud zones.
- Create policies that define the cloud zones. See Chapter 4 Building your Cloud Assembly resource infrastructure.
- Create projects that group the developers with the cloud zones. See Using Cloud Assembly project tags and custom properties.
As a cloud template developer, you are a member of one or more projects. You create and deploy templates to the cloud zones associated with one of your projects.

- Develop cloud templates for projects using the canvas. In addition, your project administrator can download templates and supporting images from the marketplace. See Getting started with Cloud Assembly designs and Other ways to create Cloud Assembly templates.
- Deploy your cloud templates to project cloud zones based on policies and constraints.
- Manage your deployments, including deleting unused applications. See Chapter 7 Managing Cloud Assembly deployments.

Welcome to Cloud Assembly. If you want an example of how to define the infrastructure, and then create an deploy a cloud template, see Tutorial: Setting up and testing multi-cloud infrastructure and deployments in Cloud Assembly.
Cloud Assembly Tutorials

The tutorials show you how to perform common tasks that help you become proficient with Cloud Assembly.

As you begin, a reminder that in addition to the steps in the tutorials, there is additional information in this guide. Links are provided to relevant topics.

Accessing user assistance

Equally important, user assistance is provided throughout the application. The user assistance helps you understand features and provides information that helps you make decisions about how to populate text boxes. The external documentation provides greater depth, code samples, and use cases.
This chapter includes the following topics:

- Tutorial: Setting up and testing vSphere infrastructure and deployments in Cloud Assembly
- Tutorial: Configuring Cloud Assembly to provision a production workload
- Tutorial: Using tags in Cloud Assembly to manage vSphere resources
- Tutorial: Adding a Cloud Assembly cloud template to the Service Broker catalog with a custom request form
- Tutorial: Onboarding and managing vSphere resources in vRealize Automation Cloud
- Tutorial: Setting up and testing multi-cloud infrastructure and deployments in Cloud Assembly
- Tutorial: Configuring VMware Cloud on AWS for vRealize Automation Cloud
- Tutorial: Configuring a provider-specific external IPAM integration for vRealize Automation Cloud
## Tutorial: Setting up and testing vSphere infrastructure and deployments in Cloud Assembly

If you are new to vRealize Automation Cloud or only need a refresher course, this tutorial guides you through the Cloud Assembly configuration process. You add cloud vSphere account endpoints, define the infrastructure, add users to projects, and then design and deploy a workload by using VMware Cloud Templates based on vSphere resource types, learning the process along the way.

Although this tutorial is just the beginning, you are on the path to delivering self-service automation and iterative development that works across multiple public and private clouds. This tutorial focuses on VMware vCenter Server and NSX-T. After you finish this workflow, you can apply what you’ve learned to add more types of cloud accounts and deliver more sophisticated cloud templates.

As you work your way through the steps, we provide data examples. Replace the examples with values that work in your environment.

With the exception of the installing the cloud proxy .ova on your vCenter Server, all the steps in this tutorial are performed in Cloud Assembly.

This configuration process is the foundation of your Cloud Assembly development experience. As you build your infrastructure and mature your cloud template development skills, you will repeat and expand on this workflow.

### Before you begin

- Verify that you have the Cloud Assembly Administrator role. See [Organization and service user roles in vRealize Automation Cloud](#).
- If you have not yet used the guided setup that is available in Cloud Assembly, you can do it now. The guided setup takes you through most but not all of the procedures that you do in this tutorial. To open the guided setup, click **Guided Setup** on the right side of the tab bar.
- Ensure that you have vCenter Server and NSX credentials. For more information about the permissions that the credentials must have, see [Credentials required for working with cloud accounts in vRealize Automation Cloud](#). If you plan to add additional users to projects, verify that they are members of the Cloud Assembly service.

### Step 1: Add the vCenter Server and NSX cloud accounts

The cloud accounts provide the credentials that vRealize Automation uses to connect to vCenter Server and the associated NSX server.

For secure communication between your cloud instance of Cloud Assembly and your on-premises vCenter and NSX servers, you must configure a cloud proxy. The cloud proxy ensures secure communication between the cloud and your local systems.

1. Add the cloud proxy.
The cloud proxy manages the connection between the vCenter and NSX server and Cloud Assembly. A single cloud proxy instance can support multiple cloud accounts.

When you configure the vCenter Server cloud account in the step following this one, you might notice that you can configure the cloud proxy as you configure the vCenter instance. The cloud proxy step is presented first to streamline this tutorial workflow. The next time you add a cloud account that requires a new cloud proxy, you can use an existing cloud proxy or add a new one on the cloud account page.

If you need more guidance than is provided here, see Add a cloud proxy to a vCenter Server in Cloud Assembly.

a. Select **Infrastructure > Connections > Cloud Proxies**.

b. Click **New** and follow the steps on the installation page.

c. Download the OVA file.

d. Import the OVA to the vCenter Server and begin the installation.

e. Provide the key.

f. Complete the installation.

In this tutorial, the sample name is **vCenter Cloud Proxy**.

2. Add the vCenter Server cloud account.
The vCenter Server cloud account provides the vCenter credentials that Cloud Assembly uses to discover resources and deploy cloud templates.

For additional information about vCenter Server cloud accounts, see Create a vCenter cloud account in vRealize Automation Cloud.

a Select Infrastructure > Connections > Cloud Accounts.
b Click Add Cloud Account and select vCenter.
c Enter the values.

d To verify the credentials, click Validate.
e To Allow provisioning to these datacenters, select one or more data centers.
f Skip the NSX cloud account. We'll configure that later, linking the vCenter Server account to the NSX cloud account.

g Click Add.

3 Add an associated NSX cloud account.

The NSX-T cloud account provides the NSX-T credentials that Cloud Assembly uses to discover network resources and deploy networks with cloud templates.

For more information about NSX-T cloud accounts, see Create a vCenter cloud account in vRealize Automation Cloud.

a Select Infrastructure > Connections > Cloud Accounts.

b Click Add Cloud Account and select either NSX-T or NSX-V. This tutorial uses NSX-T.

c Enter the values.

These values are only examples. Your values will be specific to your environment.

<table>
<thead>
<tr>
<th>Setting</th>
<th>Sample Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>NSX-T Account</td>
</tr>
<tr>
<td>vCenter IP address / FQDN</td>
<td>your-dev-NSX-vcenter.company.com</td>
</tr>
<tr>
<td>Username and Password</td>
<td><a href="mailto:NSXCredentials@yourCompany.com">NSXCredentials@yourCompany.com</a></td>
</tr>
<tr>
<td>Setting</td>
<td>Sample Value</td>
</tr>
<tr>
<td>--------------------</td>
<td>----------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>NSX mode</td>
<td>Don't know what to select? Here's a great opportunity to use the in-product help. Click the information icon to the right of field. Notice that the field-level help includes information that can help you configure the option. In this example, select Policy.</td>
</tr>
<tr>
<td>Cloud proxy</td>
<td>Add vCenter Server Cloud Proxy.</td>
</tr>
</tbody>
</table>

d  To verify the credentials, click **Validate**.

e  To associate the vCenter cloud account you created in the previous step, click **Add** and then select the **vCenter Account**.

   This vCenter cloud account association ensures network security.

f  On the NSX cloud account page, click **Add**.

**Step 2: Define the cloud zone compute resources**

The cloud zones are groups of compute resources in an account/region that are then made available to projects. The project members deploy cloud templates by using the resources in the assigned cloud zones. If you want to have more granular control over where project cloud templates are deployed, you can create multiple cloud zones with different compute resources.

Account/regions are how cloud vendors tie resources to isolated regions or datastores. The account indicates the cloud account type and the region indicates the region or datastore. vCenter Server uses datastores and the provisioning resources are the selected clusters and resource pools.

For this tutorial, you must ensure that the cloud zones include the resources that support the goals of the project development team, and your budget and management requirements.

For more information about cloud zones, see [Learn more about Cloud Assembly cloud zones](#).

1  Select **Infrastructure > Configure > Cloud Zones**.

2  Click the cloud zone that was added for your vCenter Server instance and enter the values.
A cloud zone defines a set of compute resources that can be used for provisioning.

**Account / region**

- vCenter Account / data center name

**Name**

- vCenter Account Cloud Zone

**Description**

- All vCenter Server compute resources for development.

**Placement policy**

- Default

**Folder**

- Select folder

**Capabilities**

Capability tags are effectively applied to all compute resources in this cloud zone, but only in the context of this cloud zone.

<table>
<thead>
<tr>
<th>Setting</th>
<th>Sample Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Account / region</td>
<td>vCenter Account / data center name</td>
</tr>
<tr>
<td>Name</td>
<td>vCenter Server Cloud Zone</td>
</tr>
<tr>
<td>Description</td>
<td>This value cannot be changed after you create it. If you want to configure a different data center for a different vCenter Server, you must create a new cloud zone where you can select the account/region.</td>
</tr>
<tr>
<td>Policy</td>
<td>Default&lt;br&gt;Don't forget to consult the help if you have questions about a field value.</td>
</tr>
</tbody>
</table>

Remember that all values are only examples. Your zone specifics will be specific to your environment.

3. Click the **Compute** tab and verify that the compute resources are all present.

   If you need to exclude one, switch to **Manually select compute** and add only the ones you want to include in the cloud zone.
Click **Save**.

5 Repeat the process for any additional cloud zones, but you must ensure unique zone names.

**Step 3: Configure the possible resources that are available for the account/region**

You added the account/region to the cloud zone. Now you define the possible machine sizes (flavor mappings), image mappings, network profiles, and storage profiles for the cloud account. The mapping and profile definitions are evaluated for a match when you deploy a cloud template, ensuring that the workload includes the appropriate machine size (flavor), image, networks, and storage.

1 Configure the flavor mappings for the account/regions.

   Flavors are sometimes referred to as t-shirt sizing. Depending on how your cloud template is configured, the applied flavor mapping determines the number of CPUs and memory.

   For more information about flavor mappings, see Learn more about flavor mappings in vRealize Automation Cloud.

   a Select **Infrastructure > Configure > Flavor Mappings**.

   b Click **New Flavor Mapping** and enter values that define small, medium, and large machines.

      Remember, these are sample values. You must select relevant account/regions and define the sizing.
Setting | Sample Value
---|---
Flavor name | small
Account/region | vCenter Account / data center
CPU value | 2
Memory value | 1 GB

Click **Create**.

To create additional sizes, configure medium and large flavor mappings for the account/region.

<table>
<thead>
<tr>
<th>Setting</th>
<th>Sample Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flavor name</td>
<td>medium</td>
</tr>
<tr>
<td>Account/region</td>
<td>vCenter Account / Datacenter</td>
</tr>
<tr>
<td>CPU value</td>
<td>4</td>
</tr>
<tr>
<td>Memory value</td>
<td>2 GB</td>
</tr>
</tbody>
</table>

Configure the image mappings for the account/regions.

The images are the operating system for machines in the cloud template. When you are working with vCenter Server images, you select vCenter templates.

For more information about image mappings, see Learn more about image mappings in vRealize Automation Cloud.

a. Select **Infrastructure** > **Configure** > **Image Mappings**.

b. Click **New Image Mapping** and search for the images for the account/region.

Remember, these are sample values. You must select relevant images that were discovered in your account/region.
c Click Create.

d Repeat the process to create additional image mappings. For example, an ubuntu mapping for the account/region.

3 Configure network profiles.

Network profiles define the networks and network settings that are available for an account/region. The profiles must support the target deployment environments.

This task provides the minimum configuration information for success. If you want more information about network profiles, start with Learn more about network profiles in vRealize Automation Cloud.

a Select Infrastructure > Configure > Network Profile.

b Click New Network Profile and create a profile for the vCenter Account / Datacenter account/region.
c Click the **Networks** tab and click **Add Network**.

<table>
<thead>
<tr>
<th>Network</th>
<th>Account Region</th>
<th>Zone</th>
<th>Network Domain</th>
<th>CIDR</th>
<th>default for Zone</th>
<th>Origin</th>
<th>Tags</th>
</tr>
</thead>
<tbody>
<tr>
<td>DevProject-004</td>
<td>vCenter Account / Datacenter</td>
<td>-</td>
<td>-</td>
<td>192.168.164.0/27</td>
<td>-</td>
<td>Deployed</td>
<td>-</td>
</tr>
<tr>
<td>External</td>
<td>vCenter Account</td>
<td>-</td>
<td>-</td>
<td>172.16.164.0/24</td>
<td>-</td>
<td>Discovered</td>
<td>-</td>
</tr>
<tr>
<td>DevProject-005</td>
<td>vCenter Account</td>
<td>-</td>
<td>-</td>
<td>10.244.5.0/24</td>
<td>-</td>
<td>Discovered</td>
<td>-</td>
</tr>
</tbody>
</table>

d Select the NSX networks that you want to make available for the application development team.

In this example, we had an NSX-T network named DevProject-004.

e Click the **Network Policies** tab and create a policy.
f Click **Create**.

4 Configure storage profiles.

Storage profiles define the disks for an account/region. The profiles must support the target deployment environments.

If you want more information about storage profiles, see with [Learn more about storage profiles in vRealize Automation Cloud](https://www.vmware.com/support/kb_article.html?id=kbarticle702779).

a Select **Infrastructure > Configure > Storage Profile**.

b Click **New Storage Profile** and create a profile for the vCenter Server/Datacenter account/region.

Unless specified in the table, keep the default values.
**Storage Profile**

<table>
<thead>
<tr>
<th>Setting</th>
<th>Sample Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Account/region</td>
<td>vCenter Account / Datacenter</td>
</tr>
<tr>
<td>Name</td>
<td>Storage Profile</td>
</tr>
<tr>
<td>Datastore/cluster</td>
<td>Selected a datastore with sufficient capacity and that is accessible to all the hosts.</td>
</tr>
<tr>
<td>Preferred storage for this region</td>
<td>Select the check box.</td>
</tr>
</tbody>
</table>

Step 4: Create a project

This is where you really begin thinking about the project goals.

- What users need access to the compute resources so that they can create and deploy an application cloud template? For more information about what the different project roles can see and do, see [Organization and service user roles in vRealize Automation Cloud](#).

- Will the members of the project be creating applications that go from development to production? What are the necessary resources?

- What cloud zones do they need? What priority and limits should be placed on each zone for the project?

For this tutorial, we are going to support the Development team as they create and extend an in-house software application.

c Click **Create**.
This task provides the minimum configuration information for success. If you want more information about projects, start with Learn more about Cloud Assembly projects.

1. Select **Infrastructure > Administration > Projects**.

2. Click **New Project** and enter the name **Development Project**.

3. Click the **Users** tab, and then click **Add Users**.

   You are not required to add users at the time. But if you want other users to work with cloud templates, they must be a member of the project.

4. Enter email addresses to add users as project members or administrators, depending on what permissions you want each individual to have.

5. Click **Provisioning** and click **Add Zones > Cloud Zone**.

6. Add the cloud zones that the users can deploy to.

   You can also set resource limits for the cloud zone in the project. In the future, you can set different limits for other projects.

7. Add any additional cloud zones to the project.

8. Click **Create**.
To verify that the project was added to the cloud zone, select **Infrastructure > Configure > Cloud Zones** and open the vCenter Account Zone cloud Zone card so that you can examine the **Projects** tab. You should see the Development Project.

**Step 5: Design and deploy a basic cloud template**

You design and deploy the cloud template to ensure that your infrastructure is properly configured to support the template. Later you can build on the template as you create an application that meets your project needs.

The best way to build a cloud template is component-by-component, verifying that it deploys between each change. This tutorial starts with a simple machine and then iteratively adds more resources.

The examples in this procedure use the YAML code editor. It is an easier way of providing you with code snippets. However, if you prefer a use dialog box-driven user interface, click **Inputs**.

There is so much more that you can do with cloud templates than is provided in this tutorial. If you want more information, start with Chapter 6 Designing your Cloud Assembly deployments.

This tutorial uses vSphere and NSX resource types. These resource types can be deployed only on vCenter Server cloud account endpoints. You can also use the cloud agnostic resource types to create cloud templates that can be deployed on any endpoint. For an example of how to configure the infrastructure and design the template for any endpoint, see Tutorial: Setting up and testing multi-cloud infrastructure and deployments in Cloud Assembly.

For a video that illustrates the basic steps in this procedure, see How to design and deploy a basic cloud template.

1. Select **Design > Cloud Templates**.
2. Select **New From > Blank Canvas**.
3. Enter the **Name Development Template**, select the **Project Development Project**, and click **Create**.
4. Add a vSphere machine to the design canvas, test, and deploy.
a  From the resource type pane, drag a **vSphere Machine** to the canvas.

Notice that the **Code** pane shows the YAML for the machine, with an empty value for image and predefined CPU and memory properties. You are going to make this template able to support flexible sizing.

b  To select an image value, put your pointer between the single quotes for `image` and select `centos` from the list of images that you configured.

Remember, these are sample values. If you did not configure a centos image, select an image that you did configure.

c  Create a line below the image property and enter or select `flavor`, then select the `small` from the list.

d  **Delete** `cpuCount` and `totalMemory`.

Your YAML should look similar to this example.

```yaml
formatVersion: 1
inputs: {}
resources:
  Cloud_vSphere_Machine_1:
    type: Cloud.vSphere.Machine
    properties:
      image: centos
      flavor: small
```

e  **Click Test**.

Test allows you to validate the syntax and placement of your cloud template. A successful test does not mean that you can deploy the template without errors.
If the test fails, click **Provisioning Diagram** and look for the failure points. For more information about using the diagram to troubleshoot, see **Test a basic cloud template**.

f Click **Deploy**.

g Enter **Deployment Name** as **DevTemplate - machine** and click **Deploy**.

You can track the progress of the deployment on the DevTemplate deployment details page or on the Deployments tab.

If the deployment fails, you can troubleshoot the problem and revise your template. See **What can I do if a Cloud Assembly deployment fails**.

A successful deployment looks similar to this example on the Deployments tab.

5 **Version the template and add a network.**

Versioning a cloud template is required to make it available in the Service Broker catalog, but it is useful to have a good version to revert to during development.

a Open the template in the design canvas.

b Click **Version**, enter a **Description** similar to **Simple deployable machine**, and click **Create**.

c From the resource type pane, drag an **NSX Network** resource type to the canvas.

d Connect the machine to the network.

Click the small circle on the machine component and drag the connection to the network.
Notice that the YAML now looks similar to this example.

```yaml
formatVersion: 1
inputs: {}
resources:
  Cloud_vSphere_Machine_1:
    type: Cloud.vSphere.Machine
    properties:
      image: centos
      flavor: small
      networks:
      - network: '${resource.Cloud_NSX_Network_1.id}'
      attachedDisks: []
  Cloud_NSX_Network_1:
    type: Cloud.NSX.Network
    properties:
      networkType: existing
```

e  Click **Test** to validate the template.

f  Click **Deploy**.

g  Enter the name **DevTemplate - machine - network** and click **Deploy**.

h  Track the progress and review the successful deployment.

6  Version the template and add data disk.

a  Open the template in the design canvas.

b  Version the template.

  **Enter Machine with existing network** as the description.

c  From the resource type pane, drag an **vSphere Disk** resource type to the canvas.

d  Connect the disk to the machine.
Notice that the YAML now looks similar to this example.

```yaml
formatVersion: 1
inputs: {}
resources:
  Cloud_vSphere_Disk_1:
    type: Cloud.vSphere.Disk
    properties:
      capacityGb: 1
  Cloud_vSphere_Machine_1:
    type: Cloud.vSphere.Machine
    properties:
      image: centos
      flavor: small
      networks:
        - network: '${resource.Cloud_NSX_Network_1.id}'
        attachedDisks:
        - source: '${resource.Cloud_vSphere_Disk_1.id}'
  Cloud_NSX_Network_1:
    type: Cloud.NSX.Network
    properties:
      networkType: existing
```

e  Test the template.

f  Deploy the template using the name `DevTemplate - machine - network - storage`.

g  Track the progress and review the successful deployment.

h  Version the template.

   Enter `Machine with existing network and storage disk` as the description.

This final version ensures that you can add a working template to the Service Catalog.

**Tutorial results**

You completed the workflow that configured Cloud Assembly as a working system. You are now familiar with the following concepts.

- Cloud accounts are the credentials that connect Cloud Assembly to your cloud vendor endpoints.

- Cloud zones are the selected compute resources in account/regions that you then assign to different projects based on the project needs and your goals for managing costs.

- Infrastructure resources are definitions of resources associated with account/regions that are used in cloud templates.

- Projects are how you give your users access to the cloud zones based on the project's application development goals.

- Cloud templates are the definitions of your application workloads that you iteratively develop and deploy.
This tutorial is the foundation of your Cloud Assembly development experience. You can use this process to build your infrastructure and mature your cloud template development skills.

**Tutorial: Configuring Cloud Assembly to provision a production workload**

As a cloud administrator, you want to automate the deployment process for a project so that when the cloud template designers are creating and deploying templates, Cloud Assembly does the work for you. For example, the workloads are deployed with a particular custom machine naming pattern, the machines are added to a specific Active Directory organizational unit, and specific DNS and IP ranges are used.

By automating the process for the project deployments, you can more easily manage multiple projects across various data centers and cloud environments.

You are not required to complete all of the tasks provided here. You can mix and match any of these tasks, depending on your management goals.

**Before you begin**

This tutorial requires you to have your infrastructure configured and to have successfully deployed a cloud template with a machine and a network. Verify that the following are already configured on your system.

- You successfully performed all of the steps specified in the infrastructure tutorial. See Tutorial: Setting up and testing vSphere infrastructure and deployments in Cloud Assembly.
- You have the Cloud Assembly Administrator role. See Organization and service user roles in vRealize Automation Cloud.

**Customize the machine names**

The goal of this task is to ensure that the deployed machines for the Development project are named based on the costcenter for the project, the resource type selected at deployment time, and incremented numbers to ensure uniqueness. For example, DevProject-centos-021.

You can adapt this example to your naming requirements.

For more about projects, see Chapter 5 Adding and managing Cloud Assembly projects.

For a video that illustrates this custom naming example, see How to create a custom naming template for deployments.

1. Select **Infrastructure > Projects**.
2. Select an existing project or create a new one.
   - For this tutorial, the project name is Development Project.
3. Click **Create**.
4 On the Projects page, click the project name on the tile so that you can configure the project.

5 Click the **Users** tab and add the users who are members of this project.

6 Click the **Provisioning** tab.
   a In the Zones section, click **Add Zone** and add the possible cloud zones where the workloads are deployed for this project.
   b In the Custom Properties section, add a custom property with the name **costCenter** and the value **DevProject**.
   c In the Custom Naming section, add the following naming template.

\[
\text{${resource.costCenter}-${resource.installedOS}-${###}}
\]

The **${resource.installedOS}** is based on the operating system selected when you deploy the cloud template.

7 Click **Save**.

8 Update the cloud template with an input value for the operating system type.

Input values are the direct way that you can customize the deployment request form for users and simplify your development process. By creating input values, you can use a single cloud template to deploy workloads with different configurations. For example, size or operating system.

This example uses the Development Template from a previous tutorial. See **Step 5: Design and deploy a basic cloud template**.

   a Select **Design** and open the Development Template.
   b In the Code pane, update the YAML with the following changes.

   - In the Inputs section, add **installedOS**.

   In the next step you can see that **installedOS** input is also used to specify the image. When you add the strings in the enum section, the values, in this example they are **centos** and **ubuntu**, must match the image names that you defined in **Infrastructure > Configure > Image Mappings**. For example, if your image mapping name is CentOS rather than centos, you should use CentOS in the inputs section.

```
inputs:
  installedOS:
    type: string
```
In the Cloud_vSphere_Machine_1 section, update the image to an installedOS input parameter (${input.installedOS}) and add an installedOS custom property with the same input parameter.

resources:
Cloud_vSphere_Disk_1:
  type: Cloud.vSphere.Disk
  properties:
    capacityGb: 1
Cloud_vSphere_Machine_1:
  type: Cloud.vSphere.Machine
  properties:
    image: ${input.installedOS}
    installedOS: ${input.installedOS}
    flavor: small
    networks:
      - network: '${resource.Cloud_NSX_Network_1.id}'
    attachedDisks:
      - source: '${resource.Cloud_vSphere_Disk_1.id}'
Cloud_NSX_Network_1:
  type: Cloud.NSX.Network
  properties:
    networkType: existing

c  Click Deploy and enter the name Custom name deployment test.
d  Click Next.
e  Select the centos operating system from the drop-down menu.

f  Click Deploy.

9  Track the progress and review the successful deployment.
The machine name in this example is DevProject-centos-026. Just a reminder, this example is based on the tutorial referenced at the beginning of this task.

Create Active Directory machine records

When you provision a workload, you can create machine records in Active Directory. By configuring Cloud Assembly to perform this task automatically for a project deployments, you have lightened your own workload as the cloud administrator.

1 Add an Active Directory integration.
   a Select **Infrastructure > Connections > Integrations**.

   These steps cover the basic Active Directory configuration that is related to this AD machine records tutorial. For more about the Active Directory integration, see [How do I create an Active Directory integration in Cloud Assembly](#).

   b Click **Add Integration** and click and click **Active Directory**.

   ![Active Directory Integration](image)

   c Enter the name that you are using for this integration.

   d Enter the **LDAP host / IP** and the associated credentials.

   e Enter the **Base DN**.
In this tutorial the example is `ou=AppDev,dc=cmbu,dc=local`. AppDev is the parent OU for the computer OU that you will add for the project.

1. Click **Add**.

2. Add the project to the integration.

3. In the Active Directory integration, click the **Projects** tab and click **Add Project**.

   **Add Projects**

   Select a project and the OU it will be mapped to by adding its relative DN. The effective DN is created by appending the RDN to the integration base DN (`dc=cmbu,dc=local`).

   - **Project**: App development
   - **Relative DN**: `ou=AppDev-Computers`
   - **Overrides**: Allow cloud template to override relative DN path
   - **Ignores**: Allow cloud template to skip adding machines to Active Directory

   Constraints:
The policy is applied only when at least one of the following criteria is matched

   - **Tags**: Enter tags

   Matching zones

   - **CANCEL**
   - **ADD**

   a. Select the App Development project.
   b. Enter the relative DNs. For example, `OU=AppDev-Computers`.
   c. Leave the Overrides and Ignores switches turned off.

   This procedure is focused on automating the process for a project. It is not about customizations that you can do in templates.
   d. Click **Add**.

4. To save your changes to the integration, click **Save**.

5. Deploy a cloud template for the project and verify that the machine added to the correct Active Directory OU.

**Set your network DNS and internal IP range**

Add or update a network profile to include your DNS servers and internal IP ranges.

You must have already created a cloud account for vSphere, NSX-V, or NSX-T. See Tutorial: Setting up and testing vSphere infrastructure and deployments in Cloud Assembly or Adding cloud accounts to Cloud Assembly.

1. Select **Infrastructure > Configure > Network Profiles**.
2. Select an existing profile or create one.
3. On the **Summary** tab, select an **Account/region** and enter a name.

   For this tutorial, the network profile name is Network Profile.
4 Add networks.
   a Click the **Networks** tab.
   b Click **Add Network**.
   c Add one or more NSX or vSphere networks.
   d Click **Add**.

5 Configure the DNS servers.
   a In the networks list on the **Networks** tab, click the network name.
   
   Networks listed here are used when provisioning to existing, on-demand, or p... 

   ![Networks Table]

   b Enter the DNS server IP addresses you want this network to use.

   ![DNS Configuration]

   c Click **Save**.

6 Specify the IP range for the network.
   a In the networks list, select the check box next to the network name.
b Click **Manage IP Ranges**.

c In the Manage IP Ranges dialog box, click **New IP Range**.

**New IP Range**

- **Network**: NSX-mcm1376447-151082888186
- **Source**: Internal
- **Name**: DevProject Range
- **CIDR**: 192.168.100.32/28
- **Start IP address**: 192.168.100.34
- **End IP address**: 192.168.100.46

d Enter a name.
For example, DevProject Range.

e. To define the range, enter the Start IP address and End IP address.
f. Click Add.
g. Add additional ranges or click Close.

7. Add the cloud zone containing the associated network account/region that you configured to your Development project.

8. Deploy a cloud template for the project and verify that the machine is provisioned within the specified IP range.

**Tutorial: Using tags in Cloud Assembly to manage vSphere resources**

Tags are powerful metadata that you can associate with resources and include in templates. You can use tags in a variety of management scenarios, including workload placement and resource labeling.

**Quick introduction to tags**

This section is a simple introduction to tags as they apply to the provided steps. For more in-depth information about tags, see [How do I use tags to manage Cloud Assembly resources and deployments](#).

- **Capability and constraint tags**
  
  You can use tags to control deployments based on resource capabilities. For example, as a cloud administrator you want the iteratively developed cloud templates to deploy to a development-specific resource pool and the production worthy templates to deploy to a different resource pool.

  - Capability tags are added to resources, defining their capabilities.
  
  - Constraint tags are used in cloud templates, defining what resources you want the deployed resources to consume.

- **Label tags**
  
  To manage resources, you can add tags as object labels or descriptions. The management possibilities include better resources searching results, differentiating between similar objects, annotating objects with custom information, providing information to third-party systems, creating security grouping membership criteria, ensuring consistency across linked SDDC domains.
Before you begin

- Review the resources and cloud template defined in Tutorial: Setting up and testing vSphere infrastructure and deployments in Cloud Assembly. The sample values used in that tutorial are used here.

Using tags to manage Workload placement

This simple example uses development and production environment tags to demonstrate how to use capability and constraint tags. First, you add capability tags on vCenter Server resource pool compute resources, and then you include the tags in the cloud template. The cloud template example demonstrates how to use inputs to let the deploying user select whether to deploy it to a development or to a production resource pool.

For an example of how to use the same tags to define placement in a multi-cloud environment, see Tutorial: Setting up and testing multi-cloud infrastructure and deployments in Cloud Assembly.

1. Add capability tags to resource pools.
   a. Select Infrastructure > Resources > Compute.
   b. Open the cloud zone and click Compute.
   c. Locate and click the resource pool that you want to deploy development workloads to.

   This tutorial uses the following sample values. Remember that these values are only examples. Your values will be specific to your environment.

<table>
<thead>
<tr>
<th>Sample resource pool</th>
<th>Sample tag</th>
</tr>
</thead>
<tbody>
<tr>
<td>wid01-clu01 / Development</td>
<td>env:dev</td>
</tr>
<tr>
<td>wid01-clu01 / Production</td>
<td>env:prod</td>
</tr>
</tbody>
</table>

   d. Add the tag env:dev and click Save.
Repeat the process for the resource pool that you want to deploy production workloads to and add the `env: prod` tag.

2 Verify that the capability tags were added to the resource pools in your cloud zone.
   a Select **Infrastructure > Configure > Cloud Zones**.
   b Open the cloud zone associated with the project and click **Compute**.

   In this example, the cloud zone is vCenter Account Cloud Zone and the tags were added to the two resource pools, wid01-clu01 / Development and wid01-clu01 / Production.

3 Add constraint tags to the cloud template.
   Constraint tags are used to limit where the template is deployed.
   a Select **Design > Cloud Templates** and then open your template.
      In this tutorial, the template name is Development Template.
   b Review the YAML for the template in the Code pane.
This YAML is the starting point for this tutorial.

```yaml
formatVersion: 1
inputs: {}
resources:
  Cloud_vSphere_Machine_1:
    type: Cloud.vSphere.Machine
    properties:
      image: centos
      flavor: medium
      networks:
        - network: '${resource.Cloud_NSX_Network_1.id}'
      attachedDisks:
        - source: '${resource.Cloud_vSphere_Disk_1.id}'
  Cloud_vSphere_Disk_1:
    type: Cloud.vSphere.Disk
    properties:
      capacityGb: 5
  Cloud_NSX_Network_1:
    type: Cloud.NSX.Network
    properties:
      networkType: existing
```

c Add the constraint tag to the Cloud_vSphere_Machine_1 resource using `input.placement` as a variable.

```yaml
resources:
  Cloud_vSphere_Machine_1:
    type: Cloud.vSphere.Machine
    properties:
      image: centos
      flavor: medium
      constraints:
        - tag: '${input.placement}'
      networks:
        - network: '${resource.Cloud_NSX_Network_1.id}'
      attachedDisks:
        - source: '${resource.Cloud_vSphere_Disk_1.id}'
```

d Define the placement variable in the Inputs section.

```yaml
inputs:
  placement:
    type: string
    enum:
      - env:dev
      - env:prod
    default: env:dev
    title: Select Placement for Deployment
    description: Target Environment
```
Verify that the final YAML looks similar to the following example.

```yaml
formatVersion: 1
inputs:
  placement:
    type: string
    enum:
      - 'env:dev'
      - 'env:prod'
    default: 'env:dev'
  title: Select Placement for Deployment
  description: Target Environment
resources:
  Cloud_vSphere_Machine_1:
    type: Cloud.vSphere.Machine
    properties:
      image: centos
      flavor: small
      constraints:
        - tag: '${input.placement}'
      networks:
        - network: '${resource.Cloud_NSX_Network_1.id}'
      attachedDisks:
        - source: '${resource.Cloud_vSphere_Disk_1.id}'
  Cloud_vSphere_Disk_1:
    type: Cloud.vSphere.Disk
    properties:
      capacityGb: 5
  Cloud_NSX_Network_1:
    type: Cloud.NSX.Network
    properties:
      networkType: existing
```

To try out the tag variable against the available resources, click **Test** and then select **env:dev**.

Repeat the test using **env:prod**. When both tests are successful, confirm that the template works by deploying it.

4. Deploy the template to test the workload placement.
   a. In the cloud template designer, click **Deploy**.
   b. Enter **Deployment Tag Dev** as the **Deployment Name** and click **Next**.
c. Select `env: dev` in the **Select Placement for Deployment** drop-down menu and click **Deploy**.

5. Verify that the template deployed the resources to the selected resource pool.
   a. On the **Deployments** tab, locate the Deployment Tag Dev deployment.
   b. Open the deployment details and click **Topology**.
   c. Click the vSphere machine and expand the machine information in the right pane.
   d. In the **General** section, locate **Compute host** and verify that the value matches the resource pool that matches your `env:dev` tag.

In this example, the value is **wid01-clu01 / Development**, illustrating that the workload was deployed to correct resource pool based on the selected constraint tag.

e. Repeat the deployment process, this time select `env:prod`.

Adding tags as labels that you can use in vCenter Server and NSX-T

You can add tags to deployments that you can then use to manage resources.

In this example, you add tags to identify the MySQL machine and network. You also add a tag to identify the web network. Due to how tags work on existing networks compared to on-demand networks, you have two choices.

- If you use the existing network profile that you used in the previous section, the NGINX:web tag is not added to existing objects in NSX-T. So you can ignore the verification steps regarding this tag in NSX-T.
- If you create an on-demand network profile, you can update the network in the YAML to use the routed/on-demand network. The on-demand network is used in this example so that we can demonstrate the NGINX:web tag on the new object in NSX-T.

The following YAML is from the previous example except that it uses a routed on-demand networkType. It includes the constraint tags.

This tutorial uses the following sample values. Remember that these values are only examples. Your values will be specific to your environment.
1. Select **Design > Cloud Templates** and then open your template.

2. In the `Cloud_vSphere_Machine_` properties, add the following tag.

   ```json
tags:
  - key: db
    value: mysql
   ```

3. Add VM NIC tags.

   ```json
tags:
  - key: db
    value: mysql
   ```

4. Add NSX logical switch/segment tags.

   ```json
tags:
  - key: NGINX
    value: web
   ```
5 Verify that the YAML looks similar to the following example.

```yaml
formatVersion: 1
inputs:
  placement:
    type: string
    enum:
      - 'env:dev'
      - 'env:prod'
    default: 'env:dev'
    title: Select Placement for Deployment
    description: Target Environment
resources:
  Cloud_vSphere_Machine_1:
    type: Cloud.vSphere.Machine
    properties:
      image: centos
      flavor: small
      constraints:
        - tag: '${input.placement}'
    tags:
      - key: db
        value: mysql
    networks:
      - network: '${resource.Cloud_NSX_Network_1.id}'
        tags:
          - key: db
            value: mysql
    attachedDisks:
      - source: '${resource.Cloud_vSphere_Disk_1.id}'
  Cloud_vSphere_Disk_1:
    type: Cloud.vSphere.Disk
    properties:
      capacityGb: 5
  Cloud_NSX_Network_1:
    type: Cloud.NSX.Network
    properties:
      networkType: routed
      constraints:
        - tag: 'net:od'
    tags:
      - key: NGINX
        value: web
```

6 Deploy the template.

This example uses the name **Development template w tags**.

7 To verify the tags in the deployment, open the deployment and click the **Topology** tab.
   a Click the machine in the topology.
   b Expand the **General** section for the machine and locate the Tags label.
   The tag value is **db:mysql**.
c  Expend the **Network** section and locate the network Tags column.

The tag value is `db:mysql`.

![Development template w tags](image)

![Cloud_vSphere_Machine_1](image)


d  Click the network in the topology and expand the **General** section to locate the Tag label.

The tag value is `NGINX:web`.

![Cloud_NEX_Network](image)

8  To verify the tags in vCenter Server, log in to the vCenter Server instance where this workload was deployed.

a  Locate the virtual machine and locate the Tags pane.
9. To verify the tags in NSX-T, log in to the NSX-T instance where this network is configured.
   a. Click **Policy** in the upper right corner.
   b. To locate the db:mysql tag associated with the NIC, search for **mysql**.
   c. Click **Logical Ports** and locate the deployed vSphere machine.
   d. Click the number in the Tags column.
      The Scope and Tag are **db** and **mysql** respectively.

   e. To locate the NGINX:web tag associated with segment, search for the network.
      In this example, the network name is **Cloud_NSX_Network_1-mcm1292-163799928607**.
   f. Locate the Segments row and click the number in the tags column.
      The Scope and Tag are **NGINX** and **web** respectively.
Tutorial: Adding a Cloud Assembly cloud template to the Service Broker catalog with a custom request form

During the iterative development of your cloud templates or when you have a final template, you can make the templates available to consumers in the Service Broker self-service catalog. To further enhance the user experience, you can create a custom request form. The customized form is more powerful than the simple template input options.

What to do first

- Verify that you have the infrastructure that supports your template. If you do not, start with Tutorial: Setting up and testing vSphere infrastructure and deployments in Cloud Assembly and continue with the other tutorials.

- Verify that you tagged some resource pools as `env:dev` and `env:prod`. For more information, see Tutorial: Using tags in Cloud Assembly to manage vSphere resources.

- Ensure that you have a deployable cloud template, similar to the one below. This tutorial starts with the following template.

```json
formatVersion: 1
inputs:
  installedOS:
    type: string
title: Operating System
description: Select the operating system.
enum:
  - centos
  - ubuntu
placement:
    type: string
enum:
  - 'env:dev'
  - 'env:prod'
default: 'env:dev'
title: Select Placement for Deployment
description: Target Environment
resources:

Cloud_vSphere_Disk_1:
```

VMware, Inc. 48
Step 1: Add inputs to the cloud template

In addition to the existing OS type input, this procedure updates the placement input and adds a size input. When you customize the request form in Service Broker, these are the three fields on the request form that are customized.

1. In Cloud Assembly, select **Design > Cloud Template** and create or open the template provided above.

   The sample template is used to explain the different options and includes sample values. Adapt it to your environment.

2. Add the size variable and define the sizes in the Inputs section.
   a. In the Cloud_vSphere_Machine_1 section, add a variable to the `flavor` property.

   ```
   flavor: '${input.size}'
   ```

   b. In the Inputs section, add a user input name size so that the user can select the size of the deployment. This is sometimes referred to as the t-shirt size that you defined for the cloud zones.

   ```
   size:
   ```
**3** Update placement inputs with a descriptive term rather than the tag strings.

These constraint tags will be matched with the capability tags that you added in Tutorial: Using tags in Cloud Assembly to manage vSphere resources.

a  In the Inputs section, add a user input named `placement` so that the user can select development or production as the deployment placement.

This example uses the `oneOf` attribute, which allows you to present a natural language label while still submitting strings that the deployment process requires. For example, the `env:dev` and `env:prod` tags.

```yaml
placement:
  type: string
  oneOf:
    - title: Development
      const: 'env:dev'
    - title: Production
      const: 'env:prod'
  default: 'env:dev'
  title: Select Deployment Placement
  description: Target Environment
```

**4** Review the full YAML to ensure that it looks similar to the following example.

```yaml
formatVersion: 1
inputs:
  installedOS:
    type: string
    title: Operating system
    description: Select the operating system.
    enum:
      - centos
      - ubuntu
  placement:
    type: string
    oneOf:
      - title: Development
        const: 'env:dev'
      - title: Production
        const: 'env:prod'
    default: 'env:dev'
    title: Select Deployment Placement
    description: Target Environment
size:
```
**Deployment size**

*description: Select the deployment t-shirt size.*

```
enum:
  - small
  - medium
  - large
```

**resources:**

- **Cloud_vSphere_Disk_1**
  ```
  type: Cloud.vSphere.Disk
  properties:
    capacityGb: 1
  ```

- **Cloud_vSphere_Machine_1**
  ```
  type: Cloud.vSphere.Machine
  properties:
    image: '${input.installedOS}'
    installedOS: '${input.installedOS}'
    flavor: '${input.size}'
    constraints:
      - tag: '${input.placement}'
    tags:
      - key: db
        value: mysql
    networks:
      - network: '${resource.Cloud_NSX_Network_1.id}'
        tags:
          - key: db
            value: mysql
      - source: '${resource.Cloud_vSphere_Disk_1.id}'
    attachedDisks:
      - source: '${resource.Cloud_vSphere_Disk_1.id}'
  ```

- **Cloud_NSX_Network_1**
  ```
  type: Cloud.NSX.Network
  properties:
    networkType: existing
    tags:
      - key: NGINX
        value: web
  ```

5. **Click Deploy**, verify that the second page of the request looks similar to the following example, and then you can verify that the deployment is in the selected development or production resource pool after deployment.
Step 2: Version and release the cloud template

When you have a deployable template, you can now make it available in the Service Broker catalog for other uses to deploy. To make the cloud template discoverable so that you can add it to the catalog, you must release it. In this procedure we will version it, to capture a snapshot of the template, and then release the template.

1. Select Design > Cloud Template and open the template in the design canvas.

2. Click Version and enter a description.

   Creating Version
   
   Version    7
   Last Version 8
   Description Placement inputs added and tested
   Change Log
   Release [Release this version to the catalog]
   
   This cloud template is restricted to this project in the catalog. Edit shareability in cloud template level settings

   [CANCEL] [CREATE]

3. Select the Release check box and click Create.

   Releasing the cloud template does not automatically add it to Service Broker. Releasing it makes is discoverable so the you can add it to the catalog.

Step 3: Add the cloud template to the Service Broker catalog

You can use the Service Broker catalog to provide cloud templates to other consumers in your organization where they don't need to have any awareness of how to create a template. The catalog allows them to deploy the template.

Before you can add the template as a catalog item, you must import it into Service Broker. You can only import released cloud templates.

1. To open Service Broker from Cloud Assembly, click the applications menu in the upper right corner.
2. Click **Service Broker**.

3. Import the cloud template.
   a. In Service Broker, select **Content and Policies > Content Sources**.
   b. Click **New** and then select **VMware Cloud Templates**.
   c. Enter a **Name**.
      In this tutorial, enter **Cloud Assembly DevProject**.
   d. For the **Project**, select the **Development Project** that you created in Cloud Assembly.
   e. Click **Validate**.
      The system must indicate that it found at least one item.
   f. When validated, click **Create and Import**.
      Cloud Assembly DevProject is added to the list as a content source.

4. Make the cloud template available in the catalog.
   a. Select **Content and Policies > Content Sharing**.
   b. In the **Project** drop-down list, select **Development Project**.
   c. Click **Add Items** and then select
   d. In the **Share Items** dialog box, select **Cloud Assembly DevProject** and click **Save**.

5. To verify that the Development Template was added to the catalog, click **Catalog**.

6. Click **Request** on the Development Template card.
   Notice that the inputs that you saw on the cloud template are provided here. The next step is to customize the request form.
Step 4: Create a custom form for the template

The goal for this custom form is to provide a form where the user selects the operating system and placement based on the env:dev or env:prod tags. Then the env:dev option allows the user to select small or medium, large is not an option. However, if the user selects env:prod, there is not option to select large, the size is hidden from the user but is included in the request.

1. To create a custom form in Service Broker, select Content and Policies > Content.
2. Click the vertical ellipsis to the left of the Development Template entry and click Customize form.
3. Customize the input option.
   a. In the canvas, click fields in the canvas and configure the Properties as specified in the following table.
<table>
<thead>
<tr>
<th>Canvas field name</th>
<th>Appearance</th>
<th>Values</th>
<th>Constraints</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operating system</td>
<td>Label and type</td>
<td>Value options</td>
<td></td>
</tr>
<tr>
<td></td>
<td>■ Label = Operating system</td>
<td>■ Value options = Constant</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ Value source = centos, CentOS, ubuntu</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ Value source = CentOS, ubuntu, Ubuntu</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>This example uses the value options to customize the all lower case operating system names with the preferred OS name.</td>
<td></td>
</tr>
<tr>
<td>Select Deployment Placement</td>
<td></td>
<td>Value options</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ Value options = Constant</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ Value source = env:dev, Development, env:prod, Production</td>
<td></td>
</tr>
<tr>
<td>Deployment Size</td>
<td>Visibility</td>
<td>Default value</td>
<td></td>
</tr>
<tr>
<td></td>
<td>■ Value source = Conditional value</td>
<td>■ Value source = Conditional value</td>
<td></td>
</tr>
<tr>
<td></td>
<td>■ Set value = Yes if Select Deployment Placement Equals env:dev</td>
<td>■ Set value = large if Select Deployment Placement Equals env:prod</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Value options</td>
<td></td>
</tr>
<tr>
<td></td>
<td>■ Value options = Constant</td>
<td>■ Value options = Constant</td>
<td></td>
</tr>
<tr>
<td></td>
<td>■ Value source = small, Small, medium</td>
<td>■ Value source = small, Small, medium</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Notice that the value source does not include large. Large is excluded because it is only available for Production and is the required value. The large value is included in deployment request without a user-initiated action.</td>
<td></td>
</tr>
</tbody>
</table>

b  To turn on the form in the catalog, click **Enable**.

c  Click **Save**.

4  To ensure the correct results by submitting at least a Development Small and a Production request, test the form in the catalog.
Use following examples to verify the results.

a  Test the Development Small request form by providing a name, Test small in this example, and selecting CentOS, Development, and Small for the options.

b  To verify the Development Small deployment, Select Deployments > Deployments and click the Test small deployment.

c  On the Topology tab, click the Cloud_vSphere_Machine, and then locate the Custom Properties section in the right pane.

A few of the values to review include cpuCount = 2 and flavor = small.

d  Test the Production request form by entering a name, Test large in this example, and select CentOS and Production for the options.

Remember, you configured the form to neither display nor require the user to select the size.
To verify the Production deployment, select Deployments > Deployments and click the Test large deployment.

On the Topology tab, click the Cloud_vSphere_Machine, and then locate the Custom Properties section in the right pane.

A few of the values to review include cpuCount = 8 and flavor = large.

Step 5: Control the cloud template versions in the catalog

In most cases, you want to make only the latest cloud templates available in the Service Broker catalog. The following procedure supports iterative development, where you release a version of template and add it to the catalog, but now you improved the template and want to replace the current version with the newer version.
In step 2, you versioned and released a template, so you are familiar with the process. In step 3, you added it to the catalog. The procedure ties the two steps together as you do iterative development and update the catalog with the latest version.

You do have the option to make multiple versions available in the catalog.

1. In Cloud Assembly, version the template that you now want to make available in the catalog.
   a. Select Design > Cloud Template and open the template in the design canvas.
   b. Click Version History.
   c. Locate the version that you want to add to the catalog and click Version.
   d. Enter a Description, select the Release check box, and click Create.
      At this point, you have the option to keep the old version in the catalog. If you want multiple versions, ignore the next step where you Unrelease a version.
   e. To make only one version of the template available in the catalog, review the version history list and click Unrelease on every version that you don't want in the catalog.

2. To update the Service Broker catalog with the latest version, and to replace any old version, you must collect the new version.
   a. In Service Broker, select Content and Policies > Content Sources.
   b. Click the Cloud Assembly DevProject content source that is used in this tutorial.
   c. Click Validate.
      You should see a message that an item is found.
   d. Click Save and Import.

3. Verify that the catalog displays the needed versions or no versions.
   a. In Service Broker, click Catalog.
   b. Locate the catalog item and click Request.
   c. At the top of the request form, click the Version and verify the version or versions.

The following screenshot shows 8.
Tutorial: Onboarding and managing vSphere resources in vRealize Automation Cloud

As a cloud administrator who has recently added a new cloud account, you want to begin managing some of the vCenter Server workloads using Cloud Assembly and Service Broker. This tutorial guides you through the onboarding process and how to set up a few of the management options for your existing vSphere workloads.

The sample management tasks include adding the resources to a project, creating and applying an approval policy in Service Broker, and running a few day 2 actions on the resources to demonstrate the life cycle management tools and to trigger the approval policy.

This tutorial assumes that although you might be relatively new to Cloud Assembly, you have configured new vSphere cloud account. When you add the cloud account, Cloud Assembly discovers the currently unmanaged resources on your vSphere instance.

What to do first

- Add your new vCenter Server account. For additional instructions, see Create a vCenter cloud account in vRealize Automation Cloud.
- Verify that your user account has at least Cloud Assembly Administrator and Service Broker Administrator service roles. See What are the vRealize Automation Cloud user roles.
- To properly test the approval policy from the perspective of one of your users, verify that you have a user account that has only the following user roles. In this tutorial, the user is named Sylvia.
  - Organization Member
  - Cloud Assembly User
  - Service Broker User
  For more information about user roles, see What are the vRealize Automation Cloud user roles.

Step 1: Verify that Cloud Assembly discovered the resources

When you add a vCenter Server account, Cloud Assembly discovers the resources on the vCenter Server instance. You can verify that the machines that you want to begin managing are available to onboard.

1. In Cloud Assembly, select **Infrastructure > Resources > Machines**.
2. In the grid, review the **Origin** and **Account/Region**.

   The Discovered origin type indicates that the machine is discovered on your vSphere instance rather than deployed by vRealize Automation Cloud or already onboarded. When you onboard discovered machines, the origin changes from discovered to deployed.

   In this example, the **Account/Region** is **vCenter Account / wld01-DC**.
Step 2: Create a target project

Create a project that you can assign the onboarded machines to. To manage the resources, they must be part of a project that includes the source cloud zone on which they were originally deployed.

To test this tutorial, you must have another user who is not an administrator. In this step, as an administrator, you add Sylvia as the project member.

For more information about projects, see Chapter 5 Adding and managing Cloud Assembly projects.

1. In Cloud Assembly, select **Infrastructure > Projects**.
2. On the Projects page, click **New Project**.
3. Enter the project **Name**.
   - In this tutorial, the project name is **Onboarding Project**.
4. Click the **Users** tab.
   - Click **Add Users** and add at least one user as at least a project member.
     - In this tutorial, you add Sylvia.
   - Click **Add**.
5. Click **Provisioning**.
   - Click **Add Zone**.
   - Click **Cloud Zone**.
   - Select the account/region you identified in Step 1.
     - In this tutorial, the sample value is **vCenter Account / wld01-DC**.
d  Click **Add**.

6  Click **Create**.

**Step 3: Create and run an onboarding plan**

As a cloud administrator, you onboard discovered machines from your vSphere instance so that you can apply governance and manage the resources with day 2 actions.

For more information about onboarding plans, see *What are onboarding plans in Cloud Assembly*.

1  In Cloud Assembly, select **Infrastructure > Onboarding**, and then click **New Onboarding Plan**.

2  Enter the onboarding information.

<table>
<thead>
<tr>
<th>Setting</th>
<th>Sample Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Plan name</td>
<td>wildcard-01-DC Onboarding Plan</td>
</tr>
<tr>
<td>Cloud account</td>
<td>vCenter Account</td>
</tr>
<tr>
<td>Default project</td>
<td>Onboarding Project</td>
</tr>
</tbody>
</table>

3  Click **Create**.

4  Add the machines that you want to onboard.

Do not run the onboarding plan until you complete all of the following steps.

a  Click **Machines**, and then click **Add Machines**.

b  Select the machines that you want to include in the plan, and then click **OK**.

   For this tutorial, only two machines are selected.

c  In the Create Deployments dialog box, select **Create plan deployments for each machine**, and then click **Create**.

   You select this option when you want the machines as individual deployments so that you can manage them as individual resources.
d  The selected machines are added to the list.

```
<table>
<thead>
<tr>
<th>Name</th>
<th>Status</th>
<th>Power</th>
<th>Address</th>
<th>Deployment</th>
<th>Custom properties</th>
<th>Tags</th>
</tr>
</thead>
<tbody>
<tr>
<td>DevProject-centos-0 10</td>
<td>Pending</td>
<td>On</td>
<td></td>
<td>Deployment-5e3a7...</td>
<td>Inherited</td>
<td></td>
</tr>
<tr>
<td>DevProject-centos-0 04</td>
<td>Pending</td>
<td>On</td>
<td></td>
<td>Deployment-50507...</td>
<td>Inherited</td>
<td></td>
</tr>
</tbody>
</table>
```

d01-DC Onboarding Plan

5  Rename the deployments.

   a  Click **Deployments** on the onboarding page.

   b  To change the generated deployment name, select a deployment and click **Rename**.

   c  Enter the new name, and then click **Save**.

   For example, Onboarded machine 1.

   d  Repeat as needed.

6  Assign an owner to the deployments.

   If you do not assign an owner, you become the owner. The owner must be a member of the target project.

   This tutorial assigns all the deployments to the same owner. Optionally, you can assign different deployments to different owners.

   a  Select all the deployments and click **Edit Owner**.

   b  Select the owner and click **Save**.

   Review the deployment name and owner changes in the grid.
After you run the onboarding plan, you cannot modify the name or assign owners. If you add more machines to the plan, you can modify the name or the owner.

Review the resources that you onboarded as deployments.

a. Click the main Deployments tab and click Deployments.

b. To locate deployments, you can search by deployment name, project, or owner.

Now that you have brought machines into vRealize Automation Cloud, you can begin managing them.

**Step 4: Resize a deployment**

Perform this step as a cloud administrator and familiarize yourself with how day 2 actions work. The changes that you can make to deployments are referred to as day 2 actions. Using day 2 actions are the first step in managing your resources.
For this tutorial, you think that the CPU count on a machine is too high, and you want to decrease the consumed CPUs. This procedure assumes that you are running the resize action on a vSphere machine that is powered on. It also assumes that you do not have any day 2 policies that prohibit a user from running this action.

The available actions depend on the resource type, the resource state, and the day 2 policies. For more information about day 2 actions, see What actions can I run on Cloud Assembly deployments.

1. In Cloud Assembly, select Deployments > Deployments, and then locate your onboarded deployments.
   
   You can use the search or filter options.

2. Expand the deployment using the arrow on the left, and then click the vertical ellipsis on the machine name and click Resize.

3. In the Resize dialog box, decrease the CPU count to 4 and click Submit.

   The suggested value is an example, change the CPU count to a value that works in your environment.

   The action runs on the machine.

4. To verify that the CPU count is changed, open the deployment and check the cpuCount custom property for the machine.

5. You can also verify the count in vCenter Server.
Step 5: Applying approval policies

As a cloud administrator, you can apply governance in vRealize Automation Cloud to limit what the users can do or to require them to have approval before they do it. This tutorial shows you how to apply approval policies to the resize action so that your users cannot reconfigure a machine, perhaps catastrophically, without your approval or the approval of another administrator.

The policies are created in Service Broker. However, the policies apply to the relevant requests in Cloud Assembly and Service Broker.

As an approver, you must respond to the approval request in Service Broker.

1. In Service Broker, select **Content and Policies > Policies > Definitions**, and then click **New Policy**.
2. Click **Approval Policy**.
3. Configure the approval policy.

![Resize Approval Policy](image)
The following table includes sample values that illustrate how to create the policy.

<table>
<thead>
<tr>
<th>Setting</th>
<th>Sample value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Resize Approval Policy</td>
</tr>
<tr>
<td>Scope</td>
<td>Select Project, and then select Onboarding Project. The approval policy is triggered when a user who is a member of the project runs a Resize day 2 action.</td>
</tr>
<tr>
<td>Approval type</td>
<td>User based. This value allows you to name the approvers.</td>
</tr>
<tr>
<td>Approver mode</td>
<td>Any. If you have multiple approvers, the approval request can be resolved by at least one approver.</td>
</tr>
<tr>
<td>Approvers</td>
<td>Add yourself as an approver.</td>
</tr>
<tr>
<td>Auto expiry decision</td>
<td>Reject. By rejecting an unreviewed request, you reduce the risk of making a machine either unusable or over resourced.</td>
</tr>
<tr>
<td>Auto expiry trigger</td>
<td>1</td>
</tr>
<tr>
<td>Actions</td>
<td>Select the resize action that triggers the approval policy.</td>
</tr>
<tr>
<td></td>
<td>1. Enter machine.resize in the Search.</td>
</tr>
<tr>
<td></td>
<td>2. Click Select multiple in the search results drop-down list.</td>
</tr>
<tr>
<td></td>
<td>For this tutorial, which is based on vSphere, you select the vSphere.Machine action. If you want the action policy to apply to other resource types, you can add the other Machine.Resize actions.</td>
</tr>
</tbody>
</table>

**Step 6: Request a resize request as a user**

In this step you log in to Service Broker as an Organization member and Service Broker user and run a resize day 2 request. The request creates an approval request. The user can also perform the same steps in Cloud Assembly.

In the step after this one, you log in as the user who you assigned as an approver in Step 5 and approve the request.

1. Log in to Service Broker as a user.
   
   In this tutorial, the user is Sylvia.

2. Select **Deployments > Deployments** and locate Onboarded machine 1.
   
   This deployment is the one where you ran the resize action on the machine in Step 4, changing the number of CPUs from 8 to 4. If you used a different value, modify the machine in a way that you want to test.

3. Run the **Resize** action on the machine, increasing the CPU count to 6.

4. Notice that the request is waiting for an approval.
To see the pending status, hover over the information icon in the grid or open the deployment and review the **History** tab.

5  As a user, the change the Sylvia requested does not proceed until it is approved.
6  Log out of Service Broker as the user.

In Step 7 you log in as the assigned approver and respond to the request.

**Step 7: Respond to an approval request**

When a request requires an approval, and you are the approver, you receive in email message. For this tutorial, we are not waiting for the message. Instead, the process guides you through directly to responding to approval requests using the Service Broker Approvals tab.

1  Log in to Service Broker as the user you assigned as the approver in Step 5.

   In this tutorial, the approver is Fritz.

2  Select **Deployments > Deployments** and locate Onboarded machine 1.

   The status is the grid looks the same as it did for Sylvia.

3  Click the **Approvals** tab.

   Notice that you have an approval request pending.

4  To view the request details, click the deployment name.
5  Click **Approve**, provide a comment, if needed, and click **Approve**.

6  Return to the **Deployments** page to see that the Sylvia’s resize action is now in progress.

7  When the resize action is completed, you can verify the number of CPUs in the deployment details and in the vSphere Client.

This tutorial guided you through the process of bringing the machines into vRealize Automation Cloud so that you can begin managing the life cycle of the resource.

**Tutorial: Setting up and testing multi-cloud infrastructure and deployments in Cloud Assembly**

This end-to-end Cloud Assembly tutorial shows how you might deploy in a multiple-cloud setting. You deploy the same cloud template to more than one provider, in this case AWS and Microsoft Azure.

In this example, the application is a WordPress site. Look at the sequential setup to understand the process that brings the entire design to completion.

Remember that the names and values you see are only examples. You won’t be able to use them letter-by-letter in your own environment.

To fit your own cloud infrastructure and deployment needs, consider where you would make your own substitutions for the example values.
Part 1: Configure the example Cloud Assembly infrastructure

First, configure the resources where Cloud Assembly engineering users can later develop, test, and put the application into production.

The infrastructure includes cloud targets, and definitions around the available machines, networks, and storage that the WordPress site will need.

1. Add cloud accounts

In this step, the cloud administrator adds two cloud accounts. The example project expects to do development and testing work on AWS, and go to production on Azure.

Procedure

1. Go to Infrastructure > Connections > Cloud Accounts.

2. Click Add Cloud Account, select Amazon Web Services, and enter values.

<table>
<thead>
<tr>
<th>Setting</th>
<th>Sample Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Access key ID</td>
<td>R5SDR3PXVV2ZW8B7YNSM</td>
</tr>
<tr>
<td>Secret access key</td>
<td>SZXAINXU4UHNAQ1E156S</td>
</tr>
<tr>
<td>Name</td>
<td>OurCo-AWS</td>
</tr>
<tr>
<td>Description</td>
<td>WordPress</td>
</tr>
</tbody>
</table>

Remember that all values are only examples. Your account specifics will vary.

3. To verify credentials, click Validate.

4. Click Add.

5. Edit the newly added account Configuration, and allow provisioning to us-east-1 and us-west-2 regions.

6. Click Add Cloud Account, select Microsoft Azure, and enter values.

<table>
<thead>
<tr>
<th>Setting</th>
<th>Sample Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subscription ID</td>
<td>ef2avpf-dfdv-zxluguil-94h0-i8ep2jwp4c9arbfe</td>
</tr>
<tr>
<td>Tenant ID</td>
<td>dso9wv3-4zgc-5nrcy5h3m-4skf-nnovp40wfxsro22r</td>
</tr>
<tr>
<td>Client application ID</td>
<td>bg224oq-3ptp-mbhi6aa05-q511-uflyjr2sttyik6bs</td>
</tr>
<tr>
<td>Client application secret key</td>
<td>7uqx1t57-0wtn-kymgf9wjc-t2t7-e52e4nu5fig4pmdd</td>
</tr>
<tr>
<td>Name</td>
<td>OurCo-Azure</td>
</tr>
<tr>
<td>Description</td>
<td>WordPress</td>
</tr>
</tbody>
</table>

7. To verify credentials, click Validate.
8. Click Add.

9. Edit the newly added account Configuration, and allow provisioning to the East US region.

2. Add cloud zones

In this example step, the cloud administrator adds three cloud zones, one each for development, testing, and production.

Cloud zones are the resources onto which the project will deploy the machines, networks, and storage to support the WordPress site.

Procedure

1. Go to Infrastructure > Configure > Cloud Zones.

2. Click New Cloud Zone, and enter values for the development environment.

<table>
<thead>
<tr>
<th>Cloud Zone Setting</th>
<th>Sample Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Account / region</td>
<td>OurCo-AWS/us-east-1</td>
</tr>
<tr>
<td>Name</td>
<td>OurCo-AWS-US-East</td>
</tr>
<tr>
<td>Description</td>
<td>WordPress</td>
</tr>
<tr>
<td>Placement policy</td>
<td>Default</td>
</tr>
<tr>
<td>Capability tags</td>
<td>env:dev</td>
</tr>
</tbody>
</table>

Remember that all values are only examples. Your zone specifics will vary.

3. Click Compute, and verify that the zones you expect are there.

4. Click Create.

5. Repeat the process twice, with values for the test and production environments.

<table>
<thead>
<tr>
<th>Cloud Zone Setting</th>
<th>Sample Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Account / region</td>
<td>OurCo-AWS/us-west-2</td>
</tr>
<tr>
<td>Name</td>
<td>OurCo-AWS-US-West</td>
</tr>
<tr>
<td>Description</td>
<td>WordPress</td>
</tr>
<tr>
<td>Placement policy</td>
<td>Default</td>
</tr>
<tr>
<td>Capability tags</td>
<td>env:env</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Cloud Zone Setting</th>
<th>Sample Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Account / region</td>
<td>OurCo-Azure/East US</td>
</tr>
<tr>
<td>Name</td>
<td>OurCo-Azure-East-US</td>
</tr>
<tr>
<td>Description</td>
<td>WordPress</td>
</tr>
</tbody>
</table>
Cloud Zone Setting | Sample Value
--- | ---
Placement policy | Default
Capability tags | env:prod

## 3. Add flavor mappings

In this example step, the cloud administrator adds flavor mappings to account for capacity needs that might vary depending on deployment.

Flavor mapping accounts for different size machine deployments and is informally referred to as T-shirt sizing.

### Procedure

1. Go to **Infrastructure > Configure > Flavor Mappings**. Each cloud zone needs to allow for small, medium, and large flavors.

2. Click **New Flavor Mapping**, and enter values for the development cloud zone.

<table>
<thead>
<tr>
<th>Setting</th>
<th>Sample Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flavor name</td>
<td>small</td>
</tr>
<tr>
<td>Account/region</td>
<td>OurCo-AWS/us-east-1</td>
</tr>
<tr>
<td>Value</td>
<td>t2.micro</td>
</tr>
<tr>
<td>Account/region</td>
<td>OurCo-AWS/us-west-2</td>
</tr>
<tr>
<td>Value</td>
<td>t2.medium</td>
</tr>
<tr>
<td>Account/region</td>
<td>OurCo-Azure/East US</td>
</tr>
<tr>
<td>Value</td>
<td>Standard_A0</td>
</tr>
</tbody>
</table>

Remember that all values are only examples. Your flavors will vary.

3. Click **Create**.

4. Repeat the process twice, with values for medium and large flavors.

<table>
<thead>
<tr>
<th>Setting</th>
<th>Sample Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flavor name</td>
<td>medium</td>
</tr>
<tr>
<td>Account/region</td>
<td>OurCo-AWS/us-east-1</td>
</tr>
<tr>
<td>Value</td>
<td>t2.medium</td>
</tr>
<tr>
<td>Account/region</td>
<td>OurCo-AWS/us-west-2</td>
</tr>
<tr>
<td>Value</td>
<td>t2.medium</td>
</tr>
<tr>
<td>Account/region</td>
<td>OurCo-Azure/East US</td>
</tr>
<tr>
<td>Value</td>
<td>Standard_A3</td>
</tr>
</tbody>
</table>
4. Add image mappings

In this example step, the cloud administrator adds an image mapping for Ubuntu, the host for the WordPress server and its MySQL database server.

Plan for the operating system by adding image mappings. Each cloud zone needs a Ubuntu image mapping.

Procedure

1. Go to Infrastructure > Configure > Image Mappings.
2. Click New Image Mapping, and enter values for Ubuntu servers.

<table>
<thead>
<tr>
<th>Setting</th>
<th>Sample Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Image name</td>
<td>ubuntu</td>
</tr>
<tr>
<td>Account/region</td>
<td>OurCo-AWS/us-east-1</td>
</tr>
<tr>
<td>Value</td>
<td>ubuntu-16.04-server-cloudimg-amd64</td>
</tr>
<tr>
<td>Account/region</td>
<td>OurCo-AWS/us-west-2</td>
</tr>
<tr>
<td>Value</td>
<td>ubuntu-16.04-server-cloudimg-amd64</td>
</tr>
<tr>
<td>Account/region</td>
<td>OurCo-Azure/East US</td>
</tr>
<tr>
<td>Value</td>
<td>azul-zulu-ubuntu-1604-923eng</td>
</tr>
</tbody>
</table>

Remember that all values are only examples. Your images will vary.

3. Click Create.

5. Add network profiles

In this example step, the cloud administrator adds a network profile to each cloud zone.

In each profile, the administrator adds a network for the WordPress machines, and a second network that will sit on the other side of an eventual load balancer. The second network will be the one that users eventually connect to.

Procedure

1. Go to Infrastructure > Configure > Network Profiles.
2 Click **New Network Profile**, and create a profile for the development cloud zone.

<table>
<thead>
<tr>
<th>Network Profile Setting</th>
<th>Sample Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Account / region</td>
<td>OurCo-AWS/us-east-1</td>
</tr>
<tr>
<td>Name</td>
<td>devnets</td>
</tr>
<tr>
<td>Description</td>
<td>WordPress</td>
</tr>
</tbody>
</table>

3 Click **Networks**, and click **Add Network**.

4 Select wpnet, appnet-public, and click **Add**.

Remember that all values are only examples. Your network names will vary.

5 Click **Create**.

This Wordpress example does not require that you specify network policy or network security settings.

6 Repeat the process twice, to create a network profile for the Wordpress example test and production cloud zones. In each case, add the wpnet and appnet-public networks.

<table>
<thead>
<tr>
<th>Network Profile Setting</th>
<th>Sample Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Account / region</td>
<td>OurCo-AWS/us-west-2</td>
</tr>
<tr>
<td>Name</td>
<td>testnets</td>
</tr>
<tr>
<td>Description</td>
<td>WordPress</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Network Profile Setting</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Account / region</td>
<td>OurCo-Azure/East US</td>
</tr>
<tr>
<td>Name</td>
<td>prodnets</td>
</tr>
<tr>
<td>Description</td>
<td>WordPress</td>
</tr>
</tbody>
</table>

6. **Add storage profiles**

In this example step, the cloud administrator adds a storage profile to each cloud zone.

The administrator places fast storage at the production zone and general storage at development and test.

**Procedure**

1 Go to **Infrastructure > Configure > Storage Profiles**.

2 Click **New Storage Profile**, and create a profile for the development cloud zone.

   Additional fields appear after you select the account/region.
<table>
<thead>
<tr>
<th>Storage Profile Setting</th>
<th>Sample Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Account / region</td>
<td>OurCo-AWS/us-east-1</td>
</tr>
<tr>
<td>Name</td>
<td>OurCo-AWS-US-East-Disk</td>
</tr>
<tr>
<td>Description</td>
<td>WordPress</td>
</tr>
<tr>
<td>Device type</td>
<td>EBS</td>
</tr>
<tr>
<td>Volume type</td>
<td>General Purpose SSD</td>
</tr>
<tr>
<td>Capability tags</td>
<td>storage:general</td>
</tr>
</tbody>
</table>

Remember that all values are only examples.

3 Click **Create**.

4 Repeat the process to create a profile for the test cloud zone.

<table>
<thead>
<tr>
<th>Storage Profile Setting</th>
<th>Sample Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Account / region</td>
<td>OurCo-AWS/us-west-2</td>
</tr>
<tr>
<td>Name</td>
<td>OurCo-AWS-US-West-Disk</td>
</tr>
<tr>
<td>Description</td>
<td>WordPress</td>
</tr>
<tr>
<td>Device type</td>
<td>EBS</td>
</tr>
<tr>
<td>Volume type</td>
<td>General Purpose SSD</td>
</tr>
<tr>
<td>Capability tags</td>
<td>storage:general</td>
</tr>
</tbody>
</table>

5 Repeat the process to create a profile for the production cloud zone, which has different settings because it is an Azure zone.

<table>
<thead>
<tr>
<th>Storage Profile Setting</th>
<th>Sample Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Account / region</td>
<td>OurCo-Azure/East US</td>
</tr>
<tr>
<td>Name</td>
<td>OurCo-Azure-East-US-Disk</td>
</tr>
<tr>
<td>Description</td>
<td>WordPress</td>
</tr>
<tr>
<td>Storage type</td>
<td>Managed disks</td>
</tr>
<tr>
<td>Disk type</td>
<td>Premium LRS</td>
</tr>
<tr>
<td>OS disk caching</td>
<td>Read only</td>
</tr>
<tr>
<td>Data disk caching</td>
<td>Read only</td>
</tr>
<tr>
<td>Capability tags</td>
<td>storage:fast</td>
</tr>
</tbody>
</table>
What to do next

Create a project to identify users, and to define provisioning settings. See Part 2: Create the example Cloud Assembly project.

Part 2: Create the example Cloud Assembly project

The example Cloud Assembly project enables the users who can provision, and configures how much provisioning is possible.

Projects define the user and provisioning settings.

- Users and their role level of permission
- Priority for deployments as they are being provisioned to a cloud zone
- Maximum number of deployment instances per cloud zone

Procedure

1. Go to Infrastructure > Administration > Projects.
2. Click New Project, and enter the name WordPress.
3. Click Users, and click Add Users.
4. Add email addresses and roles for the users.
   
   To successfully add a user, a VMware Cloud Services administrator must have enabled access to Cloud Assembly for the user.

   Remember that addresses shown here are only examples.

   - chris.ladd@ourco.com, Member
   - kerry.mott@ourco.com, Member
   - pat.tubb@ourco.com, Administrator
5. Click Provisioning, and click Add Cloud Zone.
6. Add the cloud zones that the users can deploy to.

<table>
<thead>
<tr>
<th>Project Cloud Zone Setting</th>
<th>Sample Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cloud zone</td>
<td>OurCo-AWS-US-East</td>
</tr>
<tr>
<td>Provisioning priority</td>
<td>1</td>
</tr>
<tr>
<td>Instances limit</td>
<td>5</td>
</tr>
<tr>
<td>Cloud zone</td>
<td>OurCo-AWS-US-West</td>
</tr>
<tr>
<td>Provisioning priority</td>
<td>1</td>
</tr>
<tr>
<td>Instances limit</td>
<td>5</td>
</tr>
<tr>
<td>Cloud zone</td>
<td>OurCo-Azure-East-US</td>
</tr>
<tr>
<td>Provisioning priority</td>
<td>0</td>
</tr>
<tr>
<td>Instances limit</td>
<td>1</td>
</tr>
</tbody>
</table>

7. Click Create.
8. Go to **Infrastructure > Configure > Cloud Zones**, and open a zone that you created earlier.

9. Click **Projects**, and verify that WordPress is a project that is allowed to provision to the zone.

10. Check the other zones that you created.

**What to do next**

Create a basic cloud template.

**Part 3: Design and deploy the example Cloud Assembly template**

Next, you define the example application—the WordPress site—in the form of a generic cloud template. The template can be deployed to different cloud vendors without needing to change its design.

The example consists of a WordPress application server, MySQL database server, and supporting resources. The template starts with a few resources, and then grows as you modify them and add more resources.

Here are the values from **Part 1: Configure the example Cloud Assembly infrastructure**, the infrastructure that was set by a cloud administrator:

- Two cloud accounts, AWS and Azure.
- Three cloud zone environments:
  - Development—OurCo-AWS-US-East
  - Test—OurCo-AWS-US-West
  - Production—OurCo-Azure-East-US
- Flavor mappings with small, medium, and large compute resources for each zone.
- Image mappings for Ubuntu configured in each zone.
- Network profiles with internal and external subnets for each zone.
- Storage on which to deploy; general storage for the development and test zone, and fast storage for the production zone.
- The example project includes all three cloud zone environments plus the users who can create designs.
Prerequisites

To follow along, you must be familiar with your own infrastructure values. This example uses AWS for development and test, and Azure for production. When creating your own cloud template, substitute your own values, typically set by your cloud administrator.

Procedure

1. Create a basic cloud template

   In this Cloud Assembly design example, you start with a cloud template that contains only minimal WordPress resources, such as having only one application server.

2. Test a basic cloud template

   During design, you often build a cloud template by starting with the essentials, then deploying and testing as the template grows. This example demonstrates some of the in-progress testing built into Cloud Assembly.

3. Expand a cloud template

   After you create and test the basic Cloud Assembly template for the example application, you expand it into a multiple tier application that is deployable to development, test, and eventually production.

Create a basic cloud template

In this Cloud Assembly design example, you start with a cloud template that contains only minimal WordPress resources, such as having only one application server.

Cloud Assembly is an infrastructure-as-code tool. You drag resources to the design canvas to get started. Then, you complete the details using the code editor to the right of the canvas.

The code editor allows you to type, cut, and paste code directly. If you're uncomfortable editing code, you can select a resource in the canvas, click the code editor Properties tab, and enter values there. Values that you enter appear in the code as if you had typed them directly.

Procedure

1. Go to Design > Cloud Templates and click New from > Blank canvas.

2. Name the cloud template Wordpress-BP.

3. Select the WordPress project, and click Create.

4. From the resources on the left of the cloud template design page, drag two cloud agnostic machines onto the canvas.

   The machines serve as WordPress application server (WebTier) and MySQL database server (DBTier).

5. On the right, edit the machine YAML code to add names, images, flavors, and constraint tags:
6 Drag a cloud agnostic network to the canvas, and edit its code:

```
WP-Network-Private:
  type: Cloud.Network
  properties:
    name: WP-Network-Private
    networkType: existing
```

7 Connect the machines to the network:

In the canvas, hover over the network block, click and hold the bubble where the line touches the block, drag to a machine block, and release.

When you create the connection lines, note that network code is automatically added to the machines in the editor.

8 Add user input prompting.

In some places, the example infrastructure was set up for multiple options. For example:

- Cloud zone environments for development, test, and production
- Flavor mappings for small, medium, and large machines
You might set a specific option directly in the cloud template, but a better approach is to let the user select the option at template deployment time. Prompting for user input lets you create one template that can be deployed many ways, instead of having many hard-coded templates.

a) Create an `inputs` section in the code so that users can select machine size and target environment at deployment time. Define the selectable values:

```yaml
inputs:
  env:
    type: string
    enum:
      - env:dev
      - env:prod
      - env:test
    default: env:dev
    title: Environment
    description: Target Environment
  size:
    type: string
    enum:
      - small
      - medium
      - large
    description: Size of Nodes
    title: Tier Machine Size
```

b) In the `resources` section of the code, add `${input.input-name}` code to prompt for the user selection:

```yaml
resources:
  WebTier:
    type: Cloud.Machine
    properties:
      name: wordpress
      image: ubuntu
      flavor: '${input.size}'
      constraints:
        - tag: '${input.env}'
      networks:
        - network: '${resource["WP-Network-Private"].id}'
  DBTier:
    type: Cloud.Machine
    properties:
      name: mysql
      image: ubuntu
      flavor: '${input.size}'
      constraints:
        - tag: '${input.env}'
      networks:
        - network: '${resource["WP-Network-Private"].id}'

WP-Network-Private:
```
Finally, enhance the WebTier and DBTier code using the following examples. The WP-Network-Private code does not need additional changes.

Note that the enhancements include login access to the database server and deployment-time cloudConfig initialization scripts.
<table>
<thead>
<tr>
<th>Component</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Additional DBTier Inputs</td>
<td></td>
</tr>
<tr>
<td>username:</td>
<td>type: string</td>
</tr>
<tr>
<td>minLength: 4</td>
<td>maxLength: 20</td>
</tr>
<tr>
<td>pattern: '[a-z]+'</td>
<td>title: Database Username</td>
</tr>
<tr>
<td>description: Database Username</td>
<td>userpassword:</td>
</tr>
<tr>
<td>type: string</td>
<td>pattern: '[a-z0-9A-Z@#$]+'</td>
</tr>
<tr>
<td>encrypted: true</td>
<td>title: Database Password</td>
</tr>
<tr>
<td>description: Database Password</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DBTier Resource</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>DBTier:</td>
<td>type: Cloud.Machine</td>
</tr>
<tr>
<td>properties:</td>
<td>name: mysql</td>
</tr>
<tr>
<td>image: ubuntu</td>
<td>flavor: '${input.size}'</td>
</tr>
<tr>
<td>constraints:</td>
<td>- tag: '${input.env}'</td>
</tr>
<tr>
<td>networks:</td>
<td>- network: '${resource[&quot;WP-Network-Private&quot;][id]}'</td>
</tr>
<tr>
<td>assignsPublicIPAddress: true</td>
<td>remoteAccess:</td>
</tr>
<tr>
<td>authentication: usernamePassword</td>
<td>username: '${input.username}'</td>
</tr>
<tr>
<td>password: '${input.userpassword}'</td>
<td></td>
</tr>
<tr>
<td>cloudConfig:</td>
<td></td>
</tr>
<tr>
<td>#cloud-config</td>
<td>repo_update: true</td>
</tr>
<tr>
<td>repo_upgrade: all</td>
<td>packages:</td>
</tr>
<tr>
<td>- mysql-server</td>
<td>runcmd:</td>
</tr>
<tr>
<td>- service mysql restart</td>
<td>- sed -e '/bind-address/ s/^#*/#/' -i /etc/mysql/mysql.conf.d/mysqld.cnf</td>
</tr>
<tr>
<td>- mysql -e &quot;GRANT ALL PRIVILEGES ON <em>.</em> TO 'root'@'%' IDENTIFIED BY 'mysqlpassword';&quot;</td>
<td></td>
</tr>
<tr>
<td>- mysql -e &quot;FLUSH PRIVILEGES;&quot;</td>
<td></td>
</tr>
<tr>
<td>attachedDisks: []</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>WebTier Resource</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>WebTier:</td>
<td>type: Cloud.Machine</td>
</tr>
<tr>
<td>properties:</td>
<td>name: wordpress</td>
</tr>
<tr>
<td>image: ubuntu</td>
<td>flavor: '${input.size}'</td>
</tr>
<tr>
<td>constraints:</td>
<td>- tag: '${input.env}'</td>
</tr>
<tr>
<td>networks:</td>
<td>- network: '${resource[&quot;WP-Network-Private&quot;][id]}'</td>
</tr>
<tr>
<td>assignsPublicIPAddress: true</td>
<td>cloudConfig:</td>
</tr>
</tbody>
</table>
Example: Completed basic cloud template code example

formatVersion: 1
inputs:
  env:
    type: string
    enum:
      - env:dev
      - env:prod
      - env:test
    default: env:dev
    title: Environment
    description: Target Environment
size:
  type: string
  enum:
    - small
    - medium
    - large
  description: Size of Nodes
  title: Tier Machine Size
username:
  type: string
  minLength: 4
  maxLength: 20

Example: Completed basic cloud template code example

```
repo_update: true
repo_upgrade: all
packages:
  - apache2
  - php
  - php-mysql
  - libapache2-mod-php
  - php-mcrypt
  - mysql-client
runcmd:
  - i=0; while [ $i -le 10 ]; do mysql --connect-timeout=3 -h ${DBTier.networks[0].address} -u root -p$mypassword -e "SHOW STATUS;" && break || sleep 15; i=$((i+1)); done
  - mysql -u root -p$mypassword -h ${DBTier.networks[0].address}
  -e "create database wordpress_blog;"
  - sed -i -e "s/'define( 'DB_NAME', 'database_name_here' );'/"define( 'DB_NAME', 'wordpress_blog' );"/ /
    /var/www/html/mywordpresssite/wp-config.php && sed -i -e "s/'define( 'DB_USER', 'username_here' );'/"define( 'DB_USER', 'root' );"/ /
    /var/www/html/mywordpresssite/wp-config.php && sed -i -e "s/'define( 'DB_PASSWORD', 'password_here' );'/"define( 'DB_PASSWORD', 'mysqlpassword' );"/ /
    /var/www/html/mywordpresssite/wp-config.php && sed -i -e "s/'define( 'DB_HOST', 'localhost' );'/"define( 'DB_HOST', '${DBTier.networks[0].address}' );"/ /
    /var/www/html/mywordpresssite/wp-config.php
  - service apache2 reload
```
pattern: '^[a-z]+'  
title: Database Username  
description: Database Username  
userpassword:  
type: string  
pattern: '^[a-z0-9A-Z@#$]+$'  
encrypted: true  
title: Database Password  
description: Database Password  
resources:  
WebTier:  
type: Cloud.Machine  
properties:  
  name: wordpress  
  image: ubuntu  
  flavor: '${input.size}'  
  constraints:  
    - tag: '${input.env}'  
  networks:  
    - network: '${resource["WP-Network-Private"].id}'  
      assignPublicIpAddress: true  
  cloudConfig: |  
  #cloud-config  
  repo_update: true  
  repo_upgrade: all  
  packages:  
    - apache2  
    - php  
    - php-mysql  
    - libapache2-mod-php  
    - php-mcrypt  
    - mysql-client  
  runcmd:  
    - i=0; while [ $i -le 10 ]; do mysql --connect-timeout=3 -h ${{DBTier.networks[0].address}} -u root -pmysqlpassword -e "SHOW STATUS;" && break || sleep 15; i=$((i+1)); done  
    - mysql -u root -pmysqlpassword -h ${DBTier.networks[0].address} -e "create database wordpress_blog;"  
    - sed -i -e s/'"define( 'DB_NAME', 'database_name_here' );"/"define( 'DB_NAME', 'wordpress_blog' );"/  
      /var/www/html/mywordpresssite/wp-config.php  
    - sed -i -e s/'"define( 'DB_USER', 'username_here' );"/"define( 'DB_USER', 'root' );"/  
      /var/www/html/mywordpresssite/wp-config.php  
    - sed -i -e s/'"define( 'DB_PASSWORD', 'password_here' );"/"define( 'DB_PASSWORD', 'mysqlpassword' );"/  
      /var/www/html/mywordpresssite/wp-config.php  
    - service apache2 reload  

DBTier:  
type: Cloud.Machine  
properties:
What to do next

Test the cloud template by checking the syntax and deploying it.

Test a basic cloud template

During design, you often build a cloud template by starting with the essentials, then deploying and testing as the template grows. This example demonstrates some of the in-progress testing built into Cloud Assembly.

To be certain that a deployment works the way that you want, you might test and deploy the cloud template several times. Gradually, you add more resources, retest, and redeploy along the way.

Prerequisites

Create the basic cloud template. See Create a basic cloud template.

Procedure

1. Click Cloud Templates, and open the WordPress-BP cloud template.

   The basic cloud template appears, in the design canvas and code editor.

2. To check template syntax, placement, and basic validity, click Test at the lower left.
3 Enter input values, and click Test.

![Basic Testing](image)

The test is only a simulation and does not actually deploy virtual machines or other resources.

![Test Result for Basic](image)

The test includes a link to a Provisioning Diagram, where you can inspect the simulated deployment flow and see what occurred. The simulation exposes potential issues, such as not having any resource capabilities defined that match hard constraints in the cloud template. In the example error that follows, a cloud zone of capability tag `env:dev` wasn’t found anywhere in the defined infrastructure.
A successful simulation doesn't guarantee that you can deploy the template without errors.

4 After the template passes the simulation, click **Deploy** at the lower left.

5 Select **Create a new deployment**.

6 Name the deployment **WordPress for OurCo** and click **Next**.

7 Enter input values, and click **Deploy**.

8 To verify that the template successfully deployed, look under **Deployments > Deployments**.

   If a deployment fails, click its name, and click the **History** tab to see messages that can help you troubleshoot.

   Some history entries might have the **Provisioning Diagram** link at the far right. The diagram is similar to the simulated one, where you inspect the flow chart of Cloud Assembly decision points in the provisioning process.

   More flow charts are available under **Infrastructure > Activity > Requests**.
9 To verify that the application is working, open the WordPress start page in a browser.
   a Wait for the WordPress servers to be fully created and initialized.
      It might take 30 minutes or more for initialization, depending on the environment.
   b To locate the site FQDN or IP address, go to Deployments > Deployments > Topology.
   c On the canvas, click the WebTier, and find the IP address in the panel on the right.
   d Enter the IP address as part of the full URL to the WordPress start page.
      In this example, the full URL is:
      http://{IP-address}/mywordpresssite
      or
      http://{IP-address}/mywordpresssite/wp-admin/install.php

10 After inspecting WordPress in a browser, if the application needs more work, make template changes and redeploy using the Update an existing deployment option.

11 Consider versioning the cloud template. You can revert to a working version if a change causes deployment to fail.
   a On the cloud template design page, click Version.
   b On the Creating Version page, enter WP-1.0.
      Do not enter spaces in version names.
   c Click Create.
      To review or revert to a version, on the design page, click the Version History tab.

12 With a basic deployment now possible, try your first deployment-time enhancement by increasing CPU and memory on the application and database servers.

   Update to a medium node size for both. Using the same template, select medium at deployment time, redeploy, and verify the application again.

What to do next

Expand the cloud template into a production-worthy application by adding even more resources.

Expand a cloud template

After you create and test the basic Cloud Assembly template for the example application, you expand it into a multiple tier application that is deployable to development, test, and eventually production.

To expand the cloud template, you add the following enhancements.

- An option to cluster application servers for increased capacity
- A public-facing network and load balancer in front of the application servers
A backup server with archive storage

**Prerequisites**

Create the basic cloud template and test it. See Create a basic cloud template and Test a basic cloud template.

**Procedure**

1. Click **Cloud Templates**, and open the WordPress-BP cloud template.
   - The basic template appears, in the design canvas and code editor.

2. Make additions and changes, using the code example and figure for guidance.
   - You use the GUI to drag new resources to the canvas, such as the load balancer, and then finish the configuration in the code editor.
   - a. Add a count input prompt to make the WordPress application server into a cluster.
   - b. Add a cloud agnostic load balancer.
   - c. Connect the load balancer to the WordPress application server cluster.
   - d. Add a cloud agnostic backup machine.
   - e. Connect the backup machine to the private/internal network.
   - f. Add a cloud agnostic public/external network.
   - g. Connect the load balancer to the public network.
   - h. Add a cloud agnostic storage volume for use as an archive disk.
   - i. Connect the archive disk to the backup machine.
   - j. Add an input prompt for the archive disk speed.

3. Deploy, test, and make changes in the same way that you did for the basic cloud template.
   - You can update existing deployments, or even deploy new instances so that you can compare deployments.
   - The goal is to reach a solid, repeatable template that can be used for production deployments.
Example: Completed expanded cloud template code example

```json
formatVersion: 1
inputs:
  env:
    type: string
    enum:
      - env:dev
      - env:prod
      - env:test
    default: env:dev
    title: Environment
    description: Target Environment
  size:
    type: string
    enum:
      - small
      - medium
      - large
    description: Size of Nodes
    title: Tier Machine Size
  username:
    type: string
    minLength: 4
    maxLength: 20
    pattern: '[a-z]+'  
    title: Database Username
    description: Database Username
  userpassword:
    type: string
    pattern: '[a-z0-9A-Z@#$]+'  
    encrypted: true
    title: Database Password
    description: Database Password
  count:
    type: integer
    default: 2
    maximum: 5
    minimum: 2
    title: WordPress Cluster Size
    description: WordPress Cluster Size (Number of Nodes)
  storagetype:
    type: string
    enum:
      - storage:general
      - storage:fast
    description: Archive Storage Disk Type
    title: Archive Disk Type
resources:
  WebTier:
    type: Cloud.Machine
    properties:
      name: wordpress
      image: ubuntu
```
flavor: '$_{input.size}''
count: '$_{input.count}''
constraints:
  - tag: '$_{input.env}''
networks:
  - network: '$_{resource["WP-Network-Private"]\[\].id}''
    assignPublicIpAddress: true
cloudConfig: |
    #cloud-config
    repo_update: true
    repo_upgrade: all
    packages:
      - apache2
      - php
      - php-mysql
      - libapache2-mod-php
      - php-mcrypt
      - mysql-client
    runcmd:
      - i=0; while [ $i -le 10 ]; do mysql --connect-timeout=3 -h ${DBTier.networks[0].address} -u root -pmysqlpassword -e "SHOW STATUS;" & & break || sleep 15; i=$((i+1)); done
      - mysql -u root -pmysqlpassword -h ${DBTier.networks[0].address} -e "create database wordpress_blog;"
      - sed -i -e "s/^define\( 'DB_NAME', 'database_name_here' \);"/"define\( 'DB_NAME', 'wordpress_blog' \);"/" /var/www/html/mywordpresssite/wp-config.php
      - sed -i -e "s/^\("DB_USER\), 'username_here' \);"/"define\( 'DB_USER', 'root' \);"/ /var/www/html/mywordpresssite/wp-config.php
      - sed -i -e "s/^\("DB_PASSWORD\), 'password_here' \);"/"define\( 'DB_PASSWORD', 'mysqlpassword' \);"/ /var/www/html/mywordpresssite/wp-config.php
      - service apache2 reload

DBTier:
type: Cloud.Machine
properties:
  name: mysql
  image: ubuntu
  flavor: '$_{input.size}''
  constraints:
    - tag: '$_{input.env}''
  networks:
    - network: '$_{resource["WP-Network-Private"]\[\].id}''
      assignPublicIpAddress: true
  remoteAccess:
    authentication: usernamePassword
    username: '$_{input.username}''
    password: '$_{input.userpassword}''
cloudConfig: |
    #cloud-config
repo_update: true
repo_upgrade: all
packages:
- mysql-server
runcmd:
- sed -e '/bind-address/ s/^#*/#/' -i /etc/mysql/mysql.conf.d/mysqld.cnf
- service mysql restart
- mysql -e "GRANT ALL PRIVILEGES ON *.* TO 'root'@'%' IDENTIFIED BY 'mysqlpassword';"
- mysql -e "FLUSH PRIVILEGES;"
attachedDisks: []
LoadBalancer:
type: Cloud.LoadBalancer
properties:
  name: myapp-lb
  network: '${resource["WP-Network-Public"].id}'
  instances:
    - '${WebTier.id}'
  routes:
    - protocol: HTTP
      port: '80'
      instanceProtocol: HTTP
      instancePort: '80'
      healthCheckConfiguration:
        protocol: HTTP
        port: '80'
        urlPath: /mywordpresssite/wp-admin/install.php
        intervalSeconds: 6
        timeoutSeconds: 5
        unhealthyThreshold: 2
        healthyThreshold: 2
        internetFacing: true
WP-Network-Private:
type: Cloud.Network
properties:
  name: WP-Network-Private
  networkType: existing
WP-Network-Public:
type: Cloud.Network
properties:
  name: WP-Network-Public
  networkType: public
backup:
type: Cloud.Machine
properties:
  name: backup
  flavor: '${input.size}'
  image: ubuntu
  networks:
    - network: '${resource["WP-Network-Private"].id}'
      attachedDisks:
        - source: '${resource.ArchiveDisk.id}'
ArchiveDisk:
type: Cloud.Volume
properties:
  name: ArchiveDisk
What to do next

Define your own infrastructure and create your own cloud templates.

See Chapter 4 Building your Cloud Assembly resource infrastructure and Chapter 6 Designing your Cloud Assembly deployments.

**Tutorial: Configuring VMware Cloud on AWS for vRealize Automation Cloud**

This Cloud Assembly tutorial shows the process of defining resource infrastructure and cloud template settings for deployment to a VMware Cloud on AWS environment.

The procedure requires that a cloud administrator has already configured your organization's VMware Cloud on AWS SDDC data center as described in *Deploying and Managing a Software-Defined Data Center* in the VMware Cloud on AWS Getting Started documentation.

If you are using the trial version, see the vRealize Automation Cloud trial for VMware Cloud on AWS users publication.

Look at the sequential setup to understand the process for configuring your environment for VMware Cloud on AWS. Remember that the values you see are only use case examples. Think about where you would make your own substitutions, or extrapolate from the example values, in order to fit your own cloud infrastructure and deployment needs.

A detailed video of a similar workflow is available from VMware Cloud Management Technical Marketing at How to Configure VMware Cloud on AWS for Cloud Assembly.

**Procedure**

1. **Configure a basic VMware Cloud on AWS workflow in vRealize Automation Cloud**
   
   This use case shows the process of defining resource infrastructure and a corresponding cloud template for deployment to a VMware Cloud on AWS environment.

2. **Configure an isolated network in VMware Cloud on AWS workflow in vRealize Automation Cloud**
   
   In this procedure, you add an isolated network for your VMware Cloud on AWS deployment in vRealize Automation Cloud.

**Configure a basic VMware Cloud on AWS workflow in vRealize Automation Cloud**

This use case shows the process of defining resource infrastructure and a corresponding cloud template for deployment to a VMware Cloud on AWS environment.
In this procedure, you deploy a new cloud proxy to your VMware Cloud on AWS SDDC in vCenter and then create a VMware Cloud on AWS cloud account that accesses the proxy. You also configure infrastructure that supports cloud template deployment to resources in your VMware Cloud on AWS environment.

**Prerequisites**

- Before you can create and configure a VMware Cloud on AWS cloud account in Cloud Assembly, you must be part of an organization in an existing VMware Cloud on AWS SDDC environment. For information about configuring the VMware Cloud on AWS service, see [VMware Cloud on AWS Documentation](#).

- Before you can use a deployed cloud proxy with a VMware Cloud on AWS cloud account, you must configure management gateway firewall rules in the SDDC’s VMware Cloud on AWS console to support cloud proxy communication. The rules must be in the Management Gateway firewall rules section. Create the firewall rules by using options on the Networking & Security tab in the SDDC console.

- Limit network traffic to ESXi for HTTPS (TCP 443) services to the discovered IP address of the cloud proxy. An example entry is shown in the following screen.

- Limit network traffic to vCenter for ICMP (All ICMP), SSO (TCP 7444), and HTTPS (TCP 443) services to the discovered IP address of the cloud proxy. An example entry is shown in the following screen.

- Limit network traffic to the NSX-T Manager for HTTPS (TCP 443) services to the discovered IP address of the cloud proxy. An example entry is shown in the following screen.

![Gateway Firewall Screen](image)

An example of how to select a source machine, such as CloudProxy1, is shown below:
For more information about how to configure gateway firewall rules in the SDDC's VMware Cloud on AWS console, see About VMware Cloud on AWS Networking with VMware NSX-T.

If you are using a static IP address for the cloud proxy, you can create the firewall rules that limit network traffic for the target vCenter and NSX Manager either before or after you deploy the cloud proxy.

**Procedure**

1. **Create and deploy a cloud proxy for a VMware Cloud on AWS cloud account in vRealize Automation Cloud**
   
   In this step, you deploy a cloud proxy to a vCenter server in a specified VMware Cloud on AWS Software-Defined Data Center (SDDC).

2. **Create a management rule to support VMware Cloud on AWS access to cloud proxy URLs in vRealize Automation Cloud**
   
   Configure a management rule in the SDDC's VMware Cloud on AWS console to support cloud proxy communication. Also configure the needed firewall rules that support access to required ports and protocols.

3. **Create a VMware Cloud on AWS cloud account in vRealize Automation Cloud within a sample workflow**
   
   In this step, you create a VMware Cloud on AWS cloud account in vRealize Automation Cloud.

4. **Create a cloud zone for VMware Cloud on AWS deployments in vRealize Automation Cloud**
   
   In this step, you create a cloud zone to specify a compute resource that the CloudAdmin user can access when working with VMware Cloud on AWS in vRealize Automation Cloud.

5. **Configure network and storage profiles for VMware Cloud on AWS deployments in vRealize Automation Cloud**
   
   In this step, you configure a network profile and a storage profile to specify resources that are available to a VMware Cloud on AWS CloudAdmin user in vRealize Automation Cloud.
Create a project to support VMware Cloud on AWS deployments in vRealize Automation Cloud

In this step, you define a vRealize Automation Cloud project that can be used to control which resources are available for VMware Cloud on AWS deployments.

Define a vCenter machine resource in a cloud template design to support VMware Cloud on AWS deployment in vRealize Automation Cloud

In this step, you drag a vCenter machine resource onto the design canvas and add settings for a VMware Cloud on AWS deployment in vRealize Automation Cloud.

Create and deploy a cloud proxy for a VMware Cloud on AWS cloud account in vRealize Automation Cloud

In this step, you deploy a cloud proxy to a vCenter server in a specified VMware Cloud on AWS Software-Defined Data Center (SDDC).

You can create a cloud proxy before you create a cloud account or during the process of creating a cloud account. Here, you'll create the cloud proxy first as a separate step. Later, you'll create a cloud account and choose this cloud proxy as part of that process.

Unless otherwise indicated, the step values that you enter in this procedure are for this example workflow only.

Prerequisites

This procedure assumes that you have VMware Cloud on AWS CloudAdmin credentials for the target SDDC in vCenter. See Credentials required for working with cloud accounts in vRealize Automation Cloud.

This procedure assumes that you have the cloud administrator user role. See What are the vRealize Automation Cloud user roles.

Verify that the required SDDC gateway firewall rules are configured in the VMware Cloud on AWS console. The gateway firewall rules enable the cloud proxy VA to communicate with the VMware Cloud on AWS SDDC. See Configure a basic VMware Cloud on AWS workflow in vRealize Automation Cloud.

To support the cloud proxy, allowed access to the following domains is required. For more information about the cloud proxy, see Understanding the VMware Cloud services cloud proxy.

- ci-data-collector.s3.amazonaws.com – enables Amazon Web Services S3 access for cloud proxy OVA download.
- symphony-docker-external.jfrog.io – allows JFrog Artifactory to access Docker images.
- data.mgmt.cloud.vmware.com – enables the data pipeline service connection to VMware Cloud services for secure data communication between cloud and on-premises elements. For non-US regions, substitute the region value. For example, for the UK, use uk.data.mgmt.cloud.vmware.com.
api.mgmt.cloud.vmware.com – enables the Web API and cloud proxy service connection to the VMware Cloud service. For non-US regions, substitute the region value. For example, for the UK, use uk.api.mgmt.cloud.vmware.com.

console.cloud.vmware.com – enables the Web API and cloud proxy service connection to the VMware Cloud service. For non-US regions, substitute the region value. For example, for the UK, use uk.console.cloud.vmware.com.

**Procedure**

1. Log in to Cloud Assembly and click **Infrastructure > Connections > Cloud Proxies.**
2. Click **New Cloud Proxy** to open the **Install Cloud Proxy** OVF deployment page.

3. Click **Download OVA.**

   The supplied **VMware-Cloud-Services-Data-Collector.ova** file is downloaded.

4. Rename the downloaded OVA to **vCenter1_vmc_va.ova.**

5. Navigate to your vSphere Web Client data center, click the name of your vCenter cluster, and select **Deploy OVF Template.**

6. Enter the **vCenter1_vmc_va.ova** OVA file name and location as prompted.

7. When asked to enter the key or token, return to the **Install Cloud Proxy** page and click **Copy.**

8. Return to the vSphere Web Client and paste the copied key value as prompted to run the cloud proxy virtual appliance.

9. In Cloud Assembly, wait for a connection to be made with your vSphere Web Client and then click **Done.**

   It might take several minutes to connect.
What to do next

To verify that the cloud proxy is running, see **Verify that a cloud proxy is running on a target virtual machine**.

You can now create the VMware Cloud on AWS cloud account. See **Create a VMware Cloud on AWS cloud account in vRealize Automation Cloud within a sample workflow**.

**Create a management rule to support VMware Cloud on AWS access to cloud proxy URLs in vRealize Automation Cloud**

Configure a management rule in the SDDC's VMware Cloud on AWS console to support cloud proxy communication. Also configure the needed firewall rules that support access to required ports and protocols.

To use VMware Cloud on AWS in Cloud Assembly cloud proxy, you must create compute gateway rules that support outbound access to the following allowed URLs:

- ci-data-collector.s3.amazonaws.com – enables Amazon Web Services S3 access for cloud proxy OVA download.
- symphony-docker-external.jfrog.io - allows JFrog Artifactory to access Docker images.
- data.mgmt.cloud.vmware.com - enables the data pipeline service connection to VMware Cloud services for secure data communication between cloud and on-premises elements. For non-US regions, substitute the region value. For example, for the UK, use uk.data.mgmt.cloud.vmware.com and for Japan, use ja.data.mgmt.cloud.vmware.com.
- api.mgmt.cloud.vmware.com – enables the Web API and cloud proxy service connection to the VMware Cloud service. For non-US regions, substitute the region value. For example, for the UK, use uk.data.mgmt.cloud.vmware.com and for Japan, use ja.data.mgmt.cloud.vmware.com.

**Prerequisites**

- This procedure is performed by a vCenter administrator using VMware Cloud on AWS administrator credentials in the SDDC's VMware Cloud on AWS console.
- Deploy the cloud proxy. See **Create and deploy a cloud proxy for a VMware Cloud on AWS cloud account in vRealize Automation Cloud**.

**Procedure**

1. Open the **Networking & Security** tab in the VMware Cloud on AWS console on the SDDC.

2. Create a management rule to allow outbound access to the following URLs:

   - ci-data-collector.s3.amazonaws.com
   - symphony-docker-external.jfrog.io
   - data.mgmt.cloud.vmware.com

   For non-US regions, substitute the region value. For example, for the UK, use uk.data.mgmt.cloud.vmware.com.
api.mgmt.cloud.vmware.com

For non-US regions, substitute the region value. For example, for the UK, use uk.api.mgmt.cloud.vmware.com.

Create a VMware Cloud on AWS cloud account in vRealize Automation Cloud within a sample workflow

In this step, you create a VMware Cloud on AWS cloud account in vRealize Automation Cloud.

For related information, see VMware Cloud on AWS documentation.

Unless otherwise indicated, the step values that you enter in this procedure are for this example workflow only.

Prerequisites

- This procedure assumes that you have the required administrator credentials, including VMware Cloud on AWS CloudAdmin credentials for the target SDDC in vCenter and that you have enabled HTTPS access on port 443. See Credentials required for working with cloud accounts in vRealize Automation Cloud.

- This procedure assumes that you have the cloud administrator user role. See What are the vRealize Automation Cloud user roles.

- Create and deploy a cloud proxy and use needed gateway firewall rules. See Create and deploy a cloud proxy for a VMware Cloud on AWS cloud account in vRealize Automation Cloud.

- Define a management rule on the VMware Cloud on AWS console to allow inbound access of the cloud proxy whitelist URLs. See Create a management rule to support VMware Cloud on AWS access to cloud proxy URLs in vRealize Automation Cloud.

- Verify that the cloud proxy VM is on and that the cloud proxy service is connected and running. See Verify that a cloud proxy is running on a target virtual machine.

Procedure

1. Select **Infrastructure > Connections > Cloud Accounts**.

2. Click **Add Cloud Account**, select VMware Cloud on AWS, and enter values.

   Sample values and supporting information are provided in the following table.
<table>
<thead>
<tr>
<th>Setting</th>
<th>Sample Value and Instruction</th>
<th>Description</th>
</tr>
</thead>
</table>
| VMC API Token                 | 1. Click the help icon at the end of the VMC API token line and click API Tokens page in the help text box to open the API Tokens tab on your organization’s My Account page.  
2. Click Generate Token to display the Generate a New API Token options.  
3. Enter a new token name, for example myinitials_mytoket.  
4. Set the Token TTL to never expire.  
If you create a token that is set to expire, then the VMware Cloud on AWS operations from vRealize Automation Cloud will stop working when the token expires and continue to not work until you update the cloud account with a new token.  
5. In the Define Scopes section, select All Roles.  
6. Click Generate.  
7. In the generated token page, click Copy and click Continue.  
8. Return to the New Cloud Account page, paste the copied token into the VMC API token row, and click Apply API token. | You can create a new token or use an existing token for your organization on the linked API Tokens page.  
In the Define Scopes section, the minimum required roles for the API token are:  
- Organizational Roles  
  - Organization Member  
  - Organization Owner  
- Service Roles - VMware Cloud on AWS  
  - Administrator  
  - NSX Cloud Administrator  
  - NSX Cloud Auditor  
Note Copy, download, or print the generated token. Once you leave this page you cannot retrieve the generated token.  
Apply the generated or supplied token to connect to the available SDDC environment in your organization’s VMware Cloud on AWS subscription and populate the list of SDDC names.  
If the vRealize Automation Cloud and VMware Cloud on AWS services are in different organizations, you should switch to the VMware Cloud on AWS organization and then generate the token.  
For more information about API tokens, see Generate API Tokens. |
| SDDC name                     | For this example, select Datacenter:Datacenter-abz.  
The valid SDDC name auto-populates the vCenter and NSX-T FQDN entries. If a cloud proxy was already deployed to the SDDC, the cloud proxy value also auto-populates. | Select from the list of available SDDCs from your VMware Cloud on AWS subscription. The list of SDDCs is based on the VMware Cloud on AWS API token.  
NSX-V SDDCs are not supported with vRealize Automation Cloud and do not appear in the list of available SDDCs. |

**Using and Managing VMware CloudAssembly**

VMware, Inc. 99
<table>
<thead>
<tr>
<th>Setting</th>
<th>Sample Value and Instruction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>vCenter IP address/FQDN</td>
<td>The address auto-populates based on your SDDC selection.</td>
<td>Enter the IP address or FQDN of the vCenter Server in the specified SDDC. The IP address defaults to the private IP address. Based on the type of network connectivity used to access your SDDC, the default address might be different than the IP address of the NSX Manager Server in the specified SDDC.</td>
</tr>
<tr>
<td>NSX Manager IP address/FQDN</td>
<td>The address auto-populates based on your SDDC selection.</td>
<td>Specifies the IP address or FQDN of the NSX Manager in the specified SDDC. The IP address defaults to the private IP address. Based on the type of network connectivity used to access your SDDC, the default address might be different than the IP address of the NSX Manager Server in the specified SDDC. VMware Cloud on AWS cloud accounts support NSX-T.</td>
</tr>
<tr>
<td>vCenter user name and password</td>
<td>The user name auto-populates as <a href="mailto:cloudadmin@vmc.local">cloudadmin@vmc.local</a>.</td>
<td>Enter your vCenter user name for the specified SDDC if it’s different than the default. The specified user requires CloudAdmin credentials. The user does not require CloudGlobalAdmin credentials. Enter the user password.</td>
</tr>
<tr>
<td>Cloud proxy</td>
<td>Select vCenter1_vmc_va.ova. This is the name of the example cloud proxy you deployed earlier in Create and deploy a cloud proxy for a VMware Cloud on AWS cloud account in vRealize Automation Cloud.</td>
<td></td>
</tr>
<tr>
<td>Validate</td>
<td>Click Validate.</td>
<td>Validate confirms your access rights to the specified vCenter and checks that the vCenter is running.</td>
</tr>
<tr>
<td>Name and Description</td>
<td>Enter OurCo-VMC for the cloud account name.</td>
<td></td>
</tr>
<tr>
<td>Allow provisioning to these data centers</td>
<td>This information is read-only.</td>
<td>Lists available data centers in your specified VMware Cloud on AWS SDDC environment.</td>
</tr>
</tbody>
</table>
### Setting | Sample Value and Instruction | Description
---|---|---
Create a cloud zone | De-select the check-box. For this example, you will create a cloud zone later in the workflow. | See [Learn more about Cloud Assembly cloud zones](#).
Capability tags | Leave this empty. This workflow does not use capability tags. | Use tags according to your organization's tag strategy. See [How do I use tags to manage Cloud Assembly resources and deployments](#) and [Creating a tagging strategy](#).

As with VMs deployed to vSphere, you can configure machine tags for a VM to be deployed on VMware Cloud on AWS. You can also update the machine tag after initial deployment. These machine tags allow vRealize Automation Cloud to dynamically assign a VM to an appropriate NSX-T security group during deployment. For related information, see [More about security group and tag resources in vRealize Automation Cloud cloud templates](#).

3. Click **Add**.

#### Results

Resources such as machines and volumes are data-collected from the VMware Cloud on AWS SDDC data center and listed in the **Resources** section of the vRealize Automation Cloud **Infrastructure** tab.

**What to do next**

*Create a cloud zone for VMware Cloud on AWS deployments in vRealize Automation Cloud.*

**Create a cloud zone for VMware Cloud on AWS deployments in vRealize Automation Cloud**

In this step, you create a cloud zone to specify a compute resource that the CloudAdmin user can access when working with VMware Cloud on AWS in vRealize Automation Cloud.

In VMware Cloud on AWS, the two primary administrator credentials are CloudGlobalAdmin and CloudAdmin. Cloud Assembly is designed to support the CloudAdmin user. Deploy to resources that are available to a VMware Cloud on AWS CloudAdmin user. Do not deploy to resources that require VMware Cloud on AWS CloudGlobalAdmin credentials.

Cloud zones identify the compute resources onto which a project cloud template deploys machines, networks, and storage. See [Learn more about Cloud Assembly cloud zones](#).

Unless otherwise indicated, the step values that you enter in this procedure are for this example workflow only.

#### Prerequisites

- Complete the [Create a VMware Cloud on AWS cloud account in vRealize Automation Cloud within a sample workflow procedure](#).
This procedure assumes that you have the required administrator credentials, including VMware Cloud on AWS CloudAdmin credentials for the target SDDC in vCenter. See Credentials required for working with cloud accounts in vRealize Automation Cloud.

This procedure assumes that you have the cloud administrator user role. See What are the vRealize Automation Cloud user roles.

**Procedure**

1. Select **Infrastructure > Configure > Cloud Zones**.
2. Click **New Cloud Zone**, and enter values for the VMware Cloud on AWS environment.

<table>
<thead>
<tr>
<th>Setting</th>
<th>Sample Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Account / region</td>
<td>OurCo-VMC / Datacenter:Datacenter-abz</td>
</tr>
<tr>
<td>Description</td>
<td>VMware Cloud on AWS resources only</td>
</tr>
<tr>
<td>Placement policy</td>
<td>Default</td>
</tr>
<tr>
<td>Capability tags</td>
<td>Leave this empty. This workflow does not use capability tags.</td>
</tr>
</tbody>
</table>

3. Click the **Compute** tab.

4. As shown in area 1 below, find and select a compute resource that is available to the CloudAdmin user. For this example, use the resource named Cluster 1/ Compute-ResourcePool.

Cluster 1/ Compute-ResourcePool is the default compute resource for VMware Cloud on AWS.
5 As shown in area 2 above, add the tag name `vmc_placements_abz`.

![Add tags](image)

6 Filter the compute resources that are used in this cloud zone by entering `vmc_placements_abz` in the **Filter tags** section.

![Filter tags](image)

7 Click **Save**.

For this example, only the compute resource named `Cluster 1/ Compute-ResourcePool` is available to the CloudAdmin user.

**What to do next**

**Configure network and storage profiles for VMware Cloud on AWS deployments in vRealize Automation Cloud.**

**Configure network and storage profiles for VMware Cloud on AWS deployments in vRealize Automation Cloud**

In this step, you configure a network profile and a storage profile to specify resources that are available to a VMware Cloud on AWS CloudAdmin user in vRealize Automation Cloud.

While an image and a flavor value are also needed, there is nothing unique about them specific to VMware Cloud on AWS user credentials. For this example, you'll use a flavor value of `small` and an image value of `ubuntu-16` when you define the cloud template.

For general information about mappings and profiles, see **Chapter 4 Building your Cloud Assembly resource infrastructure.**

Unless otherwise indicated, the step values that you enter in this procedure are for this example workflow only.

**Prerequisites**

- Create a cloud zone. See **Create a cloud zone for VMware Cloud on AWS deployments in vRealize Automation Cloud.**
This procedure assumes that you have the required administrator credentials, including VMware Cloud on AWS CloudAdmin credentials for the target SDDC in vCenter. See Credentials required for working with cloud accounts in vRealize Automation Cloud.

This procedure assumes that you have the cloud administrator user role. See What are the vRealize Automation Cloud user roles.

Procedure

1. Define a network profile for VMware Cloud on AWS deployments.
   a. Select Infrastructure > Configure > Network Profiles and click New Network Profile.

<table>
<thead>
<tr>
<th>Setting</th>
<th>Sample value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Account / region</td>
<td>OurCo-VMC / Datacenter:Datacenter-abz</td>
</tr>
<tr>
<td></td>
<td><strong>Note</strong> Select the VMware Cloud on AWS cloud account, and its matched SDDC data center, that you created in Create a VMware Cloud on AWS cloud account in vRealize Automation Cloud within a sample workflow.</td>
</tr>
<tr>
<td>Name</td>
<td>vmc-network1</td>
</tr>
<tr>
<td>Description</td>
<td>Contains networks that can be accessed by cloud template administrators who have VMware Cloud on AWS CloudAdmin credentials.</td>
</tr>
</tbody>
</table>

   b. Click the Network tab and click Add Network.

   c. Select a network that a VMware Cloud on AWS user with CloudAdmin credentials can deploy to, for example sddc-cgw-network-1.

2. Save the network profile.
3 Define a storage profile for VMware Cloud on AWS deployments.

Configure a storage profile that targets a datastore/cluster that is accessible to the CloudAdmin user.

a Select Infrastructure > Configure > Storage Profiles and click new New Storage Profile.

<table>
<thead>
<tr>
<th>Setting</th>
<th>Sample Value</th>
</tr>
</thead>
</table>
| Account / region | OurCo-VMC / Datacenter:Datacenter-abz  
Select the VMware Cloud on AWS cloud account, and its matched SDDC data center, that you created in Create a VMware Cloud on AWS cloud account in vRealize Automation Cloud within a sample workflow. |
| Name             | vmc-storage1                                                                                                                                 |
| Description      | Contains the datastore cluster that can be deployed to by cloud template administrators who have VMware Cloud on AWS CloudAdmin credentials.       |

b From the Datastore / Cluster drop-down menu, select the WorkloadDatastore datastore.

For VMware Cloud on AWS in Cloud Assembly, the storage policy must use the WorkloadDatastore datastore to support VMware Cloud on AWS deployment.

4 Save the storage profile.

What to do next

Create a project to support VMware Cloud on AWS deployments in vRealize Automation Cloud.

Create a project to support VMware Cloud on AWS deployments in vRealize Automation Cloud

In this step, you define a vRealize Automation Cloud project that can be used to control which resources are available for VMware Cloud on AWS deployments.

For information about projects, see How do Cloud Assembly projects work at deployment time.

Unless otherwise indicated, the step values that you enter in this procedure are for this example workflow only.

Prerequisites

- Complete the Configure network and storage profiles for VMware Cloud on AWS deployments in vRealize Automation Cloud procedure.
This procedure assumes that you have the required administrator credentials, including VMware Cloud on AWS CloudAdmin credentials for the target SDDC in vCenter. See Credentials required for working with cloud accounts in vRealize Automation Cloud.

This procedure assumes that you have the cloud administrator user role. See What are the vRealize Automation Cloud user roles.

Procedure

1. Select Infrastructure > Administration > Projects.
2. Click New Project and enter the project name VMC_proj-1_abz.
3. Click Users and click Add Users.
   - The users need CloudAdmin credentials to their organization's VMware Cloud on AWS subscription.
     - chris.gray@ourco.com, Administrator
     - kerry.white@ourco.com, Member
4. Click Provisioning and then click Add Cloud Zone.
5. Add the cloud zone that you configured in the earlier step.

<table>
<thead>
<tr>
<th>Setting</th>
<th>Sample Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cloud zone</td>
<td>VMC_cloud_zone-1</td>
</tr>
<tr>
<td></td>
<td>You created this cloud zone in the earlier step. Create a cloud zone for VMware Cloud on AWS deployments in vRealize Automation Cloud.</td>
</tr>
<tr>
<td>Provisioning priority</td>
<td>1</td>
</tr>
<tr>
<td>Instances limit</td>
<td>3</td>
</tr>
</tbody>
</table>

6. For this example, ignore the other options.

What to do next

Create a cloud template to deploy in your VMware Cloud on AWS environment. See Define a vCenter machine resource in a cloud template design to support VMware Cloud on AWS deployment in vRealize Automation Cloud.

**Define a vCenter machine resource in a cloud template design to support VMware Cloud on AWS deployment in vRealize Automation Cloud**

In this step, you drag a vCenter machine resource onto the design canvas and add settings for a VMware Cloud on AWS deployment in vRealize Automation Cloud.

Create a cloud template design that you can deploy to available VMware Cloud on AWS resources.
Unless otherwise indicated, the step values that you enter in this procedure are for this example workflow only.

**Prerequisites**

- This procedure assumes that you have cloud template designer credentials. See [What are the vRealize Automation Cloud user roles](#).
- This procedure assumes that you have [VMware Cloud on AWS CloudAdmin credentials](#) for the target SDDC in vCenter (Datacenter:Datacenter-abz). See [Credentials required for working with cloud accounts in vRealize Automation Cloud](#).
- Configure the resource infrastructure and project as described in the preceding sections.

**Procedure**

1. Click the **Design** tab and then click **New**.

<table>
<thead>
<tr>
<th>Setting</th>
<th>Sample Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>vmc-bp_abz</td>
</tr>
<tr>
<td>Description</td>
<td>1</td>
</tr>
<tr>
<td>Project</td>
<td>VMC_proj-1_abz</td>
</tr>
</tbody>
</table>
   
   This is the project that you created earlier, which supports the cloud zone that you also created earlier. The project is now associated with the cloud zone, which in turn is associated with the VMware Cloud on AWS cloud account/region that you created earlier.

2. Slide a vSphere machine resource onto the canvas.

3. Edit the following (bold) cloud template resource code in the machine resource.

   ```json
   formatVersion: 1
   inputs: {}
   resources:
     Cloud_vSphere_Machine_1:
       type: Cloud.vSphere.Machine
       properties:
         image: ubuntu-1604
         cpuCount: 1
         totalMemoryMB: 1024
         folderName: Workloads
   ```

   The **image** can be any value that is appropriate to your deployment needs.

   You must add the **folderName: Workloads** statement to the cloud template design code to support VMware Cloud on AWS deployment. The **folderName: Workloads** setting supports the **CloudAdmin credentials** in the VMware Cloud on AWS SDDC environment and is required.
Note: While the `folderName: Workloads` setting shown in the above code sample is required, you can add it directly in the cloud template code as shown above or you can add it in the associated cloud zone or project. If the setting is specified in more than one of these three places, the precedence is as follows:

- The project setting overrides the cloud template setting and the cloud zone setting.
- The cloud template setting overrides the cloud zone setting.

Note: You can optionally replace the `cpuCount` and `totalMemoryMB` settings with a `flavor` (sizing) entry, as shown below:

```json
formatVersion: 1
inputs: {}
resources:
  Cloud_vSphere_Machine_1:
    type: Cloud.vSphere.Machine
    properties:
      image: ubuntu-1604
      flavor: small
      folderName: Workloads
```

If the cloud zone has the folder value set to `Workloads`, you do not need to set the `folderName` property in the cloud template, unless you want to override the cloud zone folder value.

What to do next

Expand on this basic VMware Cloud on AWS workflow by adding network isolation. See Configure an isolated network in VMware Cloud on AWS workflow in vRealize Automation Cloud.

Configure an isolated network in VMware Cloud on AWS workflow in vRealize Automation Cloud

In this procedure, you add an isolated network for your VMware Cloud on AWS deployment in vRealize Automation Cloud.

When you define your VMware Cloud on AWS cloud account, NSX-T settings configured in your VMware Cloud on AWS service are available. For information about configuring NSX-T settings in your VMware Cloud on AWS service, see VMware Cloud on AWS product documentation.

vRealize Automation Cloud supports VMware Cloud on AWS with NSX-T. It does not support VMware Cloud on AWS with NSX-V.

vRealize Automation Cloud supports network isolation for VMware Cloud on AWS deployments. It does not support other network methods for VMware Cloud on AWS.

This extension of the basic VMware Cloud on AWS workflow describes the following methods of creating an isolated network for use in your cloud template:

- Configure on-demand network-based isolation.
- Configure on-demand security group-based isolation.
Prerequisites

This procedure expands on the basic VMware Cloud on AWS workflow. It uses the same cloud account and region, cloud zone, project, and network profile that you configured in the Tutorial: Configuring VMware Cloud on AWS for vRealize Automation Cloud workflow.

Procedure

1 Define an isolated network for a VMware Cloud on AWS deployment in vRealize Automation Cloud

You can configure network isolation for a VMware Cloud on AWS deployment by using either of the following procedures:

2 Define a network component in a cloud template to support network isolation for VMware Cloud on AWS in vRealize Automation Cloud

In this step, you drag a network machine component onto a vRealize Automation Cloud cloud template canvas and add settings for an isolated network deployment to your target VMware Cloud on AWS environment.

Define an isolated network for a VMware Cloud on AWS deployment in vRealize Automation Cloud

You can configure network isolation for a VMware Cloud on AWS deployment by using either of the following procedures:

- Configure on-demand network-based isolation in vRealize Automation Cloud
- Configure on-demand security group-based isolation in vRealize Automation Cloud

Configure on-demand network-based isolation in vRealize Automation Cloud

You can configure network isolation for your VMware Cloud on AWS deployment needs by specifying and using on-demand network settings in a network profile.

You can specify an isolated network by using a security group or by using on-demand network settings. In this example, you configure network isolation by specifying on-demand network settings in the network profile. Later, you access the network in a cloud template and use the cloud template in a VMware Cloud on AWS deployment.

Unless otherwise indicated, the step values that you enter in this procedure are for this example workflow only.

Prerequisites

- Complete the Configure a basic VMware Cloud on AWS workflow in vRealize Automation Cloud workflow.
- Review Configure an isolated network in VMware Cloud on AWS workflow in vRealize Automation Cloud.
This procedure assumes that you have the required administrator credentials, including VMware Cloud on AWS CloudAdmin credentials for the target SDDC in vCenter. See Credentials required for working with cloud accounts in vRealize Automation Cloud.

This procedure assumes that you have the cloud administrator user role. See What are the vRealize Automation Cloud user roles.

Procedure

1. Open the network profile that you used in the basic VMware Cloud on AWS workflow, for example vmc-network1. See Configure network and storage profiles for VMware Cloud on AWS deployments in vRealize Automation Cloud.

2. You do not need to make any selections on the Networks tab.

3. Click the Network Policies tab.

4. Select the Create an on-demand network option and select the default cgw network domain. Specify an appropriate CIDR and subnet size.

5. Click Save.

When you use this network profile, machines are deployed to a network in the default network domain. The network is isolated from other networks by using private or outbound network access.

What to do next

Configure a network component in your cloud template. See Define a network component in a cloud template to support network isolation for VMware Cloud on AWS in vRealize Automation Cloud

Configure on-demand security group-based isolation in vRealize Automation Cloud

You can configure network isolation for your VMware Cloud on AWS deployment needs by specifying and using an on-demand security group in a network profile.

You can specify an isolated network by using a security group or by using on-demand network settings. In this example, you configure network isolation by specifying an on-demand security group in the network profile. Later, you specify the network in a cloud template and use the cloud template in a VMware Cloud on AWS deployment.

Unless otherwise indicated, the step values that you enter in this procedure are for this example workflow only.

Prerequisites

- Complete the Configure a basic VMware Cloud on AWS workflow in vRealize Automation Cloud workflow.
- Review Configure an isolated network in VMware Cloud on AWS workflow in vRealize Automation Cloud.
This procedure assumes that you have the required administrator credentials, including VMware Cloud on AWS CloudAdmin credentials for the target SDDC in vCenter. See Credentials required for working with cloud accounts in vRealize Automation Cloud.

This procedure assumes that you have the cloud administrator user role. See What are the vRealize Automation Cloud user roles.

Procedure

1. Open the network profile that you used in the basic VMware Cloud on AWS workflow, for example vmc-network1. See Configure network and storage profiles for VMware Cloud on AWS deployments in vRealize Automation Cloud.

2. Select the existing network that you used in the basic VMware Cloud on AWS workflow, for example sddc-cgw-network-1. See Configure network and storage profiles for VMware Cloud on AWS deployments in vRealize Automation Cloud.

3. Click the Network Policies tab.

4. Select the Create an on-demand security group option.

5. Click Save.

When you use this network profile, machines are deployed to the selected network and are isolated by a new security group policy. The new security policy allows private or outbound network access.

What to do next

Configure a network component in your cloud template. See Define a network component in a cloud template to support network isolation for VMware Cloud on AWS in vRealize Automation Cloud

Define a network component in a cloud template to support network isolation for VMware Cloud on AWS in vRealize Automation Cloud

In this step, you drag a network machine component onto a vRealize Automation Cloud cloud template canvas and add settings for an isolated network deployment to your target VMware Cloud on AWS environment.
Add network isolation to the cloud template that you created earlier. The cloud template is already associated with a project and cloud zone that support deployment to your VMware Cloud on AWS environment, as well as the network profile and network that you configured for isolation.

Unless otherwise indicated, the step values that you enter in this procedure are for this example workflow only.

Prerequisites

- Complete the Configure on-demand security group-based isolation in vRealize Automation Cloud or Configure on-demand network-based isolation in vRealize Automation Cloud procedure.
- This procedure assumes that you have cloud template designer credentials. See What are the vRealize Automation Cloud user roles.
- This procedure assumes that you have VMware Cloud on AWS CloudAdmin credentials for the target SDDC in vCenter. See Credentials required for working with cloud accounts in vRealize Automation Cloud.

Procedure

1. Open the cloud template that you created in the previous workflow. See Define a vCenter machine resource in a cloud template design to support VMware Cloud on AWS deployment in vRealize Automation Cloud.

2. From the components on the left of the cloud template design page, drag a network component onto the canvas.

3. Edit the network component YAML code to specify a network type of either private or outbound, as shown in bold.

```
resources:  Cloud_Network_1:
    type: Cloud.Network
    properties:
        name: vmc_isolated
        networkType: private
```

OR

```
resources:  Cloud_Network_1:
    type: Cloud.Network
    properties:
        name: vmc_isolated
        networkType: outbound
```

What to do next

You are ready to deploy or close the cloud template.
Tutorial: Configuring a provider-specific external IPAM integration for vRealize Automation Cloud

You can use an external IPAM provider to manage IP address assignments for your cloud template deployments. This tutorial describes how to configure external IPAM integration in vRealize Automation Cloud using Infoblox as the external IPAM provider.

In this procedure, you use an existing IPAM provider package, in this case an Infoblox package, and an existing running environment to build a provider-specific IPAM integration point. You configure an existing network and create a network profile to support IP address allocation from the external IPAM provider. Finally, you create a cloud template that is matched to the network and network profile and deploy networked machines using IP values obtained from the external IPAM provider.

Information about how to obtain and configure the IPAM provider package, and how to configure a running environment that accesses a cloud extensibility proxy to support the IPAM provider integration, is included as reference.

Remember that the values you see are example values. You won't be able to use them letter-by-letter in your environment. Think about where you would make your own substitutions, or extrapolate from the example values, to fit your organization's needs.

To reference a similar vRealize Automation Cloud scenario that illustrates an Infoblox IPAM integration workflow in video form, see Infoblox IPAM Plug-in 1.1 Integration with vRealize Automation 8.1 / vRealize Automation Cloud.

Procedure

1. **Add required extensible attributes in the Infoblox application for integration with vRealize Automation Cloud**
   Before you can download and deploy the Infoblox provider package (infoblox.zip) for integration with vRealize Automation Cloud from either the Infoblox website or from the VMware Marketplace, you must add required extensibility attributes in Infoblox.

2. **Download and deploy an external IPAM provider package for use in vRealize Automation Cloud**
   Before you can define an external IPAM integration point in vRealize Automation Cloud, you need a configured IPAM provider package.

3. **Create a running environment for an IPAM integration point in vRealize Automation Cloud**
   Before you can define a external IPAM integration point in vRealize Automation Cloud, you need to create or access an existing running environment to serve as an intermediary between the IPAM provider and vRealize Automation Cloud. The running environment is commonly an Amazon Web Services or Microsoft Azure cloud account or an on-premises actions-based extensibility integration point that is associated to a cloud extensibility proxy.
4 Add an external IPAM integration for Infoblox in vRealize Automation Cloud

vRealize Automation Cloud supports integration with an external IPAM provider. This example uses Infoblox as the external IPAM provider.

5 Configure a network and network profile to use external IPAM for an existing network in vRealize Automation Cloud

You can define an existing network to use IP address values that are obtained from, and managed by, an external IPAM provider rather than internally from vRealize Automation Cloud.

6 Define and deploy a cloud template that uses an external IPAM provider range assignment in vRealize Automation Cloud

You can define a cloud template to obtain and manage IP address assignments from your external IPAM provider. This example uses Infoblox as the external IPAM provider.

7 Using Infoblox-specific properties and extensible attributes for IPAM integrations in vRealize Automation Cloud cloud templates

You can use Infoblox-specific properties for vRealize Automation Cloud projects that contain external IPAM integrations for Infoblox.

8 Control network data collection by using Infoblox filters in vRealize Automation Cloud

For Infoblox, you can limit the number of data collected networks to only those networks that are needed for vRealize Automation Cloud operations. This reduces the amount of transferred data and enhances system performance.

Add required extensible attributes in the Infoblox application for integration with vRealize Automation Cloud

Before you can download and deploy the Infoblox provider package (infoblox.zip) for integration with vRealize Automation Cloud from either the Infoblox website or from the VMware Marketplace, you must add required extensibility attributes in Infoblox.

This procedure is applicable if you are creating an external IPAM integration point for Infoblox integration with Cloud Assembly.

Before you can use the infoblox.zip download, you must log in to your Infoblox account, using your organization account administrator credentials, and pre-create the following Infoblox extensible attributes:

- VMware NIC index
- VMware resource ID

Prerequisites

- Verify that you have an account with Infoblox and that you have the correct access credentials to your organization's Infoblox account.

- Confirm that the Infoblox WAPI version is supported. IPAM integration with Infoblox depends on Infoblox WAPI version v2.7. All Infoblox appliances that support WAPI v2.7 are supported.

Procedure

1 Log in to your Infoblox account using administrator credentials.

These are the same administrator user name and password credentials that you specify when you create an external IPAM integration point in Cloud Assembly using the Infrastructure > Connections > Integrations > menu sequence.

2 Use the procedure described in the Infoblox documentation to create the following required extensible attributes in your Infoblox application.

- VMware NIC index - type Integer
- VMware resource ID - type String

The procedure is described in the Adding Extensible Attributes section of the Infoblox documentation topic About Extensible Attributes. Also see Managing Extensible Attributes.

What to do next

After you add the required attributes, you can resume the process of downloading and deploying the Infoblox package as described in Download and deploy an external IPAM provider package for use in vRealize Automation Cloud.

Download and deploy an external IPAM provider package for use in vRealize Automation Cloud

Before you can define an external IPAM integration point in vRealize Automation Cloud, you need a configured IPAM provider package.
You can download a provider-specific integration package from your IPAM provider's website, from the VMware solutions exchange marketplace or, if available, from the vRealize Automation Cloud Marketplace tab.

**Note** This example uses the VMware-supplied Infoblox package Infoblox.zip, which is available for download from VMware Marketplace as follows. While all packages are compatible with earlier vRealize Automation Cloud iterations, VMware recommends that you use the latest Infoblox package for the current vRealize Automation Cloud iteration.

- **Infoblox plug-in version 1.4** - Compatible with the current vRealize Automation Cloud iteration and providing all the functionality of previous versions. With this version, you can use the same host name with a different DNS suffix for two NICs. See plug-in release notes for details.
- **Infoblox plug-in version 1.3** - Compatible with the current vRealize Automation Cloud iteration and providing additional network data collection filters. See Control network data collection by using Infoblox filters in vRealize Automation Cloud.
- **Infoblox plugin version 1.2**
- **Infoblox plugin version 1.1**
- **Infoblox plugin version 1.0**
- **Infoblox plugin version 0.4**

IPAM integration with Infoblox depends on Infoblox WAPI version v2.7. All Infoblox appliances that support WAPI v2.7 are supported.

For information about how to create an IPAM integration package for other IPAM providers, if one does not already exist in the Marketplace, see How do I use the IPAM SDK to create a provider-specific external IPAM integration package for vRealize Automation Cloud.

The IPAM provider package contains scripts that are packaged with metadata and other configurations. The scripts contain the source code used for the operations that vRealize Automation Cloud performs in coordination with the external IPAM provider. Example operations include Allocate an IP address for a virtual machine, Fetch a list of IP ranges from the provider, and Update the MAC address of a host record in the provider.

**Prerequisites**

- Verify that you have cloud administrator credentials. See Credentials required for working with cloud accounts in vRealize Automation Cloud.
- Verify that you have the cloud administrator user role. See What are the vRealize Automation Cloud user roles.
- Verify that you have an account with the external IPAM provider, for example Infoblox or Bluecat, and that you have the correct access credentials to your organization's account with the IPAM provider.
If you are using Infoblox as your external IPAM provider, verify that you have added the required extensible attributes to your Infoblox account before continuing. See Add required extensible attributes in the Infoblox application for integration with vRealize Automation Cloud.

**Note**  A certificate chain issue exists that is derived from how the Python element in the Infoblox plug-in handles SSL handshakes. For information about the issue and its required actions, see Knowledge Base Article vRA Cloud Infoblox Plugin throws a certificate chain error during authentication process (88057).

**Procedure**

1. Navigate to the correct Infoblox plug-in page, for example Infoblox plugin version 1.4. See above for the Infoblox plugin options that are available in the VMware Marketplace.

2. Log in and download the plug-in package.

3. If you have not already done so, add the required extensible attributes in Infoblox. See Add required extensible attributes in the Infoblox application for integration with vRealize Automation Cloud.

**Results**

The package is now available for you to deploy by using the Integrations > Add Integration > IPAM > Manage Providers > Import package menu sequence as described in Add an external IPAM integration for Infoblox in vRealize Automation Cloud.

Create a running environment for an IPAM integration point in vRealize Automation Cloud

Before you can define a external IPAM integration point in vRealize Automation Cloud, you need to create or access an existing running environment to serve as an intermediary between the IPAM provider and vRealize Automation Cloud. The running environment is commonly an Amazon Web Services or Microsoft Azure cloud account or an on-premises actions-based extensibility integration point that is associated to a cloud extensibility proxy.

External IPAM integration requires a running environment. When you define the IPAM integration point, you create a connection between Cloud Assembly and your IPAM provider by specifying an available running environment.

IPAM integration uses a set of downloaded provider-specific scripts or plug-ins in a running environment that is facilitated by a Feature-as-a-Services (FaaS) provider such as Amazon Web Services Lambda, Microsoft Azure Functions or an on-premises actions-based extensibility (ABX) integration point that is associated to a cloud extensibility proxy. The running environment is used to connect to the external IPAM provider, for example Infoblox.

**Note**  An Infoblox IPAM integration point requires an on-premises actions-based extensibility (ABX) point that is associated to a deployed cloud extensibility proxy in vCenter.
Each type of runtime environment has advantages and disadvantages:

- Actions-based extensibility (ABX) integration point
  - free, no additional vendor usage costs
  - can connect to IPAM vendor appliances that reside in an on-premises data center behind a NAT/firewall that is not publicly accessible, for example Infoblox
  - slower and slightly less reliable performance than commercial cloud vendors
  - must be associated to a cloud extensibility proxy

- Amazon Web Services
  - has associated vendor FaaS connection/usage costs
  - cannot connect to IPAM vendor appliances that reside in an on-premises data center behind a NAT/firewall that is not publicly accessible
  - has fast and highly reliable performance
  - is not associated to, and does not require, a cloud extensibility proxy

- Microsoft Azure
  - has associated vendor FaaS connection/usage costs
  - cannot connect to IPAM vendor appliances that reside in an on-premises data center behind a NAT/firewall that is not publicly accessible
  - has fast and highly reliable performance
  - is not associated to, and does not require, a cloud extensibility proxy

Prerequisites

- Verify that you have cloud administrator credentials. See Credentials required for working with cloud accounts in vRealize Automation Cloud.

- Verify that you have the cloud administrator user role. See What are the vRealize Automation Cloud user roles.

- Verify that you have an account with the external IPAM provider, for example Infoblox or Bluecat, and that you have the correct access credentials to your organization's account with the IPAM provider.

- Verify that you have access to a deployed integration package for your IPAM provider, such as Infoblox or BlueCat. The deployed package is initially obtained as a .zip download from your IPAM provider website or from the Cloud Assembly Marketplace and then deployed in Cloud Assembly.

For information about how to deploy the provider package .zip file and make it available as a Provider value on the IPAM Integration page, see Download and deploy an external IPAM provider package for use in vRealize Automation Cloud.
Procedure

To create the example running environment that is used in this sample workflow, create an on-premises actions-based extensibility integration point as described in Configure an on-premises action-based extensibility integration in Cloud Assembly. For information about how to download and deploy a cloud extensibility proxy for use with the on-premises actions-based extensibility integration point, see Download and deploy a cloud extensibility proxy.

For related information about the running environment, see this Infoblox IPAM Plug-in 1.1 Integration blog video at approximately 24 minutes into the video.

Add an external IPAM integration for Infoblox in vRealize Automation Cloud

vRealize Automation Cloud supports integration with an external IPAM provider. This example uses Infoblox as the external IPAM provider. You can use a provider-specific IPAM integration point to obtain and manage IP addresses and related network characteristics for cloud template deployments.

In this example, you create an external IPAM integration point to support access to your organization's account with an external IPAM provider. In this example workflow, the IPAM provider is Infoblox and the provider-specific integration package already exists. While these instructions are specific to an Infoblox integration, they can be used as reference if creating an IPAM integration for a different external IPAM provider.

You can obtain a provider-specific integration package from your IPAM provider's website, from the VMware solutions exchange marketplace or, if available, from the Cloud Assembly Marketplace tab.

This example uses the VMware-supplied Infoblox package Infoblox.zip, which is available for download from the VMware solutions exchange marketplace as follows:

- vRA Cloud Infoblox plugin version 1.1 - supports latest vRealize Automation Cloud iteration
- vRA Cloud Infoblox plugin version 1.0 - supports earlier vRealize Automation Cloud iterations
- vRA Cloud Infoblox plugin version 0.1 - supports earlier vRealize Automation Cloud iterations

Prerequisites

- Verify that you have cloud administrator credentials. See Credentials required for working with cloud accounts in vRealize Automation Cloud.
- Verify that you have the cloud administrator user role. See What are the vRealize Automation Cloud user roles.
- Verify that you have an account with external IPAM provider and that you have the correct access credentials to your organization's account with the IPAM provider.
Verify that you have access to a deployed integration package for your IPAM provider. The deployed package is initially obtained as a .zip download from your IPAM provider website, or from the VMware solutions exchange marketplace, and then deployed to vRealize Automation Cloud.

For information about how to download and deploy the provider package .zip file and make it available as a Provider value on the IPAM Integration page, see Download and deploy an external IPAM provider package for use in vRealize Automation Cloud.

Verify that you have access to a configured running environment for the IPAM provider. The running environment is typically a cloud account or integration point that contains an associated cloud extensibility endpoint that has been deployed to the target vCenter.

For information about running environment characteristics, see Create a running environment for an IPAM integration point in vRealize Automation Cloud.

Enable required extensible attributes in your Infoblox application. See Add required extensible attributes in the Infoblox application for integration with vRealize Automation Cloud.

Verify that you have the required user credentials to access and use your Infoblox IPAM product. For example, open the Administration tab in the Infoblox appliance and customize administrator, groups, and roles entries. You must be a member of a group that has administrator or superuser permissions or a custom group that has DHCP, DNS, IPAM, and Grid permissions. These settings allow access to all the functionality that is available in the Infoblox plug-in, enabling you to create an Infoblox IPAM integration and designers to use that IPAM integration in cloud templates and deployments. For more information about user permissions, see your Infoblox product documentation.

Procedure

1. Select Infrastructure > Connections > Integrations and click Add Integration.

2. Click IPAM.

3. In the Provider drop-down, select a configured IPAM provider package from the list, for example Infoblox_hrg.

If the list is empty, click Import Provider Package, navigate to an existing provider package .zip file, and select it. If you do not have the provider .zip file, you can obtain it from your IPAM provider’s web site or from the Cloud Assembly Marketplace tab.

For information about how to deploy the provider package .zip file in vCenter and make it available as a Provider value on the Integration page, see Download and center deploy an external IPAM provider package for use in vRealize Automation Cloud.

For information about how to upgrade an existing IPAM integration to use a more recent version of a vendor’s IPAM integration package, see How to upgrade to a newer external IPAM integration package in vRealize Automation Cloud.
4 Enter your administrator user name and password credentials for your account with the external IPAM provider, along with all other (if any) mandatory fields, such as the host name of your provider.

In this example, you obtain the host name of your Infoblox IPAM provider using the following steps:

a In a separate browser tab, log in to your IPAM provider account using your Infoblox administrator credentials.

b Copy your host name URL.

c Paste your host name URL in the Hostname field on the IPAM Integration page.

5 In the Running Environment drop-down list, select an existing on-premises actions-based extensibility integration point, for example Infoblox_abx_intg.

The running environment supports communication between vRealize Automation Cloud and the external IPAM provider.

**Note** If you use an Amazon Web Services or Microsoft Azure cloud account as the integration running environment, be sure that the IPAM provider appliance is accessible from the Internet and is not behind a NAT or firewall and that it has a publicly resolvable DNS name. If the IPAM provider is not accessible, the Amazon Web Services Lambda or Microsoft Azure Functions cannot connect to it and the integration will fail. For related information, see Create a running environment for an IPAM integration point in vRealize Automation Cloud.

The IPAM framework supports any of the following three types of running environments:

- an on-premises actions-based extensibility (ABX) integration point
- an Amazon Web Services (FaaS) cloud account
- a Microsoft Azure (FaaS) cloud account

**Note** An Infoblox IPAM integration point requires an on-premises actions-based extensibility (ABX) point that is associated to a deployed cloud extensibility proxy in vCenter.

Infoblox appliances are typically located in on-premises data centers that are not publicly accessible from the Internet. This make an on-premises actions-based extensibility integration point the logical choice for an Infoblox running environment. Using an on-premises actions-based extensibility integration point allows you to deploy the cloud extensibility proxy in your own data center and open a tunnel link between Cloud Assembly and Infoblox through the cloud extensibility proxy.

For information about how to create the example on-premises actions-based extensibility integration point used in this workflow, see Configure an on-premises action-based extensibility integration in Cloud Assembly.
The configured cloud account or integration point allows communication between vRealize Automation Cloud and the IPAM provider, in this example Infoblox, through an associated cloud extensibility proxy. You can select a provider that has already been created or you can create one.

For information about how to create a running environment, see Create a running environment for an IPAM integration point in vRealize Automation Cloud.

6 Click Validate.

Because this example uses the on-premises actions-based extensibility integration for the running environment, you can view the validation action.

   a Click the Extensibility tab.

   b Click Activity > Action Runs and select either All Runs or Integration runs from the filter to note that an endpoint validation action is initiated and running.

7 When prompted to trust the self-signed certificate from the IPAM provider, click Accept.

   After you accept the self-signed certificate, the validation action can continue to completion.

8 Enter a Name for this IPAM integration point, such as Infoblox_Integration, and a Description, such as Infoblox IPAM with ABX integration for team HRG.

9 Click Add to save the new external IPAM integration point.

   A data collection action is imitated. Networks and IP ranges are data-collected from the IPAM provider. You can view the data collection action as follows:

   a Click the Extensibility tab.

   b Click Activity > Action Runs and note that a data collection action is initiated and running.

   You can open and view the action run content.

Results

The provider-specific external IPAM integration is now available for use with networks and network profiles.

Configure a network and network profile to use external IPAM for an existing network in vRealize Automation Cloud

You can define an existing network to use IP address values that are obtained from, and managed by, an external IPAM provider rather than internally from vRealize Automation Cloud.

You can define a network to access existing IP settings that you have defined in your organization’s external IPAM provider account. This step expands on the Infoblox provider integration that you created in the previous step.
In this example, you configure a network profile with existing networks that were data-collected from vCenter. You then configure these networks to obtain IP information from an external IPAM provider, in this case Infoblox. Virtual machines that you provision from vRealize Automation Cloud that can be matched with this network profile obtain their IP and other TCP/IP related settings from the external IPAM provider.

For more information about networks, see Network resources in vRealize Automation Cloud. For more information about network profiles, see How to add network profiles in vRealize Automation Cloud and Learn more about network profiles in vRealize Automation Cloud.

For related information, see How do I configure a network profile to support an on-demand network for an external IPAM integration in vRealize Automation Cloud.

Prerequisites

This sequence of steps is shown in the context of an IPAM provider integration workflow. See Tutorial: Configuring a provider-specific external IPAM integration for vRealize Automation Cloud.

- Verify that you have cloud administrator credentials. See Credentials required for working with cloud accounts in vRealize Automation Cloud.
- Verify that you have the cloud administrator user role. See What are the vRealize Automation Cloud user roles.
- Verify that you have an account with the external IPAM provider, for example Infoblox or Bluecat, and that you have the correct access credentials to your organization's account with the IPAM provider. In this example workflow, the IPAM provider is Infoblox.
- Verify that you have an IPAM integration point for the IPAM provider. See Add an external IPAM integration for Infoblox in vRealize Automation Cloud.

Procedure

1. To configure a network, click Infrastructure > Resources > Networks.

2. On the Networks tab, select an existing network to use with the IPAM provider integration point. In this example, the network name is net.23.117-only-IPAM.

Listed networks have been data-collected by vRealize Automation Cloud from a vCenter in your organization.

3. To obtain values from the external IPAM provider, verify that except for the Account/region, Name, and Network domain, all other network settings are empty, including the following:
   - Domain (See Note in step 8)
   - CIDR
   - Default gateway
   - DNS servers
   - DNS search domains

4. Click the IP Ranges tab and click Add IPAM IP Range.
5 From the **Network** menu, select the network that you just configured, for example `net.23.117-only-IPAM`.

6 From the **Provider** menu, select the **Infoblox_Integration** IPAM integration point that you created earlier in the workflow.

7 From the now-visible **Address Space** drop-down menu, select one of the listed network views.

An address space in Infoblox is referred to as a network view.

The network views are obtained from your IPAM provider account. This example uses the network subnet that you just configured, for example `net.23.117-only-IPAM`, the **Infoblox_Integration** integration point that you created earlier in the workflow, and an address space named `default`.

Listed address space values are obtained from the external IPAM provider.

8 From the list of displayed networks that are available for the selected address space, select one or more networks, for example select `10.23.117.0/24`.

For this example, the **Domains** and **DNS Servers** column values for the selected network contain values from Infoblox.

---

**Note** If you select a network in Step 3 that had a Domain specified for vRealize Automation Cloud, and then select a network from the external IPAM provider address space that contains a Domain value, the Domain value in the external IPAM provider network takes precedence over the Domain specified in vRealize Automation Cloud. If the IPAM IP range setting doesn’t have a Domain value, specified in either Cloud Assembly or in the external IPAM provider as described above, provisioning fails.

For Infoblox, you can use the blueprint property `Infoblox.IPAM.Network.dnsSuffix` at the machine level to overwrite the Domain value. For related information, see [Using Infoblox-specific properties and extensible attributes for IPAM integrations in vRealize Automation Cloud cloud templates](#).

9 Click **Add** to save the IPAM IP range for the network.

The range is visible in the **IP Ranges** table.

10 Click the **IP Addresses** tab.

After you provision a machine by using the new address range from the external IPAM provider, a new record will be visible in the **IP Addresses** table.

11 To configure a network profile to use the network, click **Infrastructure > Configure > Network Profiles**.

12 Name the network profile, for example `Infoblox-NP`, and add the following sample settings.

   - **Summary tab**
     - Specify a vSphere cloud account/region.
Add a capability tag for the network profile, for example named `infoblox_abx`.

Make note of the capability tag, as you must also use it as a cloud template constraint tag to make the provisioning association in the cloud template.

Networks tab

Add the network that you created earlier, for example `net.23.117-only-IPAM`.

13 Click **Save** to save the network profile with these settings.

**Results**

The network and network profile setting are now configured for an existing network type to be used for the Infoblox IPAM integration in a cloud template design.

**Define and deploy a cloud template that uses an external IPAM provider range assignment in vRealize Automation Cloud**

You can define a cloud template to obtain and manage IP address assignments from your external IPAM provider. This example uses Infoblox as the external IPAM provider.

In this final step in the external IPAM integration workflow, you define and deploy a cloud template that connects your previously defined network and network profile to your organization's Infoblox account to obtain and manage IP address assignments for deployed VMs from the external IPAM provider rather than from Cloud Assembly.

This workflow uses Infoblox as the external IPAM provider and in some steps, the example values are unique to Infoblox, although the intent is that the procedure can be applied to other external IPAM integrations.

The Automate IPAM and DNS for VMs using VMware vRealize Automation and Infoblox DDI Infoblox blog provides related information.

After you deploy the cloud template and the VM is started, the IP address used for each VM in the deployment appears as a network entry in the **Resources > Networks** page, as a new host record in the IPAM provider network in your IPAM provider’s account, and in the vSphere Web Client record for each deployed VM in the host vCenter.

**Prerequisites**

This sequence of steps is shown in the context of an external IPAM provider integration workflow. See Tutorial: Configuring a provider-specific external IPAM integration for vRealize Automation Cloud.

- Verify that you have cloud administrator credentials. See Credentials required for working with cloud accounts in vRealize Automation Cloud.

- Verify that you have the cloud administrator user role. See What are the vRealize Automation Cloud user roles.
Verify that you have an account with the external IPAM provider, for example Infoblox or BlueCat, and that you have the correct access credentials to your organization's account with the IPAM provider.

Verify that you have administrator access to the host account and any role requirements needed to display status records in the vSphere web client record for your deployed VMs in the host vCenter.

Verify that you have an IPAM integration point for the external IPAM provider. See Add an external IPAM integration for Infoblox in vRealize Automation Cloud.

Verify that you have configured a Cloud Assembly network and network profile that support external IPAM integration for your intended IPAM integration point. See Configure a network and network profile to use external IPAM for an existing network in vRealize Automation Cloud.

Verify that your project and cloud zone are tagged to match tags in the IPAM integration point and network or network profile. Optionally configure the project to support custom resource naming.

For more information than provided about the role of a project and cloud zone, as well as the role of other infrastructure elements in your cloud template, see Tutorial: Setting up and testing multi-cloud infrastructure and deployments in Cloud Assembly. For more information about tagging, see How do I use tags to manage Cloud Assembly resources and deployments.

For information about custom naming VMs by using settings in your project, see Custom naming for deployed resources in Cloud Assembly.

Procedure

1. Click Cloud templates > New, enter the following information in the New cloud template page, and click Create.
   - **Name** = ipam-bpa
   - **Description** = Cloud template that uses Infoblox IPAM integration
   - **Project** = 123VC

2. For this example, add a cloud agnostic machine component and a cloud agnostic network component to the cloud template canvas and connect the two components.

3. Edit the cloud template code to add a constraint tag to the network component that matches the capability tag that you added to the network profile. For this example, that tag value is infoblox_abx.

4. Edit the cloud template code to specify that the network assignment type is static.

   When using an external IPAM provider, the assignment: static setting is required.
For this example, the specified IP address 10.23.117.4 is known to be currently available in the external IPAM address space that we selected for the network in the associated network profile. While the assignment: static setting is required, the address: value setting is not. You can choose to begin external IP address selection at a particular address value, but doing so is not required. If you do not specify an address: value setting, the external IPAM provider selects the next available address in the external IPAM network.

5 Verify the cloud template code against the following example.

```json
formatVersion: 1
inputs: {}
resources:
  Cloud_Network_1:
    type: Cloud.Network
    properties:
      networkType: existing
      name: ipam
  Cloud_Machine_1:
    type: Cloud.Machine
    properties:
      image: ubuntu
      flavor: small
      networks:
        - network: '${resource.Cloud_Network_1.id}'
          assignment: static
          address: 10.23.117.4
          name: '${resource.Cloud_Network_1.name}'
```

For examples of Infoblox properties that are available for specifying DNS and DHCP settings in cloud templates, see Using Infoblox-specific properties and extensible attributes for IPAM integrations in vRealize Automation Cloud cloud templates.

6 Click Deploy on the cloud template page, name the deployment Infoblox-1, and click Deploy on the Deployment Type page.

7 As the cloud template is being deployed, click the Extensibility tab and select Activity > Action Runs to see the Infoblox_AllocateIP_n extensibility action running.

After the extensibility action is completed and the machine is provisioned, the Infoblox_Update_n action propagates the MAC address to Infoblox.

8 You can log in to and open your Infoblox account to see the new host record for the IPAM address in the associated 10.23.117.0/24 network. You can also open the DNS tab in Infoblox to see the new DNS host record.

9 To verify that the VM is being provisioned, log in to your host vCenter and vSphere Web Client to locate the provisioned machine and view the DNS name and IP address.

After the provisioned VM is started, the MAC address is propagated to Infoblox by an Infoblox_AllocateIP extensibility action.
To view the new network record in Cloud Assembly, select **Infrastructure > Resources > Networks** and click to open the **IP Addresses** tab.

If you delete the deployment, the IPAM address of VMs in the deployment are released and the IP addresses are again available to the external IPAM provider for other allocations. The extensibility action for this event in Cloud Assembly is **Infoblox_Deallocate**.

**Using Infoblox-specific properties and extensible attributes for IPAM integrations in vRealize Automation Cloud cloud templates**

You can use Infoblox-specific properties for vRealize Automation Cloud projects that contain external IPAM integrations for Infoblox.

The following Infoblox properties are available for use with your Infoblox IPAM integrations in cloud template designs and deployments. You can use them in vRealize Automation Cloud to further control IP address allocation during cloud template deployment. Use of these properties is optional.

These properties are available and included in the most recent version Infoblox plug-in for vRealize Automation Cloud. For information about Infoblox plug-in versions and where to obtain the most recent version of the Infoblox plug-in for your IPAM integration in vRealize Automation Cloud, see **Download and deploy an external IPAM provider package for use in vRealize Automation Cloud**.

- **Infoblox.IPAM.createFixedAddress**
  
  This property enables you to create a fixed address record inside Infoblox. Possible values are True and False. By default, a host record is created. The default value is False.

- **Infoblox.IPAM.Network.dnsView**
  
  This property enables you to use a DNS view when creating a host record inside Infoblox.

- **Infoblox.IPAM.Network.enableDns**
  
  When allocating an IP in Infoblox, this property enables you to also create a DNS record. Possible values are True and False. The default value is True.

- **Infoblox.IPAM.Network.enableDhcp**
  
  You can set this option to True to enable the DHCP configuration for the host address.

- **Infoblox.IPAM.Network.dnsSuffix**
  
  This property enables you to overwrite the *domain* DHCP option of an Infoblox network with a new one. This capability is useful if the Infoblox network does not have the *domain* DHCP option set or if the the *domain* DHCP option must be overwritten. The default value is null (empty string).
When using an external IPAM provider such as Infoblox, you must specify a DNS suffix when provisioning a machine. While the DNS suffix is required, you can specify it in either of the following ways:

- Specify the DNS suffix on the vSphere network subnet in vRealize Automation Cloud.
- Specify the `Infoblox.IPAM.Network.dnsSuffix` property in the machine resource code in the cloud template.

An example is shown below in the `Infoblox.IPAM.Network.hostnameNicSuffix` section.

`Infoblox.IPAM.Network.dnsSuffix` is only applicable if `Infoblox.IPAM.Network.enableDns` is set to True.

`Infoblox.IPAM.Network.hostnameNicSuffix`

You can use this property to specify a NIC index suffix when generating a host name.

This allows you to provision a machine with more than one NIC such that the host names for each NIC are distinguished by a custom-defined suffix. As seen in the following example, you can provision a machine, for example `my-machine`, that has 2 NICs so that the host name suffix for the first NIC is `-nic1` and the other is `-nic2`.

You can also specify a DNS suffix as shown in the example. The `Infoblox.IPAM.Network.dnsSuffix` property is used with a value of `test.local` to result in the first NIC being named `my-machine-nic1.test.local` and the other `my-machine-nic2.test.local`.

```yaml
formatVersion: 1
inputs: {}
resources:
  Cloud_Machine_1:
    type: Cloud.Machine
    properties:
      Infoblox.IPAM.Network.dnsSuffix: test.local
      Infoblox.IPAM.Network0.hostnameNicSuffix: -nic1
      Infoblox.IPAM.Network1.hostnameNicSuffix: -nic2
    image: ubuntu
    flavor: small
    networks:
      - network: '${resource.Cloud_Network_1.id}'
        deviceIndex: 0
      - network: '${resource.Cloud_Network_2.id}'
        deviceIndex: 1
  Cloud_Network_1:
    type: Cloud.Network
    properties:
      networkType: existing
  Cloud_Network_2:
    type: Cloud.Network
    properties:
      networkType: existing
```
This property was introduced with Infoblox plug-in version 1.3. See Download and deploy an external IPAM provider package for use in vRealize Automation Cloud.

- You can also specify properties by using an extensibility subscription.

  For related information about Infoblox extensible attributes relative to this use case, see Add required extensible attributes in the Infoblox application for integration with vRealize Automation Cloud.

Using Infoblox properties on different machine NICs in a cloud template

The following Infoblox properties can support a different value for each machine NIC in the cloud template:

- Infoblox.IPAM.Network.enableDhcp
- Infoblox.IPAM.Network.dnsView
- Infoblox.IPAM.Network.enableDns
- Infoblox.IPAM.Network.hostnameNicSuffix

For example, to use a different Infoblox.IPAM.Network.dnsView value for each NIC, use a Infoblox.IPAM.Network<nicIndex>.dnsView entry for each NIC. The following sample shows different values Infoblox.IPAM.Network.dnsView for two NICs.

```json
formatVersion: 1
inputs: {}
resources:
  Cloud_Machine_1:
    type: Cloud.Machine
    properties:
      Infoblox.IPAM.Network0.dnsView: default
      Infoblox.IPAM.Network1.dnsView: my-net
    image: ubuntu
    flavor: small
    networks:
      - network: '${resource.Cloud_Network_1.id}'
        deviceIndex: 0
      - network: '${resource.Cloud_Network_2.id}'
        deviceIndex: 1
  Cloud_Network_1:
    type: Cloud.Network
    properties:
      networkType: existing
  Cloud_Network_2:
    type: Cloud.Network
    properties:
      networkType: existing
```
By default, the Infoblox integration creates a DNS host record in the default DNS view in Infoblox. If your Infoblox administrator has created custom DNS views, you can overwrite the default integration behavior and specify a named view by using the Infoblox.IPAM.Network.dnsView property in the machine component. For example, you can add the following property to the Cloud_Machine_1 component to specify a named DNS view in Infoblox.

```
Cloud_Machine_1:
  type: Cloud.Machine
  properties:
    image: ubuntu
    flavor: small
    Infoblox.IPAM.Network.dnsView:<dns-view-name>
```

For information about configuring and using DNS views, see DNS Views in Infoblox product documentation. For examples in the Infoblox integration workflow, see Define and deploy a cloud template that uses an external IPAM provider range assignment in vRealize Automation Cloud.

**How to specify Infoblox properties**

You can specify an Infoblox property using one of the following methods in Cloud Assembly:

- You can specify properties in a project by using the Custom Properties section on your Infrastructure > Administration > Projects page. Using this method, the specified properties are applied to all machines that are provisioned in the scope of this project.

- You can specify properties on each machine component in a cloud template. Sample cloud template code illustrating use of the Infoblox.IPAM.Network.dnsView property is shown below:

```
formatVersion: 1
inputs: {}
resources:
  Cloud_vSphere_Machine_1:
    type: Cloud.vSphere.Machine
    properties:
      Infoblox.IPAM.Network.dnsView: default
      image: ubuntu
      cpuCount: 1
      totalMemoryMB: 1024
      networks:
        - network: '${resource.Cloud_Network_1.id}'
  Cloud_Network_1:
    type: Cloud.Network
    properties:
      networkType: existing
      constraints:
        - tag: mk-ipam-demo
```
Control network data collection by using Infoblox filters in vRealize Automation Cloud

For Infoblox, you can limit the number of data collected networks to only those networks that are needed for vRealize Automation Cloud operations. This reduces the amount of transferred data and enhances system performance.

vRealize Automation Cloud collects data every 10 minutes from the external IPAM system. For Infoblox, you can filter in several ways to discover and data-collect only a subset of networks that are used by vRealize Automation Cloud operations.

To filter data collection for networks that use Infoblox-generated IP addresses, use the following properties on the IPAM integration tab. The filter properties are available as you create or edit the external IPAM integration point for Infoblox.

These filters are only available with the Infoblox v1.3 plug-in.

- **Infoblox.IPAM.NetworkContainerFilter**
  Filters on network containers.

- **Infoblox.IPAM.NetworkFilter**
  Filter on networks.

- **Infoblox.IPAM.RangeFilter**
  Filter on IP address ranges.

These options are available for use with the vRA Cloud Infoblox plugin version 1.3.

Be cautious when applying these data collection filters to networks that have already been data-collected. If you apply filters to prevent some networks from being data-collected, the networks that are not collected are assumed to be unnecessary and are deleted from vRealize Automation Cloud. The exception are networks that are associated to vRealize Automation Cloud subnets. Previously data-collected networks that are not subsequently discovered and data-collected, for example because they were filtered out of the data collection task, are deleted from the vRealize Automation Cloud database. However, if the previously data-collected networks are in use in vRealize Automation Cloud, they are not deleted.

These filters are applied as query parameters in the search requests for the different network objects. You can use any search parameters that Infoblox supports. You filter by CIDR or extensible attributes that are based on regular expressions or exact matches. The format uses the Infoblox WAPI filtration format, as described in Infoblox WAPI documentation. Methods of filtering by CIDR or extensible attributes are shown in the following examples:

- **Filter based on CIDR for networks and network containers. Examples:**
  - **Exact match** - Infoblox.IPAM.NetworkFilter: network=192.168.0.0
  - **Match by extensible attribute** - Infoblox.IPAM.NetworkFilter: network~=192.168

- **Filter based on CIDR for IP address range. Example:**
Match by regular expression and network view name - Infoblox.IPAM.RangeFilter:

- network=192.168.0.0/16 & network_view=my_view

- Filter based on extensible attributes for networks, IP ranges, and network containers.
  Syntax uses the `filter_name=*ext_attr=ext_attr_value` format. Examples:

  - **Exact match** - *Building=Data Center
  - **Match by regular expression with `~`** - *Building=*Center
  - **Case sensitive match with `:`** - *Building:=data center
  - **Exclude match with `!`** - *Building!=Data Center
  - **Match by regular expression (case sensitive and exclude can be combined):** *Building!=~:=Data Center / *Building:~:=center

- Filter based on CIDR and extensible attributes using syntax from the above methods of filtering. Example:

  - network=192.168.0.0/16 & *Building=Data Center

For more information about using extensible attributes and regular expressions in these properties, see [Infoblox Supported Expressions for Search Parameters](https://infoblox.com/doc/infoblox-supported-expressions) and [Infoblox REST API Reference Guide](https://infoblox.com/doc/infoblox-rest-api).
Setting up Cloud Assembly for your organization

As a Cloud Assembly administrator, you must understand the user roles and set up connections with your cloud account vendor and integration applications. When you configure the cloud accounts and integrations, you are configuring the communication between Cloud Assembly and those target systems.

This chapter includes the following topics:

- What are the vRealize Automation Cloud user roles
- Adding cloud accounts to Cloud Assembly
- Integrating vRealize Automation Cloud with other applications
- What are onboarding plans in Cloud Assembly
- Advanced configuration for Cloud Assembly environment

What are the vRealize Automation Cloud user roles

vRealize Automation Cloud has several levels of user roles. These different levels control access to the organization, the services, the projects that produce or consume the cloud templates, catalog items, and pipelines, and the ability for users to use or see individual parts of the user interface. These different levels give cloud administrators different tools to apply any level of granularity that is required by their operational needs.

General role descriptions

The user roles are defined at different levels. The service level roles are defined for each service. More details for the service roles is provided below this table.
<table>
<thead>
<tr>
<th>Role</th>
<th>General permissions</th>
<th>Where the role is defined</th>
</tr>
</thead>
<tbody>
<tr>
<td>Organization Owner</td>
<td>Can access the console and add users to organization. The organization owner cannot access a service unless they have a service role. More about the Organization User Roles</td>
<td>Organization console</td>
</tr>
<tr>
<td>Organization Member</td>
<td>Can access the console. The organization member cannot access a service unless they have a service role. More about the Organization User Roles</td>
<td>Organization console</td>
</tr>
<tr>
<td>Service Administrator</td>
<td>Can access the console and has full view, update, and delete privileges in the service. Cloud Assembly Service Roles, Service Broker Service Roles, Code Stream Service Roles</td>
<td>Organization console</td>
</tr>
<tr>
<td>Service User</td>
<td>Can access the console and the service with limited permissions. The service member has limited user interface. What they can see or do depends on their project membership. Cloud Assembly Service Roles, Service Broker Service Roles, Code Stream Service Roles</td>
<td>Organization console</td>
</tr>
<tr>
<td>Service Viewer</td>
<td>Can access the console and the service in a view-only mode. Cloud Assembly Service Roles, Service Broker Service Roles, Code Stream Service Roles</td>
<td>Organization console</td>
</tr>
<tr>
<td>Executor (Code Stream only)</td>
<td>Can access the console and manage pipeline executions. Code Stream Service Roles</td>
<td>Organization console</td>
</tr>
<tr>
<td>Role</td>
<td>General permissions</td>
<td>Where the role is defined</td>
</tr>
<tr>
<td>--------------------</td>
<td>-------------------------------------------------------------------------------------</td>
<td>----------------------------------------------------------------</td>
</tr>
<tr>
<td>Project roles</td>
<td>Can view and manage project resources depending on project role.</td>
<td>Cloud Assembly, Service Broker, and Code Stream</td>
</tr>
<tr>
<td></td>
<td>Project roles include administrator, member, and viewer.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Organization and service user roles in vRealize Automation Cloud</td>
<td></td>
</tr>
<tr>
<td>Custom roles</td>
<td>The permissions are defined by the Cloud Assembly for all the services.</td>
<td>Cloud Assembly and Service Broker</td>
</tr>
<tr>
<td></td>
<td>The user must have at least a service viewer role in the relevant services so that</td>
<td></td>
</tr>
<tr>
<td></td>
<td>they can access the service. The custom roles take precedence over the service roles.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Custom user roles in vRealize Automation Cloud</td>
<td></td>
</tr>
</tbody>
</table>

### Organization and service user roles in vRealize Automation Cloud

The organization and service user roles that you defined for the Cloud Assembly, Service Broker, and Code Stream services determine what the user can see and do in each service.

#### Organization User Roles

User roles are defined for the organization in the vRealize Automation Cloud console by an organization owner. There are two types of roles, organization roles and service roles.

The organization roles are global and apply to all services in the organization. The organization-level roles are Organization owner or Organization Member role.

The Cloud Assembly service roles, which are service-specific permissions, are also assigned at the organization level in the console.

#### Service Roles

These service roles are assigned by the organization owner.

This article includes information about the following services.

- **Cloud Assembly Service Roles**
- **Service Broker Service Roles**
- **Code Stream Service Roles**

### Cloud Assembly Service Roles

The Cloud Assembly service roles determine what you can see and do in Cloud Assembly. These service roles are defined in the console by an organization owner.
Table 3-1. Cloud Assembly Service Role Descriptions

<table>
<thead>
<tr>
<th>Role</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cloud Assembly Administrator</td>
<td>A user who has read and write access to the entire user interface and API resources. This is the only user role that can see and do everything, including add cloud accounts, create new projects, and assign a project administrator.</td>
</tr>
<tr>
<td>Cloud Assembly User</td>
<td>A user who does not have the Cloud Assembly Administrator role. In a Cloud Assembly project, the administrator adds users to projects as project members, administrators, or viewers. The administrator can also add a project administrator.</td>
</tr>
<tr>
<td>Cloud Assembly Viewer</td>
<td>A user who has read access to see information but cannot create, update, or delete values. This is a read-only role across all projects. Users with the viewer role can see all the information that is available to the administrator. They cannot take any action unless you make them a project administrator or a project member. If the user is affiliated with a project, they have the permissions related to the role. The project viewer would not extend their permissions the way that the administrator or member role does.</td>
</tr>
</tbody>
</table>

In addition to the service roles, Cloud Assembly has project roles. Any project is available in all of the services.

The project roles are defined in Cloud Assembly and can vary between projects.

In the following tables, which tells you what the different service and project roles can see and do, remember that the service administrators have full permission on all areas of the user interface.

The descriptions of project roles will help you decide what permissions to give your users.

- Project administrators leverage the infrastructure that is created by the service administrator to ensure that their project members have the resources they need for their development work.
- Project members work within their projects to design and deploy cloud templates.
- Project viewers are restricted to read-only access, except in a few cases where they can do non-destructive things like download cloud templates.
- Project supervisors are approvers in Service Broker for their projects where an approval policy is defined with a project supervisor approver. To provide the supervisor with context for approvals, consider also granting them the project member or viewer role.
Table 3-2. Cloud Assembly service roles and project roles

<table>
<thead>
<tr>
<th>UI Context</th>
<th>Task</th>
<th>Cloud Assembly Administrator</th>
<th>Cloud Assembly Viewer</th>
<th>Cloud Assembly User</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Project Administrator</td>
<td>Project Member</td>
<td>Project Viewer</td>
</tr>
<tr>
<td>Access Cloud Assembly</td>
<td></td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Console</td>
<td>In the vRA console, you can see and open Cloud Assembly</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Infrastructure</td>
<td>See and open the Infrastructure tab</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Configure - Projects</td>
<td>Create projects</td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Update, or delete values from project summary, provisioning, Kubernetes, integrations, and test project configurations</td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Add users and groups, and assign roles in projects</td>
<td>Yes</td>
<td></td>
<td>Yes. Your projects</td>
</tr>
<tr>
<td>View projects</td>
<td></td>
<td>Yes</td>
<td>Yes</td>
<td>Yes. Your projects</td>
</tr>
<tr>
<td>Configure - Cloud Zones</td>
<td>Create, update, or delete cloud zones</td>
<td>Yes</td>
<td></td>
<td>Yes. Your projects</td>
</tr>
<tr>
<td></td>
<td>View cloud zones</td>
<td>Yes</td>
<td></td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>View cloud zone Insights dashboard</td>
<td>Yes</td>
<td></td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>View cloud zones alerts</td>
<td>Yes</td>
<td></td>
<td>Yes</td>
</tr>
</tbody>
</table>
Table 3-2. Cloud Assembly service roles and project roles (continued)

<table>
<thead>
<tr>
<th>UI Context</th>
<th>Task</th>
<th>Cloud Assembly Administrator</th>
<th>Cloud Assembly Viewer</th>
<th>Cloud Assembly User</th>
</tr>
</thead>
<tbody>
<tr>
<td>Configure - Kubernetes Zones</td>
<td>Create, update, or delete Kubernetes zones</td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>View Kubernetes zones</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Configure - Flavors</td>
<td>Create, update, or delete flavors</td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>View flavors</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Configure - Image Mappings</td>
<td>Create, update, or delete image mappings</td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>View image mappings</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Configure - Network Profiles</td>
<td>Create, update, or delete network profiles</td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>View image network profiles</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Configure - Storage Profiles</td>
<td>Create, update, or delete storage profiles</td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>View image storage profiles</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Configure - Pricing Cards</td>
<td>Create, update, or delete pricing cards</td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>View the pricing cards</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Configure - Tags</td>
<td>Create, update, or delete tags</td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>View tags</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Resources - Compute</td>
<td>Add tags to discovered compute resources</td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Table 3-2. Cloud Assembly service roles and project roles (continued)

<table>
<thead>
<tr>
<th>UI Context</th>
<th>Task</th>
<th>Cloud Assembly Administrator</th>
<th>Cloud Assembly Viewer</th>
<th>Cloud Assembly User User must be a project administrator or member to see and do project-related tasks.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Project Administrator</td>
</tr>
<tr>
<td>Resources - Networks</td>
<td>Modify network tags, IP ranges, IP addresses</td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>View discovered network resources</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Resources - Security</td>
<td>Add tags to discovered security groups</td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>View discovered security groups</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Resources - Storage</td>
<td>Add tags to discovered storage</td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>View storage</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Resources - Machines</td>
<td>Add and delete machines</td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>View machines</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes. Your projects</td>
</tr>
<tr>
<td>Resources - Volumes</td>
<td>Delete discovered storage volumes</td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>View discovered storage volumes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes. Your projects</td>
</tr>
<tr>
<td>Resources - Kubernetes</td>
<td>Deploy or add Kubernetes clusters, and create or add namespaces</td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Table 3-2. Cloud Assembly service roles and project roles (continued)

<table>
<thead>
<tr>
<th>UI Context</th>
<th>Task</th>
<th>Cloud Assembly Administrator</th>
<th>Cloud Assembly Viewer</th>
<th>Cloud Assembly User</th>
</tr>
</thead>
<tbody>
<tr>
<td>UI Context</td>
<td></td>
<td>User must be a project administrator or member to see and do project-related tasks.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Activity - Requests</td>
<td>Delete deployment request records</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes. Your projects</td>
</tr>
<tr>
<td>Activity - Event Logs</td>
<td>View event logs</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes. Your projects</td>
</tr>
<tr>
<td>Connections - Cloud Accounts</td>
<td>Create, update, or delete cloud accounts</td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Connections - Integrations</td>
<td>Create, update, or delete integrations</td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Connections - Cloud Proxies</td>
<td>Create, update, or delete cloud proxies</td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Onboarding</td>
<td>Create, update, or delete onboarding plans</td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Marketplace</td>
<td>See and open the Marketplace tab</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
</tbody>
</table>
Table 3-2. Cloud Assembly service roles and project roles (continued)

<table>
<thead>
<tr>
<th>UI Context</th>
<th>Task</th>
<th>Cloud Assembly Administrator</th>
<th>Cloud Assembly Viewer</th>
<th>Cloud Assembly User</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>User must be a project administrator or member to see and do project-related tasks.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Project Administrator</td>
<td>Project Member</td>
<td>Project Viewer</td>
</tr>
<tr>
<td>Marketplace - Cloud Templates</td>
<td>Use the downloaded cloud templates on the Design tab</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Marketplace - Images</td>
<td>Download a cloud template</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Marketplace - Downloads</td>
<td>View the cloud templates</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>Download images</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>View images</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Marketplace - Downloads</td>
<td>View the log of all downloaded items</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Extensibility</td>
<td>See and open the Extensibility tab</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Events</td>
<td>View extensibility events</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Subscriptions</td>
<td>Create, update, or delete extensibility subscriptions</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Subscriptions</td>
<td>Deactivate subscriptions</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Library - Event topics</td>
<td>View event topics</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Library - Actions</td>
<td>Create, update, or delete extensibility actions</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>
Table 3-2. Cloud Assembly service roles and project roles (continued)

<table>
<thead>
<tr>
<th>UI Context</th>
<th>Task</th>
<th>Cloud Assembly Administrator</th>
<th>Cloud Assembly Viewer</th>
<th>Cloud Assembly User</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Project Administrator</td>
<td>Project Member</td>
<td>Project Viewer</td>
</tr>
<tr>
<td>View extensibility actions</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Library - Workflows</td>
<td>View extensibility workflows</td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Activity - Action Runs</td>
<td>Cancel or delete extensibility action runs</td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Activity - Workflow Runs</td>
<td>View extensibility workflow runs</td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Design</td>
<td>Open the Design tab</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Cloud Templates</td>
<td>Create, update, and delete cloud templates</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>View cloud templates</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Download cloud templates</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Upload cloud templates</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Deploy cloud templates</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Version and restore cloud templates</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Release cloud templates to the catalog</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>
Table 3-2. Cloud Assembly service roles and project roles (continued)

<table>
<thead>
<tr>
<th>UI Context</th>
<th>Task</th>
<th>Cloud Assembly Administrator</th>
<th>Cloud Assembly Viewer</th>
<th>Cloud Assembly User</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>User must be a project administrator or member to see and do project-related tasks.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Custom Resources</td>
<td>Create, update or delete custom resources</td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>View custom resources</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes. Your projects</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Custom Actions</td>
<td>Create, update, or delete custom actions</td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>View custom actions</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes. Your projects</td>
</tr>
<tr>
<td>Deployments</td>
<td>See and open the Deployments tab</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>View deployments, including deployment details, deployment history, price, monitor, alerts, optimize, and troubleshooting information</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes. Your projects</td>
</tr>
<tr>
<td>Alerts</td>
<td>Manage alerts</td>
<td>Yes</td>
<td>Yes. Your project</td>
<td>Yes. your project</td>
</tr>
<tr>
<td></td>
<td>Run day 2 actions on deployments based on policies</td>
<td>Yes</td>
<td>Yes. Your projects</td>
<td>Yes. Your projects</td>
</tr>
<tr>
<td></td>
<td>See and open the Alerts tab</td>
<td>Yes</td>
<td>Yes. Your project</td>
<td>Yes</td>
</tr>
</tbody>
</table>

VMware, Inc. 144
Table 3-2. Cloud Assembly service roles and project roles (continued)

<table>
<thead>
<tr>
<th>UI Context</th>
<th>Task</th>
<th>Cloud Assembly Administrator</th>
<th>Cloud Assembly Viewer</th>
<th>Cloud Assembly User</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td><strong>User must be a project administrator or member to see and do project-related tasks.</strong></td>
</tr>
<tr>
<td>Manage alerts</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes. Your projects</td>
<td>Yes. Your projects</td>
</tr>
<tr>
<td>View alerts</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes. Your projects</td>
<td>Yes. Your projects</td>
</tr>
</tbody>
</table>

Service Broker Service Roles

The Service Broker service roles determine what you can see and do in Service Broker. These service roles are defined in the console by an organization owner.

Table 3-3. Service Broker Service Role Descriptions

<table>
<thead>
<tr>
<th>Role</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Service Broker Administrator</td>
<td>Must have read and write access to the entire user interface and API resources. This is the only user role that can perform all tasks, including creating a new project and assigning a project administrator.</td>
</tr>
<tr>
<td>Service Broker User</td>
<td>Any user who does not have the Service Broker Administrator role. In a Service Broker project, the administrator adds users to projects as project members, administrators, or viewers. The administrator can also add a project administrator.</td>
</tr>
<tr>
<td>Service Broker Viewer</td>
<td>A user who has read access to see information but cannot create, update, or delete values. Users with the viewer role can see all the information that is available to the administrator. They cannot take any action unless you make them a project administrator or a project member. If the user is affiliated with a project, they have the permissions related to the role. The project viewer would not extend their permissions the way that the administrator or member role does.</td>
</tr>
</tbody>
</table>

In addition to the service roles, Service Broker has project roles. Any project is available in all of the services.

The project roles are defined in Service Broker and can vary between projects.

In the following tables, which tells you what the different service and project roles can see and do, remember that the service administrators have full permission on all areas of the user interface.
Use the following descriptions of project roles will help you as you decide what permissions to give your users.

- Project administrators leverage the infrastructure that is created by the service administrator to ensure that their project members have the resources they need for their development work.
- Project members work within their projects to design and deploy cloud templates.
- Project viewers are restricted to read-only access.
- Project supervisors are approvers in Service Broker for their projects where an approval policy is defined with a project supervisor approver. To provide the supervisor with context for approvals, consider also granting them the project member or viewer role.

Table 3-4. Service Broker Service Roles and Project Roles

<table>
<thead>
<tr>
<th>UI Context</th>
<th>Task</th>
<th>Service Broker Administrator</th>
<th>Service Broker Viewer</th>
<th>Service Broker User</th>
</tr>
</thead>
<tbody>
<tr>
<td>Access</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Service</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Broker</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Console</td>
<td></td>
<td>In the console, you can see</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>and open Service Broker</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Infrastructure</td>
<td></td>
<td>See and open the Infrastructure tab</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Configure - Projects</td>
<td>Create projects</td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Update, or delete values from project summary, provisioning, Kubernetes, integrations, and test project configurations.</td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Add users and groups, and assign roles in projects.</td>
<td>Yes</td>
<td>Yes. Your projects.</td>
<td></td>
</tr>
<tr>
<td>View projects</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes. Your projects</td>
<td>Yes. Your projects</td>
</tr>
</tbody>
</table>
Table 3-4. Service Broker Service Roles and Project Roles (continued)

<table>
<thead>
<tr>
<th>UI Context</th>
<th>Task</th>
<th>Service Broker Administrator</th>
<th>Service Broker Viewer</th>
<th>Project Administrator</th>
<th>Project Member</th>
<th>Project Viewer</th>
<th>Project Supervisor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Configure - Cloud Zones</td>
<td>Create, update, or delete cloud zones</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>View cloud zones</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Configure - Kubernetes Zones</td>
<td>Create, update, or delete Kubernetes zones</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>View Kubernetes zones</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Connections - Cloud Accounts</td>
<td>Create, update, or delete cloud accounts</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>View cloud accounts</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Connections - Integrations</td>
<td>Create, update, or delete integrations</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>View integrations</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Connections - Cloud Proxies</td>
<td>Create, update, or delete cloud proxies</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>View cloud proxies</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Activity - Requests</td>
<td>Delete deployment request records</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>View deployment request records</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Activity - Event Logs</td>
<td>View event logs</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Content and Policies</strong></td>
<td>See and open the Content and Policies tab</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Table 3-4. Service Broker Service Roles and Project Roles (continued)

<table>
<thead>
<tr>
<th>UI Context</th>
<th>Task</th>
<th>Service Broker Administrator</th>
<th>Service Broker Viewer</th>
<th>Service Broker User</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Project Administrator</td>
<td>Project Member</td>
<td>Project Viewer</td>
</tr>
<tr>
<td>Content Sources</td>
<td>Create, update, or delete content sources</td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>View content sources</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Content Sharing</td>
<td>Add or remove shared content</td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>View shared content</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Content</td>
<td>Customize form and configure item</td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>View content</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Policies - Definitions</td>
<td>Create, update, or delete policy definitions</td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>View policy definitions</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Policies - Enforcement</td>
<td>View enforcement log</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Catalog</td>
<td>See and open the Catalog tab</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>View available catalog items</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes. Your projects</td>
</tr>
<tr>
<td></td>
<td>Request a catalog item</td>
<td>Yes</td>
<td></td>
<td>Yes. Your projects</td>
</tr>
<tr>
<td>Deployments</td>
<td>See and open the Deployments tab</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>
### Table 3-4. Service Broker Service Roles and Project Roles (continued)

<table>
<thead>
<tr>
<th>UI Context</th>
<th>Task</th>
<th>Service Broker Administrator</th>
<th>Service Broker Viewer</th>
<th>Service Broker User</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>View deployments, including deployment details, deployment history, price, monitor, alerts, optimize, and troubleshooting information</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes. Your projects</td>
</tr>
<tr>
<td></td>
<td>Manage alerts</td>
<td>Yes</td>
<td>Yes. Your projects</td>
<td>Yes. Your projects</td>
</tr>
<tr>
<td></td>
<td>Run day 2 actions on deployments based on policies</td>
<td>Yes</td>
<td>Yes. Your projects</td>
<td>Yes. Your projects</td>
</tr>
</tbody>
</table>

#### Approvals

<table>
<thead>
<tr>
<th></th>
<th>See and open the Approvals tab</th>
<th>Yes</th>
<th>Yes</th>
<th>Yes</th>
<th>Yes</th>
<th>Yes</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Respond to approval requests</td>
<td>Yes</td>
<td>Yes. Your projects and the policy approver is Project Administrator</td>
<td>Only if you are a named approver</td>
<td>Only if you are a named approver</td>
<td>Yes. Your projects and the policy approver is Project Supervisor</td>
</tr>
</tbody>
</table>

### Code Stream Service Roles

The Code Stream service roles determine what you can see and do in Code Stream. These roles are defined in the console by the organization owner. Any project is available in all of the services.
Table 3-5. Code Stream Service Role Descriptions

<table>
<thead>
<tr>
<th>Role</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code Stream Administrator</td>
<td>A user who has read and write access to the entire user interface and API resources. This is the only user role that can see and do everything, including create projects, integrate endpoints, add triggers, create pipelines and custom dashboards, mark endpoints and variables as restricted resources, run pipelines that use restricted resources, and request that pipelines be published in Service Broker.</td>
</tr>
<tr>
<td>Code Stream Developer</td>
<td>A user who can work with pipelines, but cannot work with restricted endpoints or variables. If a pipeline includes a restricted endpoint or variable, this user must obtain approval on the pipeline task that uses the restricted endpoint or variable.</td>
</tr>
<tr>
<td>Code Stream Executor</td>
<td>A user who can run pipelines and approve or reject user operation tasks. This user can resume, pause, and cancel pipeline executions, but cannot modify pipelines.</td>
</tr>
<tr>
<td>Code Stream User</td>
<td>A user who can access Code Stream, but does not have any other privileges in Code Stream.</td>
</tr>
<tr>
<td>Code Stream Viewer</td>
<td>A user who has read access to see pipelines, endpoints, pipeline executions, and dashboards, but cannot create, update, or delete them. A user who also has the Service viewer role can see all the information that is available to the administrator. They cannot take any action unless you make them a project administrator or a project member. If the user is affiliated with a project, they have the permissions related to the role. The project viewer would not extend their permissions the way that the administrator or member role does.</td>
</tr>
</tbody>
</table>

In addition to the service roles, Code Stream has project roles. Any project is available in all the services.

The project roles are defined in Code Stream and can vary between projects.

In the following tables, which tell you what the different service and project roles can see and do, remember that the service administrators have full permission on all areas of the user interface.

Use the following descriptions of project roles to help you decide what permissions to give your users.

- Project administrators leverage the infrastructure that is created by the service administrator to ensure that their project members have the resources they need for their development work. The project administrator can add members.
- Project members who have a service role can use services.
- Project viewers can see projects but cannot create, update, or delete them.

All actions except restricted means this role has permission to perform create, read, update, and delete actions on entities except for restricted variables and endpoints.
## Table 3-6. Code Stream service role capabilities

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Pipelines</td>
<td>View pipelines</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>Create pipelines</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Run pipelines</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Run pipelines that include restricted endpoints or variables</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Update pipelines</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Delete pipelines</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pipeline Executions</td>
<td>View pipeline executions</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Resume, pause, and cancel pipeline executions</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Resume pipelines that stop for approval on restricted resources</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Custom Integrations</td>
<td>Create custom integrations</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Read custom integrations</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Update custom integrations</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Endpoints</td>
<td>View executions</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Create executions</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Update executions</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Delete executions</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mark resources as restricted</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Table 3-6. Code Stream service role capabilities (continued)

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Mark an endpoint or variable as restricted</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Dashboards

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>View dashboards</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Create dashboards</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Update dashboards</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Delete dashboards</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Custom user roles in vRealize Automation Cloud

As a Cloud Assembly administrator, you can create custom roles that define what users can see and do in vRealize Automation Cloud. You can then assign users to those roles.

Custom User Role Permissions

Using Cloud Assembly, you can define more granular user roles and then assign users to those roles. The custom roles have two categories, view and manage.

- **View.** A user assigned to a role with this permission can see all the items for all projects in the selected sections of the user interface. This role is useful for users who need to see accounts, configurations, or assigned values.

- **Manage.** A user assigned to a role with this permission can see all the items and has full add, edit, and delete permissions for all projects in the selected sections of the user interface.

These permissions extend the privileges that are granted by the other roles and are not restricted by project membership. For example, you can expand a project administrator’s permissions to manage parts of the infrastructure or give a service viewer an ability to review and respond to approvals requests.

To define the user roles and assign users, open Cloud Assembly or Service Broker as a service administrator and select **Infrastructure > Administration > Custom Roles.** You cannot configure the custom roles in Code Stream, however the roles apply to all the services.

Table 3-7. Custom Roles

<table>
<thead>
<tr>
<th>User Interface</th>
<th>Permission</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Infrastructure</td>
<td>View Cloud Accounts.</td>
<td>View cloud accounts.</td>
</tr>
</tbody>
</table>
**Table 3-7. Custom Roles (continued)**

<table>
<thead>
<tr>
<th>User Interface</th>
<th>Permission</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Manage Cloud Accounts</td>
<td>Create, update, or delete cloud accounts.</td>
<td></td>
</tr>
<tr>
<td>View Image Mappings</td>
<td>View image mappings.</td>
<td></td>
</tr>
<tr>
<td>Manage Image Mappings</td>
<td>Create, update, or delete image mappings.</td>
<td></td>
</tr>
<tr>
<td>View Flavor Mappings</td>
<td>View flavor mappings.</td>
<td></td>
</tr>
<tr>
<td>Manage Flavor Mappings</td>
<td>Create, update, or delete flavor mappings.</td>
<td></td>
</tr>
<tr>
<td>View Cloud Zones</td>
<td>View cloud zones, Insights, and alerts.</td>
<td></td>
</tr>
<tr>
<td>Manage Cloud Zones</td>
<td>Create, update, or delete cloud zones. Mange alerts.</td>
<td></td>
</tr>
<tr>
<td>View Machines</td>
<td>View machines.</td>
<td></td>
</tr>
<tr>
<td>View Requests</td>
<td>View activity requests.</td>
<td></td>
</tr>
<tr>
<td>Manage Requests</td>
<td>Delete activity requests.</td>
<td></td>
</tr>
<tr>
<td>View Integrations</td>
<td>View integrations.</td>
<td></td>
</tr>
<tr>
<td>Manage Integrations</td>
<td>Create, update, or delete integrations.</td>
<td></td>
</tr>
<tr>
<td>View Projects</td>
<td>View projects.</td>
<td></td>
</tr>
<tr>
<td>Manage Projects</td>
<td>Create projects. Add users and assign roles in projects. Update, or delete values from project summary, users, provisioning, Kubernetes, integrations, and test project configurations.</td>
<td></td>
</tr>
<tr>
<td>View Onboarding Plans</td>
<td>View onboarding plans</td>
<td></td>
</tr>
<tr>
<td>Manage Onboarding Plans</td>
<td>Create, update, run, or delete onboarding plans</td>
<td></td>
</tr>
</tbody>
</table>

**Catalog**

<table>
<thead>
<tr>
<th>View Content</th>
</tr>
</thead>
</table>

| Manage Content | Add, update, delete content sources. Share content. Customize the content, including the catalog icons and request forms. |

**Policies**

| View Policies | View policy definitions. |
Table 3-7. Custom Roles (continued)

<table>
<thead>
<tr>
<th>User Interface</th>
<th>Permission</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Manage Policies</td>
<td>Create, update, or delete policy definitions.</td>
</tr>
<tr>
<td>Deployments</td>
<td>View Deployments</td>
<td>View all deployments, including deployment details, deployment history, alerts, and troubleshooting information.</td>
</tr>
<tr>
<td></td>
<td>Manage Deployments</td>
<td>View all deployments, respond to alerts, and run all day 2 actions that the day 2 policies allow an administrator to run on deployments and deployment components.</td>
</tr>
<tr>
<td>Cloud Templates</td>
<td>View Cloud Templates</td>
<td>View cloud templates.</td>
</tr>
<tr>
<td></td>
<td>Manage Cloud Templates</td>
<td>Create, update, test, delete, version, share cloud templates, and release/unrelease a cloud template version.</td>
</tr>
<tr>
<td></td>
<td>Edit Cloud Templates</td>
<td>Create, update, test, version, share cloud templates, and release/unrelease a cloud template version. The role does not have permission to delete cloud templates.</td>
</tr>
<tr>
<td></td>
<td>Deploy Cloud Templates</td>
<td>Test and deploy any cloud template in any project.</td>
</tr>
<tr>
<td></td>
<td>Deploy In-line Cloud Template Content</td>
<td>Deploy any cloud template in the projects that the assignees are associated with. The project roles can be administrator, member, or viewer.</td>
</tr>
<tr>
<td>XaaS</td>
<td>View Custom Resources</td>
<td>View custom resources.</td>
</tr>
<tr>
<td></td>
<td>Manage Custom Resources</td>
<td>Create, update or delete custom resources.</td>
</tr>
<tr>
<td></td>
<td>View Resource Actions</td>
<td>View custom actions.</td>
</tr>
<tr>
<td></td>
<td>Manage Resource Actions</td>
<td>Create, update, or delete custom actions</td>
</tr>
<tr>
<td>Extensibility</td>
<td>View Extensibility Resources</td>
<td>View events, subscriptions, event topics, actions, workflows, action runs, and workflow runs.</td>
</tr>
</tbody>
</table>
### Table 3-7. Custom Roles (continued)

<table>
<thead>
<tr>
<th>User Interface</th>
<th>Permission</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Manage Extensibility Resources</td>
<td>Create, update, delete, and deactivate extensibility subscriptions. Create, update, or delete extensibility actions. Cancel or delete extensibility action runs.</td>
</tr>
<tr>
<td>Pipeline</td>
<td>Manage Pipelines</td>
<td>Create, edit, and delete pipeline, endpoint, variable, and trigger configurations. Restricted models are excluded.</td>
</tr>
<tr>
<td></td>
<td>Manage Restricted Pipelines</td>
<td>Create, edit, and delete pipeline, endpoint, variable, and trigger configurations. Restricted models are included.</td>
</tr>
<tr>
<td></td>
<td>Manage Custom Integrations</td>
<td>Add, edit, and delete custom integrations.</td>
</tr>
<tr>
<td></td>
<td>Execute Pipelines</td>
<td>Run pipeline model executions and triggers, and pause, cancel, resume, or re-run the executions and triggers.</td>
</tr>
<tr>
<td></td>
<td>Execute Restricted Pipelines</td>
<td>Run pipeline model executions and triggers, and pause, cancel, resume, or re-run the executions and triggers. Resolve restricted endpoints and variables.</td>
</tr>
<tr>
<td></td>
<td>Manage Executions</td>
<td>Run pipeline model executions and triggers, and pause, cancel, resume, or re-run the executions and triggers. Resolve restricted endpoints and variables. Delete executions.</td>
</tr>
<tr>
<td>Approval</td>
<td>Manage Approvals</td>
<td>View the Approvals tab where you can approve or reject approval requests. Approver with this role will not receive an email notification about an approval request unless they are an approver in the policy.</td>
</tr>
</tbody>
</table>

### Use cases: How can user roles help me control access in vRealize Automation Cloud

As a cloud administrator, you want to control the tasks that your users can perform in vRealize Automation Cloud. Depending on your management goals and application development team
responsibilities, there are different ways that you can configure the user roles to support those goals.

The following Cloud Assembly and Service Broker examples are based on three use cases. These examples provide only enough instruction to illustrate the application of user roles.

The target audience for these use cases is the cloud administrator, who is also considered the cloud administrator, and the service administrators.

The use cases build on each other. If you are ready to go directly to use case 3, you might need to review use cases 1 and 2 to better understand why you configure the roles in the ways specified.

The purpose of the use cases is to demonstrate user roles, not to provide detailed information about configuring your infrastructure, managing projects, creating cloud templates, and working with deployments.

Before you begin, you must understand the levels of user roles that are configured by a cloud administrator in the vRealize Automation Cloud Console.

- Organization Roles
  
The organization roles control who can access the console.

  As an organization owner, you must ensure that all users of any of the services are assigned at least an organization member role.

<table>
<thead>
<tr>
<th>Role</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Organization Owner</td>
<td>An administrator can add users, change the role of users, and remove users from the organization. The owner manages which services users have access to.</td>
</tr>
<tr>
<td>Organization Member</td>
<td>A general user can log in to the organization console. To access the services, an organization owner must assign the users service roles.</td>
</tr>
</tbody>
</table>

- Service Roles
  
The service roles control who can access their assigned services.
As an organization owner, you must ensure that the users who need access to the services are assigned the appropriate role. You use the roles to control how much the user can do in each service.

Table 3-8. Cloud Assembly Service Role Descriptions

<table>
<thead>
<tr>
<th>Role</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cloud Assembly Administrator</td>
<td>A user who has read and write access to the entire user interface and API resources. This is the only user role that can see and do everything, including add cloud accounts, create new projects, and assign a project administrator.</td>
</tr>
<tr>
<td>Cloud Assembly User</td>
<td>A user who does not have the Cloud Assembly Administrator role. In a Cloud Assembly project, the administrator adds users to projects as project members, administrators, or viewers. The administrator can also add a project administrator.</td>
</tr>
<tr>
<td>Cloud Assembly Viewer</td>
<td>A user who has read access to see information but cannot create, update, or delete values. This is a read-only role across all projects. Users with the viewer role can see all the information that is available to the administrator. They cannot take any action unless you make them a project administrator or a project member. If the user is affiliated with a project, they have the permissions related to the role. The project viewer would not extend their permissions the way that the administrator or member role does.</td>
</tr>
</tbody>
</table>

Table 3-9. Service Broker Service Role Descriptions

<table>
<thead>
<tr>
<th>Role</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Service Broker Administrator</td>
<td>Must have read and write access to the entire user interface and API resources. This is the only user role that can perform all tasks, including creating a new project and assigning a project administrator.</td>
</tr>
<tr>
<td>Service Broker User</td>
<td>Any user who does not have the Service Broker Administrator role. In a Service Broker project, the administrator adds users to projects as project members, administrators, or viewers. The administrator can also add a project administrator.</td>
</tr>
<tr>
<td>Service Broker Viewer</td>
<td>A user who has read access to see information but cannot create, update, or delete values. Users with the viewer role can see all the information that is available to the administrator. They cannot take any action unless you make them a project administrator or a project member. If the user is...</td>
</tr>
</tbody>
</table>
Table 3-9. Service Broker Service Role Descriptions (continued)

<table>
<thead>
<tr>
<th>Role</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project membership roles</td>
<td>Affiliated with a project, they have the permissions related to the role. The project viewer would not extend their permissions the way that the administrator or member role does.</td>
</tr>
</tbody>
</table>

Table 3-10. Code Stream Service Role Descriptions

<table>
<thead>
<tr>
<th>Role</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code Stream Administrator</td>
<td>A user who has read and write access to the entire user interface and API resources. This is the only user role that can see and do everything, including create projects, integrate endpoints, add triggers, create pipelines and custom dashboards, mark endpoints and variables as restricted resources, run pipelines that use restricted resources, and request that pipelines be published in Service Broker.</td>
</tr>
<tr>
<td>Code Stream Developer</td>
<td>A user who can work with pipelines, but cannot work with restricted endpoints or variables. If a pipeline includes a restricted endpoint or variable, this user must obtain approval on the pipeline task that uses the restricted endpoint or variable.</td>
</tr>
<tr>
<td>Code Stream Executor</td>
<td>A user who can run pipelines and approve or reject user operation tasks. This user can resume, pause, and cancel pipeline executions, but cannot modify pipelines.</td>
</tr>
<tr>
<td>Code Stream User</td>
<td>A user who can access Code Stream, but does not have any other privileges in Code Stream.</td>
</tr>
<tr>
<td>Code Stream Viewer</td>
<td>A user who has read access to see pipelines, endpoints, pipeline executions, and dashboards, but cannot create, update, or delete them. A user who also has the Service viewer role can see all the information that is available to the administrator. They cannot take any action unless you make them a project administrator or a project member. If the user is affiliated with a project, they have the permissions related to the role. The project viewer would not extend their permissions the way that the administrator or member role does.</td>
</tr>
</tbody>
</table>

- Project membership roles

The project membership determines what infrastructure resources and cloud templates are available.
Project membership is defined in the service by a user with a service administrator role. The service administrator must ensure that the users who need access to one or more projects are assigned the appropriate project role in each project.

Table 3-11. Project Roles

<table>
<thead>
<tr>
<th>Role</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project Administrator</td>
<td>A project administrator can manage their own projects, create and deploy cloud templates associated with their projects, and manage project deployments for all project members.</td>
</tr>
<tr>
<td>Project Member</td>
<td>A project member can create and deploy cloud templates associated with their projects, manage their own deployments, and manage any shared deployments.</td>
</tr>
<tr>
<td>Project Viewer</td>
<td>A project viewer is a member of the project with read-only access to their project resources, cloud templates, and deployments.</td>
</tr>
</tbody>
</table>

- Custom roles

The custom roles are created by the Cloud Assembly to refine the member and viewer roles.

The procedures provided in these use cases are meant to highlight the user roles. They are not detailed or definitive procedures for setting up vRealize Automation Cloud.

As you configure roles, remember that users who are running API operations are subject to the roles that you assign here.

Prerequisites

- Verify that you have the Organization Owner role. You must see the Identity and Access Management tab with you log in to the console. If not, contact the organization owner.

- Verify that you have the service administrator role for the various services. If you are not certain about your role, contact the organization owner.

- Verify that your users are added to vRealize Automation Cloud.

You might need to add them.
For a more detailed task and role list for various roles, see Organization and service user roles in vRealize Automation.

Procedure

1 User role use case 1: Set up the vRealize Automation Cloud user roles to support a small application development team

As a vRealize Automation Cloud cloud administrator, you are responsible for managing the access and the budget for your infrastructure resources. You add yourself and two others as administrators. This small team can create the infrastructure and develop the cloud templates that match the business goals of the teams that consume the cloud templates. You and your small team of administrators then deploy the cloud templates for your non-administrator consumers. You don't allow non-administrators to access vRealize Automation Cloud.

2 User role use case 2: Set up vRealize Automation Cloud user roles to support larger development teams and the catalog

As a vRealize Automation Cloud organization owner, you are responsible for managing the access and the budget for your infrastructure resources. You have a team of cloud template developers who iteratively create and deploy templates for different projects until they are ready to deliver to their consumers. You then deliver the deployable resources to the consumers in a catalog.

3 User role use case 3: Set up vRealize Automation Cloud custom user roles to refine system roles

As a vRealize Automation Cloud organization owner or service administrator, you manage user access using the organization and service system roles. However, you also want to create custom roles to that selected users and perform tasks or see content that is outside of their system roles.

User role use case 1: Set up the vRealize Automation Cloud user roles to support a small application development team

As a vRealize Automation Cloud cloud administrator, you are responsible for managing the access and the budget for your infrastructure resources. You add yourself and two others as administrators. This small team can create the infrastructure and develop the cloud templates that match the business goals of the teams that consume the cloud templates. You and your small team of administrators then deploy the cloud templates for your non-administrator consumers. You don't allow non-administrators to access vRealize Automation Cloud.

In this use case, you are the organization owner and you have a small team where they all have the service administrator role.

The following procedure follows one user all the way through the process. You can do each step for multiple users.
Prerequisites

- Verify that you meet all the prerequisites stipulated in the use case introduction. See Use cases: How can user roles help me control access in vRealize Automation Cloud.

Procedure

1. Assign organization roles. Click **Identity and Access Management**.
   
a. Log in to the vRealize Automation Cloud console.
   
b. Click **Identity and Access Management**.
   
c. Select the user name and click **Edit Roles**.
   
d. In the **Assign Organization Roles** drop-down menu, select **Organization Member**.

The organization member role ensures that the user can access the console and any services that you add them to. They cannot manage organization users.

Leave the Edit Role page open for this user and continue to the next step.
2. Assign Cloud Assembly Administrator role to yourself and to the one or two other administrators in this scenario.

The service administrator role has full privileges to add, edit, and delete infrastructure, projects, cloud templates, and deployments. Defining an administrator role for one person and the user role for a different person is covered in Scenario 2. This example uses Sylvia.

a. Click **Add Service Access**.

b. Configure the user with the following value.

<table>
<thead>
<tr>
<th>Service</th>
<th>Role</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cloud Assembly</td>
<td>Cloud Assembly Administrator</td>
</tr>
</tbody>
</table>

Using and Managing VMware Cloud Assembly

3. Create a project in Cloud Assembly that you use to group resources and manage resource billing for different business groups.

a. In the console, click the **Services** tab, and then click **Cloud Assembly**.

b. Select **Infrastructure > Projects > New Project**.

   This user role use case is focused on providing examples of how you can implement user roles, not on creating the fully defined system.

   For information about configuring the infrastructure, see Chapter 4 Building your Cloud Assembly resource infrastructure. For more about projects, see Chapter 5 Adding and managing Cloud Assembly projects.

   c. Enter **WebAppTeam** as the project name.

   d. Click **Users**, and then click **Add Users**.
Enter email addresses for the individuals who can help you build and manage the infrastructure and cloud templates. For example, tony@mycompany.com,syliva@mycompany.com.

In the **Assign role** drop-down menu, select **Administrator**.

As Cloud Assembly administrators, these two users already have administrator access to the cloud accounts, infrastructure, and all projects. This step helps you understand the roles used in the later scenarios. In the later scenarios, you define project administrator and project member roles, which have different permissions.

Click the **Provisioning** tab and add one or more cloud zones. Another reminder. This use case is about user roles.

### 4 Develop a simple cloud template so that you can test the WebAppTeam project.

This cloud template section is abbreviated. The focus is users and user roles as defined by projects, not how to create a cloud template.

- **Select Cloud Templates > New.**
- **For the new cloud template name, enter WebApp.**
- **For Project, select WebAppTeam.**

  ![New Cloud Template](image)

- **Select **Share only with the project.**

  This setting ensures that the cloud template is only available to project members. When you are ready to provide the cloud templates to other teams, you can select Allow an administrator to share with any project in this organization. Sharing the cloud template with other projects means that you do not have to maintain duplicate instances of the same base templates. You can move cloud templates from development projects to production projects so that catalog consumers can deploy to production infrastructure resources.

- **Click Create.**
In the cloud template designer, drag the **Cloud Agnostic > Machine** component to the canvas.

For more about configuring cloud templates, see Chapter 6 Designing your Cloud Assembly deployments.

Click **Deploy**.

Continue iterating on the cloud template until you are ready to provide it to your consumers.

Click **Version** and release and version the cloud template.

Send the users the log in information using your most common method.

**Results**

In this use case, you made your two colleagues organization members. You then made Sylvia a Cloud Assembly administrator. You made Tony a WebApp project administrator. This user role configuration only works for small teams where you deliver deployed applications to your consumers rather than providing them with self-service access or a catalog.

**User role use case 2: Set up vRealize Automation Cloud user roles to support larger development teams and the catalog**

As a vRealize Automation Cloud organization owner, you are responsible for managing the access and the budget for your infrastructure resources. You have a team of cloud template developers who iteratively create and deploy templates for different projects until they are ready to deliver to their consumers. You then deliver the deployable resources to the consumers in a catalog.

This use case assumes that you understand that use case 1 is an administrator-only use case. You now want to expand your system to support more teams and larger goals.

- Let developers create and deploy their own application cloud templates during development. You add yourself as administrator, then add additional users with both the service user and the service viewer role. Next, you add the users as project members. The project members can develop and deploy their own cloud templates.

- Publish cloud templates to a catalog where you make them available for non-developers to deploy. Now you are assigning user roles for Service Broker. Service Broker provides a catalog for the cloud template consumers. You can also use it to create policies, including leases and entitlements, but that functionality is not part of this user role use case.

**Prerequisites**

- Review first use case. See User role use case 1: Set up the vRealize Automation Cloud user roles to support a small application development team.

- Identify the following users based on what permissions you want them to have:
  - cloud template developers who will be Cloud Assembly users and viewers
- A Service Broker administrator
- Non-developer users who will be catalog consumers as Service Broker users

**Procedure**

1. Assign organization member roles to your cloud template developer users.
   
   If you need instructions, see the User role use case 1: Set up the vRealize Automation Cloud user roles to support a small application development team.

2. Assign the Cloud Assembly service member role to your cloud template developers.
   a. Click **Add Service Access**.

   ![Add Service Access](image)

   b. Configure the user with the following value.

<table>
<thead>
<tr>
<th>Service</th>
<th>Role</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cloud Assembly</td>
<td>Cloud Assembly User</td>
</tr>
<tr>
<td>Cloud Assembly</td>
<td>Cloud Assembly Viewer</td>
</tr>
</tbody>
</table>

   In this use case, your developers need to see the infrastructure to ensure that they are building deployable cloud templates. As users that you will assign as project administrators and project members in the next step, they cannot see the infrastructure. As service viewers they can see how the infrastructure is configured, but cannot make any changes. As the cloud administrator, you remain in control, but give them access to the information they need to develop cloud templates.
Create projects in Cloud Assembly that you use to group resources users.

In this use case, you create two projects. The first project is PersonnelAppDev and the second is PayrollAppDev.

a In the console, click the Services tab, and then click Cloud Assembly.
b Select Infrastructure > Projects > New Project.
c Enter PersonnelAppDev as the name.
d Click Users, and then click Add Users.
e Add project members and assign a project administrator.

<table>
<thead>
<tr>
<th>Project Role</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project User</td>
<td>A project member is the primary developer user role in a project. Projects determine what cloud resources are available when you are ready to test your development work by deploying a cloud template.</td>
</tr>
<tr>
<td>Project Administrator</td>
<td>A project administrator supports their developers by adding and removing users for your projects. You can also delete your projects. To create a project, you must have service administrator privileges.</td>
</tr>
</tbody>
</table>

f For the users that you are adding as project members, enter the email address of each user, separated by a comma, and select User in the Assign role drop-down menu.

For example, tony@mycompany.com,sylvia@mycompany.com.

g For the designated administrators, select Administrator in the Assign role drop-down menu and provide the necessary email address.
h  Click the Provisioning tab and add one or more cloud zones.

When the cloud template developers who are part of this project deploy a template, it is deployed to the resources available in the cloud zones. You must ensure that the cloud zone resources match the needs of the project development team templates.

i  Repeat the process to add the PayrollAppDev project with the necessary users and an administrator.

4  Provide the service user with the necessary login information and verify that the members of each project can do the following tasks.

a  Open Cloud Assembly.

b  See the infrastructure across all projects.

c  Create a cloud template for the project that they are a member of.

d  Deploy the cloud template to the cloud zone resources defined in the project.

e  Manage their deployments.

5  Assign organization member roles to your cloud template developer users.

If you need instructions, see the User role use case 1: Set up the vRealize Automation Cloud user roles to support a small application development team.

6  Assign roles to a catalog administrator, catalog consumers, and cloud template developers based on their job.

a  Click Add Service Access.

b  Configure the catalog administrator with the following value.

This role might be you, the cloud administrator, or it might be someone else on your application development team.

<table>
<thead>
<tr>
<th>Service</th>
<th>Role</th>
</tr>
</thead>
<tbody>
<tr>
<td>Service Broker</td>
<td>Service Broker Administrator</td>
</tr>
</tbody>
</table>
c  Configure the cloud template consumers with the following value.

<table>
<thead>
<tr>
<th>Service</th>
<th>Role</th>
</tr>
</thead>
<tbody>
<tr>
<td>Service Broker</td>
<td>Service Broker User</td>
</tr>
</tbody>
</table>

**Identity & Access Management**

<table>
<thead>
<tr>
<th>Active Users</th>
<th>Enterprise Groups</th>
</tr>
</thead>
</table>

**Edit Roles**

You are editing the roles of the user Gloria Martinez (gloria@coke.sqa-horizon.local)

Assign Organization Roles Organization Member

Assign Service Roles

<table>
<thead>
<tr>
<th>Service Broker</th>
<th>with roles</th>
<th>Service Broker User</th>
</tr>
</thead>
</table>

+ ADD SERVICE ACCESS

d  Configure the cloud template developers with the following value.

<table>
<thead>
<tr>
<th>Service</th>
<th>Role</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cloud Assembly</td>
<td>Cloud Assembly User</td>
</tr>
</tbody>
</table>

7  Create projects in Cloud Assembly that you use to group resources and users.

In this use case, you create two projects. The first project is PersonnelAppDev and the second is PayrollAppDev.

If you need instructions, see the User role use case 2: Set up vRealize Automation Cloud user roles to support larger development teams and the catalog.

8  Create and release cloud templates for each project team.

If you need instructions, see the User role use case 1: Set up the vRealize Automation Cloud user roles to support a small application development team.

9  Import a Cloud Assembly cloud template into Service Broker.

You must log in as a user with the Service Broker Administrator role.

a  Log in as a user with the Service Broker Administrator role.

b  In the console, click Service Broker.
c. Select **Content and Policies > Content Sources**, and click **New**.

![New Content Source](image)

d. Select **Cloud Assembly Cloud Template**.
e. Enter **PersonnelAppImport** as the name.
f. In the **Source project** drop-down menu, select PersonnelAppDev and click **Validate**.
g. When the source is validated, click **Create and Import**.
h. Repeat for PayrollAppDev using PayrollAppImport as the content source name.

---

10. Share an imported cloud template with a project.

Although the cloud template is already associated with a project, you share it in Service Broker to make it available in the catalog.

a. Continue as a user with the Service Broker administrator role.
b. In Service Broker, select **Content and Policies > Content Sharing**.
c. Select the **PersonnelAppDev** project, which includes the users who must be able to deploy the cloud template from the catalog.
d. Click **Add Items** and then select the PersonnelApp cloud template to share with the project members.

Share Items with PersonnelAppDev

Select the templates to share with the project members.

<table>
<thead>
<tr>
<th>CONTENT SOURCES</th>
</tr>
</thead>
<tbody>
<tr>
<td>Items Shared with this Project</td>
</tr>
<tr>
<td>PersonnelAppImport</td>
</tr>
</tbody>
</table>

e. Click **Save**.

11. Verify that the cloud template is available in the Service Broker catalog to the project members.

   a. Request that a project member log in and click the **Catalog** tab.

   b. Click Request on the PersonnelApp cloud template card.

   c. Complete the form and click **Submit**.
12 Verify that the project member can monitor the deployment process.
   
a  Request that the project member click the **Deployments** tab and locate their provisioning request.

   ![Deployment screenshot](image)

   b  When the cloud template is deployed, verify that the requesting user access the application.

13 Repeat the process for the additional projects.

**Results**

In this use case, recognizing that need to delegate the cloud template development to the developers, you add more organization members. You made them Cloud Assembly users. You then made them members of relevant projects so that they can create and deploy cloud templates. As project members, they cannot see or alter the infrastructure that you continue to manage, but you gave them full service viewer permissions so that they could understand the constraints of infrastructure that they are designing for.

In this use case, you configure users with various roles, including the Service Broker administrator and users. You then provide the non-developer users with the Service Broker catalog.

**What to do next**

To learn how to define and assign custom roles to user, see [User role use case 3: Set up vRealize Automation Cloud custom user roles to refine system roles](#).

**User role use case 3: Set up vRealize Automation Cloud custom user roles to refine system roles**

As a vRealize Automation Cloud organization owner or service administrator, you manage user access using the organization and service system roles. However, you also want to create custom roles to that selected users and perform tasks or see content that is outside of their system roles.

This scenario assumes that you understand the service user and viewer, and the project member and viewer roles that are defined in use case 2. You can see that they are more restrictive than the service and project administrator roles used in use case 1. Now you have identified some local use cases where you want some users to have full management permissions to on some features, view permissions on others, and you do not want them to even view yet another set of features. You use custom roles define those permission.
This use case is based on three possible local use cases. This procedure shows you how to create permissions for the following custom roles.

- **Restricted Infrastructure Administrator.** You want some service users, who are not service administrators, to have broader infrastructure permissions. As the administrator, you want them to help set up cloud zones, images, and flavors. You also want them to be able on on-board and manage discovered resources. Notice they cannot add cloud accounts or integrations, they can only define the infrastructure for those endpoints.

- **Extensibility Developer.** You want some service users to have full permissions to use the extensibility actions and subscriptions as part of cloud template development for their project team and for other projects. They will also develop custom resource types and custom actions for multiple projects.

- **XaaS Developer.** You want some service users to have full permissions to develop custom resource types and custom actions for multiple projects.

- **Deployment Troubleshooter.** You want your project administrators to have permissions they need to troubleshoot and perform root cause analysis on failed deployments. You give them manage permissions on non-destructive or less expensive categories such as image and flavor mappings. You also want the project administrators to have permission to set approvals and day 2 policies as part of the failed deployment troubleshooting role.

**Prerequisites**

- Review the Cloud Assembly and Service Broker service roles and project roles tables in [*What are the vRealize Automation Cloud user roles*](#). You must understand what each service user role can see and do in those services.

- Review the [Custom user roles in vRealize Automation Cloud](#) descriptions so that you know more about how you can refine the permissions for your users.

- Review the first use case so that you understand organization roles and the service administrator roles. See [*User role use case 1: Set up the vRealize Automation Cloud user roles to support a small application development team*](#).

- Review the second use case so that you understand the service user and project member roles. See [*User role use case 2: Set up vRealize Automation Cloud user roles to support larger development teams and the catalog*](#).

- Familiarize yourself with Service Broker. See [*Adding content to the catalog*](#).

**Procedure**

1. Assign organization member roles to your cloud template developer users.

   If you need instructions, see the [*User role use case 1: Set up the vRealize Automation Cloud user roles to support a small application development team*](#).
2 Assign Cloud Assembly and Service Broker service roles for your cloud template developers and catalog consumers.

If you need instructions, see the User role use case 2: Set up vRealize Automation Cloud user roles to support larger development teams and the catalog.

3 Create projects in Cloud Assembly that you use to group resources and users.

The steps below for the custom roles also includes project roles.

If you need instructions for creating projects, see the User role use case 2: Set up vRealize Automation Cloud user roles to support larger development teams and the catalog.

4 Create and release cloud templates for each project team.

If you need instructions, see the User role use case 1: Set up the vRealize Automation Cloud user roles to support a small application development team.

5 Log in to Cloud Assembly as a service administrator and select **Infrastructure > Administration > Custom Roles**.

6 Create a Restricted Infrastructure Administrator role.

In this example, you have a user, Tony, who is expert at setting up the infrastructure for various projects, but you don’t want to give him full service permissions. Instead, Tony builds the core infrastructure the supports the work of all the projects. You give him limited infrastructure management permissions. Tony, or an outside contractor, might also have similar permissions for onboarding discovered machines and bringing them under vRealize Automation Cloud management.

a Add Tony to Cloud Assembly as a service user and viewer.

With his viewer permissions, he can see the underlying cloud accounts and integrations if he needs to troubleshoot his work, but he cannot make changes.

b Create a project and add Tony as project member.

c To create the custom role, select **Infrastructure > Administration > Custom Roles**, and click **New Custom Role**.

d Enter the name **Restricted Infrastructure Administrator** and select the following permissions.

<table>
<thead>
<tr>
<th>Select this permission ...</th>
<th>So that the users can ...</th>
</tr>
</thead>
<tbody>
<tr>
<td>Infrastructure &gt; Manage Cloud Zones</td>
<td>Create, update, and delete cloud zones.</td>
</tr>
<tr>
<td>Infrastructure &gt; Manage Flavor Mappings</td>
<td>Create, update, and delete flavor mappings.</td>
</tr>
<tr>
<td>Infrastructure &gt; Manage Image Mappings</td>
<td>Create, update, and delete image mappings.</td>
</tr>
</tbody>
</table>

e Click **Create**.
f. On the Custom Roles page, select the Restricted Infrastructure Administrator role and click **Assign**.

g. Enter Tony’s email account and click **Add**.

   For example, enter Tony@yourcompany.com.

   You can also enter any defined Active Directory user groups.

h. Have Tony verify that when he logs in, he can add, edit, and delete values in the areas defined by the custom role.

7. Create an Extensibility Developer role.

   In this example, you have several cloud template developers, Sylvia and Igor, who are knowledgeable about how to use extensibility actions and subscriptions to manage daily development tasks. They are also experienced with vRealize Orchestrator, so you task them with providing custom resources and actions for various projects. You give them additional permissions manage extensibility by managing custom resources and actions, and by managing extensibility actions and subscriptions.

   a. Add Sylvia and Igor as Cloud Assembly users.

   b. Add them as members of the projects that they are contributing their extensibility skills to.

   c. Create a custom user role that you name **Extensibility Developer** and select the following permissions.

<table>
<thead>
<tr>
<th>Select this permission ...</th>
<th>So that the users can ...</th>
</tr>
</thead>
<tbody>
<tr>
<td>XaaS &gt; Manage Custom Resources</td>
<td>Create, update, or delete custom resources.</td>
</tr>
<tr>
<td>XaaS &gt; Manage Resource Actions</td>
<td>Create, update, or delete custom actions.</td>
</tr>
<tr>
<td>Extensibility &gt; Manage Extensibility Resources</td>
<td>Create, update, or delete extensibility actions and subscriptions. Disable subscriptions. Cancel and delete action runs.</td>
</tr>
</tbody>
</table>

d. Click **Create**.

e. Assign Sylvia and Igor to the Extensibility Developer role.

f. Verify that Sylvia and Igor can manage the custom resources and actions, and that they can manage the various options on the Extensibility tab.

8. Create a Deployment Troubleshooter role.

   In this example, you give your project administrators more manage permission so that they can remedy deployment failures for their teams.

   a. Add your project administrators, Shauna, Pratap, and Wei, as Cloud Assembly and Service Broker service users.

   b. In their projects, add them as project administrators.
c Create a custom user role that you name **Deployment Troubleshooter** and select the following permissions.

<table>
<thead>
<tr>
<th>Select this permission ...</th>
<th>So that the users can ...</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Infrastructure &gt; Manage Flavor Mappings</strong></td>
<td>Create, update, and delete flavor mappings.</td>
</tr>
<tr>
<td><strong>Infrastructure &gt; Manage Image Mappings</strong></td>
<td>Create, update, and delete image mappings.</td>
</tr>
<tr>
<td><strong>Deployments &gt; Manage Deployments</strong></td>
<td>View all deployments, across projects, and run all day 2 actions on deployments and deployment components.</td>
</tr>
<tr>
<td><strong>Policy &gt; Manage Policies</strong></td>
<td>Create, update, or delete policy definitions.</td>
</tr>
</tbody>
</table>

d Click **Create**.

e Assign Shauna, Pratap, and Wei to the Deployment Troubleshooter role.

f Verify that they can manage flavor mappings, image mappings, and policies in Service Broker.

**Results**

In this use case, you configure different users with various roles, including custom roles that expand their service and project roles.

**What to do next**

Create custom roles that address your local use cases.

**Adding cloud accounts to Cloud Assembly**

Cloud accounts are the configured permissions that Cloud Assembly uses to collect data from the regions or data centers, and to deploy cloud templates to those regions.

The collected data includes the regions that you later associate with cloud zones.

When you later configure cloud zones, mappings, and profiles, you select the cloud account to which they are associated.

As a cloud administrator, you create cloud accounts for the projects in which team members work. Resource information such as network and security, compute, storage, and tags content is data-collected from your cloud accounts.

**Note** If the cloud account has associated machines that have already been deployed in the region, you can bring those machines into Cloud Assembly management by using an onboarding plan. See [What are onboarding plans in Cloud Assembly](#).

If you remove a cloud account that is used in a deployment, resources that are part of that deployment become unmanaged.

Some cloud accounts and some integrations require a cloud proxy.
What are cloud proxies

A cloud proxy routes information to, and collects data from, a Cloud Assembly account and your on-premises applications such as vCenter and NSX and integrations such as Puppet and Google Cloud Platform. You deploy a supplied cloud proxy to a vCenter Server of your choice by using on-screen instructions. You can configure one cloud proxy for multiple vCenter Server instances, or you can have one cloud proxy for a specific vCenter Server.

Cloud accounts that require a cloud proxy:

- vCenter
- NSX-T
- NSX-V
- VMware Cloud on AWS

Integrations that require a cloud proxy:

- vRealize Orchestrator
- Ansible
- Puppet

For details about how the cloud proxy works, see Understanding the VMware Cloud Services Cloud Proxy.

Credentials required for working with cloud accounts in vRealize Automation Cloud

To configure and work with cloud accounts in vRealize Automation Cloud, verify that you have the following credentials.

Required cloud account credentials

<table>
<thead>
<tr>
<th>To...</th>
<th>You need...</th>
</tr>
</thead>
</table>
| Sign up for and log in to Cloud Assembly | A VMware ID.  
  - Set up a My VMware account by using your corporate email address. |
| Connect to vRealize Automation Cloud services | HTTPS port 443 open to outgoing traffic with access through the firewall to:  
  - *.vmwareidentity.com  
  - gaz.csp-vidm-prod.com  
  - *.vmware.com  
  For more information about ports and protocols, see VMware Ports and Protocols. |
<table>
<thead>
<tr>
<th>To...</th>
<th>You need...</th>
</tr>
</thead>
</table>
| Add an Amazon Web Services (AWS) cloud account | Provide a power user account with read and write privileges. The user account must be a member of the power access policy (PowerUserAccess) in the AWS Identity and Access Management (IAM) system.  
  - 20-digit Access Key ID and corresponding Secret Access Key  
If you are using an external HTTP Internet proxy, it must be configured for IPv4.  
vRealize Automation actions-based extensibility (ABX) and external IPAM integration may require additional permissions.  
The following AWS permissions are suggested to allow autoscaling functions:  
  - Autoscaling actions:  
    - autoscaling:DescribeAutoScalingInstances  
    - autoscaling:AttachInstances  
    - autoscaling:DeleteLaunchConfiguration  
    - autoscaling:DescribeAutoScalingGroups  
    - autoscaling:CreateAutoScalingGroup  
    - autoscaling:UpdateAutoScalingGroup  
    - autoscaling:DeleteAutoScalingGroup  
    - autoscaling:DescribeLoadBalancers  
  - Autoscaling resources:  
    - *  
      Provide all autoscaling resource permissions.  
The following permissions are required to allow AWS Security Token Service (AWS STS) functions to support temporary, limited-privilege credentials for AWS identity and access:  
  - AWS STS resources:  
    - *  
      Provide all STS resource permissions.  
The following AWS permissions are required to allow EC2 functions:  
  - EC2 actions:  
    - ec2:AttachVolume  
    - ec2:AuthorizeSecurityGroupIngress  
    - ec2:DeleteSubnet  
    - ec2:DeleteSnapshot  
    - ec2:DescribeInstances  
    - ec2:DeleteTags  
    - ec2:DescribeRegions  
    - ec2:DescribeVolumesModifications  
    - ec2:CreateVpc  
    - ec2:DescribeSnapshots  
    - ec2:DescribeInternetGateways  
    - ec2:DeleteVolume  
    - ec2:DescribeNetworkInterfaces  
    - ec2:StartInstances  
    - ec2:DescribeAvailabilityZones |
<table>
<thead>
<tr>
<th>To...</th>
<th>You need...</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ec2:CreateInternetGateway</td>
</tr>
<tr>
<td></td>
<td>ec2:CreateSecurityGroup</td>
</tr>
<tr>
<td></td>
<td>ec2:DescribeVolumes</td>
</tr>
<tr>
<td></td>
<td>ec2:CreateSnapshot</td>
</tr>
<tr>
<td></td>
<td>ec2:ModifyInstanceAttribute</td>
</tr>
<tr>
<td></td>
<td>ec2:DescribeRouteTables</td>
</tr>
<tr>
<td></td>
<td>ec2:DescribeInstanceTypes</td>
</tr>
<tr>
<td></td>
<td>ec2:DescribeInstanceTypeOfferings</td>
</tr>
<tr>
<td></td>
<td>ec2:DescribeInstanceStatus</td>
</tr>
<tr>
<td></td>
<td>ec2:DetachVolume</td>
</tr>
<tr>
<td></td>
<td>ec2:RebootInstances</td>
</tr>
<tr>
<td></td>
<td>ec2:AuthorizeSecurityGroupEgress</td>
</tr>
<tr>
<td></td>
<td>ec2:ModifyVolume</td>
</tr>
<tr>
<td></td>
<td>ec2:TerminateInstances</td>
</tr>
<tr>
<td></td>
<td>ec2:DescribeSpotFleetRequestHistory</td>
</tr>
<tr>
<td></td>
<td>ec2:DescribeTags</td>
</tr>
<tr>
<td></td>
<td>ec2:CreateTags</td>
</tr>
<tr>
<td></td>
<td>ec2:RunInstances</td>
</tr>
<tr>
<td></td>
<td>ec2:DescribeNatGateways</td>
</tr>
<tr>
<td></td>
<td>ec2:StopInstances</td>
</tr>
<tr>
<td></td>
<td>ec2:DescribeSecurityGroups</td>
</tr>
<tr>
<td></td>
<td>ec2:CreateVolume</td>
</tr>
<tr>
<td></td>
<td>ec2:DescribeSpotFleetRequests</td>
</tr>
<tr>
<td></td>
<td>ec2:DescribeImages</td>
</tr>
<tr>
<td></td>
<td>ec2:DescribeVpcs</td>
</tr>
<tr>
<td></td>
<td>ec2:DeleteSecurityGroup</td>
</tr>
<tr>
<td></td>
<td>ec2:DeleteVpc</td>
</tr>
<tr>
<td></td>
<td>ec2:CreateSubnet</td>
</tr>
<tr>
<td></td>
<td>ec2:DescribeSubnets</td>
</tr>
<tr>
<td></td>
<td>ec2:RequestSpotFleet</td>
</tr>
</tbody>
</table>

**Note**  The SpotFleet request permission is not required for vRealize Automation actions-based extensibility (ABX) or external IPAM integrations.

- **EC2 resources:**
  - *

  Provide all EC2 resource permissions.

The following AWS permissions are required to allow elastic load balancing functions:

- **Load balancer actions:**
  - elasticloadbalancing:DeleteLoadBalancer
  - elasticloadbalancing:DescribeLoadBalancers
  - elasticloadbalancing:RemoveTags
  - elasticloadbalancing:CreateLoadBalancer
  - elasticloadbalancing:DescribeTags
<table>
<thead>
<tr>
<th>To...</th>
<th>You need...</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>elasticloadbalancing:ConfigureHealthCheck</td>
</tr>
<tr>
<td></td>
<td>elasticloadbalancing:AddTags</td>
</tr>
<tr>
<td></td>
<td>elasticloadbalancing:CreateTargetGroup</td>
</tr>
<tr>
<td></td>
<td>elasticloadbalancing:DeleteLoadBalancerListeners</td>
</tr>
<tr>
<td></td>
<td>elasticloadbalancing:DeregisterInstancesFromLoadBalancer</td>
</tr>
<tr>
<td></td>
<td>elasticloadbalancing:RegisterInstancesWithLoadBalancer</td>
</tr>
<tr>
<td></td>
<td>elasticloadbalancing:CreateLoadBalancerListeners</td>
</tr>
<tr>
<td></td>
<td>Load balancer resources:</td>
</tr>
<tr>
<td></td>
<td>*</td>
</tr>
</tbody>
</table>

Provide all load balancer resource permissions.

The following AWS Identity and Access Management (IAM) permissions can be enabled, however they are not required:

- iam:SimulateCustomPolicy
- iam:GetUser
- iam:ListUserPolicies
- iam:GetUserPolicy
- iam:ListAttachedUserPolicies
- iam:GetPolicyVersion
- iam:ListGroupForUser
- iam:ListGroupPolicies
- iam:GetGroupPolicy
- iam:ListAttachedGroupPolicies
- iam:ListGroupVersions
<table>
<thead>
<tr>
<th>To...</th>
<th>You need...</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add a Microsoft Azure cloud account</td>
<td>Configure a Microsoft Azure instance and obtain a valid Microsoft Azure subscription from which you can use the subscription ID. Create an Active Directory application as described in How to: Use the portal to create an Azure AD application and service principal that can access resources in Microsoft Azure product documentation. If you are using an external HTTP Internet proxy, it must be configured for IPv4. Make note of the following information:</td>
</tr>
<tr>
<td>Subscription ID</td>
<td>Allows you to access to your Microsoft Azure subscriptions.</td>
</tr>
<tr>
<td>Tenant ID</td>
<td>The authorization endpoint for the Active Directory applications you create in your Microsoft Azure account.</td>
</tr>
<tr>
<td>Client application ID</td>
<td>Provides access to Microsoft Active Directory in your Microsoft Azure individual account.</td>
</tr>
<tr>
<td>Client application secret key</td>
<td>The unique secret key generated to pair with your client application ID.</td>
</tr>
<tr>
<td>The following permissions are needed for creating and validating Microsoft Azure cloud accounts:</td>
<td></td>
</tr>
<tr>
<td>Microsoft Compute</td>
<td>Microsoft.Compute/virtualMachines/extensions/write</td>
</tr>
<tr>
<td></td>
<td>Microsoft.Compute/virtualMachines/extensions/read</td>
</tr>
<tr>
<td></td>
<td>Microsoft.Compute/virtualMachines/extensions/delete</td>
</tr>
<tr>
<td></td>
<td>Microsoft.Compute/virtualMachines/deallocate/action</td>
</tr>
<tr>
<td></td>
<td>Microsoft.Compute/virtualMachines/delete</td>
</tr>
<tr>
<td></td>
<td>Microsoft.Compute/virtualMachines/powerOff/action</td>
</tr>
<tr>
<td></td>
<td>Microsoft.Compute/virtualMachines/read</td>
</tr>
<tr>
<td></td>
<td>Microsoft.Compute/virtualMachines/restart/action</td>
</tr>
<tr>
<td></td>
<td>Microsoft.Compute/virtualMachines/start/action</td>
</tr>
<tr>
<td></td>
<td>Microsoft.Compute/virtualMachines/write</td>
</tr>
<tr>
<td></td>
<td>Microsoft.Compute/availabilitySets/write</td>
</tr>
<tr>
<td></td>
<td>Microsoft.Compute/availabilitySets/read</td>
</tr>
<tr>
<td></td>
<td>Microsoft.Compute/disks/delete</td>
</tr>
<tr>
<td></td>
<td>Microsoft.Compute/disks/read</td>
</tr>
<tr>
<td>Microsoft Network</td>
<td>Microsoft.Network/loadBalancers/backendAddressPools/join/action</td>
</tr>
<tr>
<td></td>
<td>Microsoft.Network/loadBalancers/delete</td>
</tr>
<tr>
<td></td>
<td>Microsoft.Network/loadBalancers/read</td>
</tr>
<tr>
<td></td>
<td>Microsoft.Network/loadBalancers/write</td>
</tr>
<tr>
<td></td>
<td>Microsoft.Network/networkInterfaces/join/action</td>
</tr>
<tr>
<td></td>
<td>Microsoft.Network/networkInterfaces/read</td>
</tr>
<tr>
<td>To...</td>
<td>You need...</td>
</tr>
<tr>
<td>-------</td>
<td>-------------</td>
</tr>
<tr>
<td></td>
<td>Microsoft.Network/networkInterfaces/write</td>
</tr>
<tr>
<td></td>
<td>Microsoft.Network/networkInterfaces/delete</td>
</tr>
<tr>
<td></td>
<td>Microsoft.Network/networkSecurityGroups/join/action</td>
</tr>
<tr>
<td></td>
<td>Microsoft.Network/networkSecurityGroups/read</td>
</tr>
<tr>
<td></td>
<td>Microsoft.Network/networkSecurityGroups/write</td>
</tr>
<tr>
<td></td>
<td>Microsoft.Network/publicIPAddresses/delete</td>
</tr>
<tr>
<td></td>
<td>Microsoft.Network/publicIPAddresses/join/action</td>
</tr>
<tr>
<td></td>
<td>Microsoft.Network/publicIPAddresses/read</td>
</tr>
<tr>
<td></td>
<td>Microsoft.Network/publicIPAddresses/write</td>
</tr>
<tr>
<td></td>
<td>Microsoft.Network/virtualNetworks/read</td>
</tr>
<tr>
<td></td>
<td>Microsoft.Network/virtualNetworks/subnets/delete</td>
</tr>
<tr>
<td></td>
<td>Microsoft.Network/virtualNetworks/subnets/join/action</td>
</tr>
<tr>
<td></td>
<td>Microsoft.Network/virtualNetworks/subnets/read</td>
</tr>
<tr>
<td></td>
<td>Microsoft.Network/virtualNetworks/subnets/write</td>
</tr>
<tr>
<td></td>
<td>Microsoft.Network/virtualNetworks/write</td>
</tr>
<tr>
<td></td>
<td>Microsoft.Resources/subscriptions/resourcegroups/delete</td>
</tr>
<tr>
<td></td>
<td>Microsoft.Resources/subscriptions/resourcegroups/read</td>
</tr>
<tr>
<td></td>
<td>Microsoft.Resources/subscriptions/resourcegroups/write</td>
</tr>
<tr>
<td></td>
<td>Microsoft.Storage/storageAccounts/delete</td>
</tr>
<tr>
<td></td>
<td>Microsoft.Storage/storageAccounts/listKeys/action</td>
</tr>
<tr>
<td></td>
<td>Microsoft.Storage/storageAccounts/read</td>
</tr>
<tr>
<td></td>
<td>Microsoft.Storage/storageAccounts/write</td>
</tr>
<tr>
<td></td>
<td>Microsoft.Web/sites/read</td>
</tr>
<tr>
<td></td>
<td>Microsoft.Web/sites/write</td>
</tr>
<tr>
<td></td>
<td>Microsoft.Web/sites/delete</td>
</tr>
<tr>
<td></td>
<td>Microsoft.Web/sites/config/read</td>
</tr>
<tr>
<td></td>
<td>Microsoft.Web/sites/config/write</td>
</tr>
<tr>
<td></td>
<td>Microsoft.Web/sites/config/list/action</td>
</tr>
<tr>
<td></td>
<td>Microsoft.Web/sites/publishxml/action</td>
</tr>
<tr>
<td></td>
<td>Microsoft.Web/serverfarms/write</td>
</tr>
<tr>
<td></td>
<td>Microsoft.Web/serverfarms/delete</td>
</tr>
<tr>
<td></td>
<td>Microsoft.Web/sites/hostruntime/functions/keys/read</td>
</tr>
<tr>
<td></td>
<td>Microsoft.Web/sites/hostruntime/host/read</td>
</tr>
<tr>
<td></td>
<td>Microsoft.web/sites/functions/masterkey/read</td>
</tr>
</tbody>
</table>

If you are using Microsoft Azure with action-based extensibility, the following permissions are required, in addition to the minimal permissions:

- Microsoft.Web/sites/read
- Microsoft.Web/sites/write
- Microsoft.Web/sites/delete
- Microsoft.Web/sites/*/action
- Microsoft.Web/sites/config/read
<table>
<thead>
<tr>
<th>To...</th>
<th>You need...</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>- Microsoft.Web/sites/config/write</td>
</tr>
<tr>
<td></td>
<td>- Microsoft.Web/sites/config/list/action</td>
</tr>
<tr>
<td></td>
<td>- Microsoft.Web/sites/publishxml/action</td>
</tr>
<tr>
<td></td>
<td>- Microsoft.Web/serverfarms/write</td>
</tr>
<tr>
<td></td>
<td>- Microsoft.Web/serverfarms/delete</td>
</tr>
<tr>
<td></td>
<td>- Microsoft.Web/sites/hostruntime/functions/keys/read</td>
</tr>
<tr>
<td></td>
<td>- Microsoft.Web/sites/hostruntime/host/read</td>
</tr>
<tr>
<td></td>
<td>- Microsoft.Web/sites/functions/masterkey/read</td>
</tr>
<tr>
<td></td>
<td>- Microsoft.Web/apimanagementaccounts/apis/read</td>
</tr>
<tr>
<td></td>
<td>- Microsoft.Authorization/roleAssignments/read</td>
</tr>
<tr>
<td></td>
<td>- Microsoft.Authorization/roleAssignments/write</td>
</tr>
<tr>
<td></td>
<td>- Microsoft.Authorization/roleAssignments/delete</td>
</tr>
<tr>
<td></td>
<td>- Microsoft.Insights/Components/Read</td>
</tr>
<tr>
<td></td>
<td>- Microsoft.Insights/Components/Write</td>
</tr>
<tr>
<td></td>
<td>- Microsoft.Insights/Components/Query/Read</td>
</tr>
<tr>
<td></td>
<td>If you are using Microsoft Azure with action-based extensibility with extensions, the following permissions are also needed:</td>
</tr>
<tr>
<td></td>
<td>- Microsoft.Compute/virtualMachines/extensions/write</td>
</tr>
<tr>
<td></td>
<td>- Microsoft.Compute/virtualMachines/extensions/read</td>
</tr>
<tr>
<td></td>
<td>- Microsoft.Compute/virtualMachines/extensions/delete</td>
</tr>
<tr>
<td>To...</td>
<td>You need...</td>
</tr>
<tr>
<td>-------</td>
<td>-------------</td>
</tr>
<tr>
<td>Add a Google Cloud Platform (GCP) cloud account</td>
<td>The Google Cloud Platform cloud account interacts with the Google Cloud Platform compute engine. The Project Admin and Owner credentials are required for creating and validating Google Cloud Platform cloud accounts. If you are using an external HTTP Internet proxy, it must be configured for IPv4. The compute engine service must be enabled. When creating the cloud account in vRealize Automation Cloud, use the service account that was created when the compute engine was initialized. The following compute engine permissions are also needed, depending on the actions that the user can take:</td>
</tr>
</tbody>
</table>
| | roles/compute.admin  
Provides full control of all compute engine resources. |
| | roles/iam.serviceAccountUser  
Provides access to users who manage virtual machine instances that are configured to run as a service account. Grant access to the following resources and services: |
| | compute.*  
resourcemanager.projects.get  
resourcemanager.projects.list  
serviceusage.quotas.get  
serviceusage.services.get  
serviceusage.services.list |
| | roles/compute.imageUser  
Provides permission to list and read images without having other permissions on the image. Granting the compute.imageUser role at the project level gives users the ability to list all images in the project. It also allows users to create resources, such as instances and persistent disks, based on images in the project. |
| | compute.images.get  
compute.images.getFromFamily  
compute.images.list  
compute.images.useReadOnly  
resourcemanager.projects.get  
resourcemanager.projects.list  
serviceusage.quotas.get  
serviceusage.services.get  
serviceusage.services.list |
| | roles/compute.instanceAdmin  
Provides permissions to create, modify, and delete virtual machine instances. This includes permissions to create, modify, and delete disks, and also to configure shielded VMBETA settings. For users that manage virtual machine instances (but not network or security settings or instances that run as service accounts), grant this role to the organization, folder, or project that contains the instances, or to the individual instances. |
To... You need...

Users that manage virtual machine instances that are configured to run as a service account also need the roles/iam.serviceAccountUser role.

- compute.acceleratorTypes
- compute.addresses.get
- compute.addresses.list
- compute.addresses.use
- compute.autoscalers
- compute.diskTypes
- compute.disks.create
- compute.disks.createSnapshot
- compute.disks.delete
- compute.disks.get
- compute.disks.list
- compute.disks.resize
- compute.disks.setLabels
- compute.disks.update
- compute.disks.use
- compute.disks.useReadOnly
- compute.globalAddresses.get
- compute.globalAddresses.list
- compute.globalAddresses.use
- compute.globalOperations.get
- compute.globalOperations.list
- compute.images.get
- compute.images.getFromFamily
- compute.images.list
- compute.images.useReadOnly
- compute.instanceGroupManagers
- compute.instanceGroups
- compute.instanceTemplates
- compute.instances
- compute.licenses.get
- compute.licenses.list
- compute.machineTypes
- compute.networkEndpointGroups
- compute.networks.get
- compute.networks.list
- compute.networks.use
- compute.networks.useExternalIP
- compute.projects.get
- compute.regionOperations.get
- compute.regionOperations.list
- compute.regions
- compute.reservations.get
- compute.reservations.list
<table>
<thead>
<tr>
<th>To...</th>
<th>You need...</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>- compute.subnetworks.get</td>
</tr>
<tr>
<td></td>
<td>- compute.subnetworks.list</td>
</tr>
<tr>
<td></td>
<td>- compute.subnetworks.use</td>
</tr>
<tr>
<td></td>
<td>- compute.subnetworks.useExternalIp</td>
</tr>
<tr>
<td></td>
<td>- compute.targetPools.get</td>
</tr>
<tr>
<td></td>
<td>- compute.targetPools.list</td>
</tr>
<tr>
<td></td>
<td>- compute.zoneOperations.get</td>
</tr>
<tr>
<td></td>
<td>- compute.zoneOperations.list</td>
</tr>
<tr>
<td></td>
<td>- compute.zones</td>
</tr>
<tr>
<td></td>
<td>- resourcemanager.projects.get</td>
</tr>
<tr>
<td></td>
<td>- resourcemanager.projects.list</td>
</tr>
<tr>
<td></td>
<td>- serviceusage.quotas.get</td>
</tr>
<tr>
<td></td>
<td>- serviceusage.services.get</td>
</tr>
<tr>
<td></td>
<td>- serviceusage.services.list</td>
</tr>
<tr>
<td></td>
<td>- roles/compute.instanceAdmin.v1</td>
</tr>
</tbody>
</table>

Provides full control of compute engine instances, instance groups, disks, snapshots, and images. Also provides read access to all compute engine networking resources.

**Note** If you grant a user this role at the instance level, that user cannot create new instances.
<table>
<thead>
<tr>
<th>To...</th>
<th>You need...</th>
</tr>
</thead>
<tbody>
<tr>
<td>compute.httpHealthChecks.get</td>
<td></td>
</tr>
<tr>
<td>compute.httpHealthChecks.list</td>
<td></td>
</tr>
<tr>
<td>compute.httpsHealthChecks.get</td>
<td></td>
</tr>
<tr>
<td>compute.httpsHealthChecks.list</td>
<td></td>
</tr>
<tr>
<td>compute.images</td>
<td></td>
</tr>
<tr>
<td>compute.instanceGroups</td>
<td></td>
</tr>
<tr>
<td>compute.instanceTemplate</td>
<td></td>
</tr>
<tr>
<td>compute.instances</td>
<td></td>
</tr>
<tr>
<td>compute.interconnectAttachments.get</td>
<td></td>
</tr>
<tr>
<td>compute.interconnectAttachments.list</td>
<td></td>
</tr>
<tr>
<td>compute.interconnectLocations</td>
<td></td>
</tr>
<tr>
<td>compute.interconnects.get</td>
<td></td>
</tr>
<tr>
<td>compute.interconnects.list</td>
<td></td>
</tr>
<tr>
<td>compute.licenseCodes</td>
<td></td>
</tr>
<tr>
<td>compute.licenses</td>
<td></td>
</tr>
<tr>
<td>compute.machineTypes</td>
<td></td>
</tr>
<tr>
<td>compute.networkEndpointGroups</td>
<td></td>
</tr>
<tr>
<td>compute.networks.get</td>
<td></td>
</tr>
<tr>
<td>compute.networks.list</td>
<td></td>
</tr>
<tr>
<td>compute.networks.use</td>
<td></td>
</tr>
<tr>
<td>compute.networks.useExternalIp</td>
<td></td>
</tr>
<tr>
<td>compute.projects.get</td>
<td></td>
</tr>
<tr>
<td>compute.projects.setCommonInstanceMetadata</td>
<td></td>
</tr>
<tr>
<td>compute.regionBackendServices.get</td>
<td></td>
</tr>
<tr>
<td>compute.regionBackendServices.list</td>
<td></td>
</tr>
<tr>
<td>compute.regionOperations.get</td>
<td></td>
</tr>
<tr>
<td>compute.regionOperations.list</td>
<td></td>
</tr>
<tr>
<td>compute.regions</td>
<td></td>
</tr>
<tr>
<td>compute.reservations.get</td>
<td></td>
</tr>
<tr>
<td>compute.reservations.list</td>
<td></td>
</tr>
<tr>
<td>compute.resourcePolicies</td>
<td></td>
</tr>
<tr>
<td>compute.routers.get</td>
<td></td>
</tr>
<tr>
<td>compute.routers.list</td>
<td></td>
</tr>
<tr>
<td>compute.routes.get</td>
<td></td>
</tr>
<tr>
<td>compute.routes.list</td>
<td></td>
</tr>
<tr>
<td>compute.snapshots</td>
<td></td>
</tr>
<tr>
<td>compute.sslCertificates.get</td>
<td></td>
</tr>
<tr>
<td>compute.sslCertificates.list</td>
<td></td>
</tr>
<tr>
<td>compute.sslPolicies.get</td>
<td></td>
</tr>
<tr>
<td>compute.sslPolicies.list</td>
<td></td>
</tr>
<tr>
<td>compute.sslPolicies.listAvailableFeatures</td>
<td></td>
</tr>
<tr>
<td>compute.subnetworks.get</td>
<td></td>
</tr>
<tr>
<td>compute.subnetworks.list</td>
<td></td>
</tr>
<tr>
<td>compute.subnetworks.use</td>
<td></td>
</tr>
<tr>
<td>To...</td>
<td>You need...</td>
</tr>
<tr>
<td>-------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>- compute.subnetworks.useExternalIp</td>
<td></td>
</tr>
<tr>
<td>- compute.targetHttpProxies.get</td>
<td></td>
</tr>
<tr>
<td>- compute.targetHttpProxies.list</td>
<td></td>
</tr>
<tr>
<td>- compute.targetHttpsProxies.get</td>
<td></td>
</tr>
<tr>
<td>- compute.targetHttpsProxies.list</td>
<td></td>
</tr>
<tr>
<td>- compute.targetInstances.get</td>
<td></td>
</tr>
<tr>
<td>- compute.targetInstances.list</td>
<td></td>
</tr>
<tr>
<td>- compute.targetPools.get</td>
<td></td>
</tr>
<tr>
<td>- compute.targetPools.list</td>
<td></td>
</tr>
<tr>
<td>- compute.targetSslProxies.get</td>
<td></td>
</tr>
<tr>
<td>- compute.targetSslProxies.list</td>
<td></td>
</tr>
<tr>
<td>- compute.targetTcpProxies.get</td>
<td></td>
</tr>
<tr>
<td>- compute.targetTcpProxies.list</td>
<td></td>
</tr>
<tr>
<td>- compute.targetVpnGateways.get</td>
<td></td>
</tr>
<tr>
<td>- compute.targetVpnGateways.list</td>
<td></td>
</tr>
<tr>
<td>- compute.urlMaps.get</td>
<td></td>
</tr>
<tr>
<td>- compute.urlMaps.list</td>
<td></td>
</tr>
<tr>
<td>- compute.vpnTunnels.get</td>
<td></td>
</tr>
<tr>
<td>- compute.vpnTunnels.list</td>
<td></td>
</tr>
<tr>
<td>- compute.zoneOperations.get</td>
<td></td>
</tr>
<tr>
<td>- compute.zoneOperations.list</td>
<td></td>
</tr>
<tr>
<td>- compute.zones</td>
<td></td>
</tr>
<tr>
<td>- resourcemanager.projects.get</td>
<td></td>
</tr>
<tr>
<td>- resourcemanager.projects.list</td>
<td></td>
</tr>
<tr>
<td>- serviceusage.quotas.get</td>
<td></td>
</tr>
<tr>
<td>- serviceusage.services.get</td>
<td></td>
</tr>
<tr>
<td>- serviceusage.services.list</td>
<td></td>
</tr>
</tbody>
</table>

Add an NSX-T cloud account

Provide an account with the following read and write privileges:
- NSX-T Enterprise Administrator role and access credentials
- NSX-T IP address or FQDN
- Permissions required to install a cloud proxy on the vCenter Server instance that manages this NSX-T instance

Administrators also require access to the vCenter Server as described in the following vSphere agent requirements for vCenter-based cloud accounts section on this page.

Add an NSX-V cloud account

Provide an account with the following read and write privileges:
- NSX-V Enterprise Administrator role and access credentials
- NSX-V IP address or FQDN
- Permissions required to install a cloud proxy on the vCenter Server instance that manages this NSX-V instance

Administrators also require access to the vCenter Server as described in the following vSphere agent requirements for vCenter-based cloud accounts section on this page.
To... You need...

Add a vCenter cloud account

Provide an account with the following read and write privileges:
- vCenter IP address or FQDN
- Permissions required to install a cloud proxy on the vCenter Server
  Administrators also require access to the vCenter Server as described in the following vSphere agent requirements for vCenter-based cloud accounts section on this page.

Add a VMware Cloud on AWS (VMC) cloud account

Provide an account with the following read and write privileges:
- The cloudadmin@vmc.local account or any user account in the CloudAdmin group
- NSX Enterprise Administrator role and access credentials
- NSX Cloud Admin access to your organization’s VMware Cloud on AWS SDDC environment
- Administrator access to your organization’s VMware Cloud on AWS SDDC environment
- The VMware Cloud on AWS API token for your VMware Cloud on AWS environment in your organization’s VMware Cloud on AWS service
- vCenter IP address or FQDN
- Permissions required to install a cloud proxy on the vCenter Server
  Administrators also require access to the vCenter that is used by your target VMware Cloud on AWS SDDC that has all the permissions listed in the following vSphere agent requirements for vCenter-based cloud accounts section on this page.
  For more information about the permissions needed to create and use VMware Cloud on AWS cloud accounts, see Managing the VMware Cloud on AWS Data Center in VMware Cloud on AWS product documentation.

vSphere agent requirements for vCenter-based cloud accounts

The following table lists the permissions needed to manage VMware Cloud on AWS and vCenter cloud accounts. The permissions must be enabled for all clusters in the vCenter Server, not just clusters that host endpoints.

For all vCenter Server-based cloud accounts - including NSX-V, NSX-T, vCenter, and VMware Cloud on AWS - the administrator must have vSphere endpoint credentials, or the credentials under which the agent service runs in vCenter, that provide administrative access to the host vCenter Server.

For more information about vSphere agent requirements, see VMware vSphere product documentation.

Table 3-12. Privileges Required for vSphere Agent to Manage vCenter Server Instance

<table>
<thead>
<tr>
<th>Attribute Value</th>
<th>Privilege</th>
</tr>
</thead>
<tbody>
<tr>
<td>Datastore</td>
<td>Allocate space</td>
</tr>
<tr>
<td></td>
<td>Browse datastore</td>
</tr>
<tr>
<td></td>
<td>Low level file operations</td>
</tr>
<tr>
<td>Datastore Cluster</td>
<td>Configure a datastore cluster</td>
</tr>
</tbody>
</table>

Using and Managing VMware Cloud Assembly
Table 3-12. Privileges Required for vSphere Agent to Manage vCenter Server Instance (continued)

<table>
<thead>
<tr>
<th>Attribute Value</th>
<th>Privilege</th>
</tr>
</thead>
<tbody>
<tr>
<td>Folder</td>
<td><img src="attr.png" alt="Create folder" /></td>
</tr>
<tr>
<td></td>
<td><img src="attr.png" alt="Delete folder" /></td>
</tr>
<tr>
<td>Global</td>
<td><img src="attr.png" alt="Manage custom attributes" /></td>
</tr>
<tr>
<td></td>
<td><img src="attr.png" alt="Set custom attribute" /></td>
</tr>
<tr>
<td>Network</td>
<td>Assign network</td>
</tr>
<tr>
<td>Permissions</td>
<td>Modify permission</td>
</tr>
<tr>
<td>Resource</td>
<td><img src="attr.png" alt="Assign VM to Res Pool" /></td>
</tr>
<tr>
<td></td>
<td><img src="attr.png" alt="Migrate powered off virtual machine" /></td>
</tr>
<tr>
<td></td>
<td><img src="attr.png" alt="Migrate powered on virtual machine" /></td>
</tr>
<tr>
<td>Profile-driven Storage</td>
<td><img src="attr.png" alt="Profile-driven storage view" /></td>
</tr>
<tr>
<td></td>
<td>To return a list of storage policies that can be mapped to a storage profile, grant the StorageProfile.View privilege to all accounts that connect vRealize Automation Cloud to vCenter Server.</td>
</tr>
<tr>
<td>Content Library</td>
<td>To assign a privilege on a content library, an administrator must grant the privilege to the user as a global privilege. For related information, see Hierarchical Inheritance of Permissions for Content Libraries in vSphere Virtual Machine Administration at VMware vSphere Documentation.</td>
</tr>
<tr>
<td></td>
<td><img src="attr.png" alt="Add library item" /></td>
</tr>
<tr>
<td></td>
<td><img src="attr.png" alt="Create local library" /></td>
</tr>
<tr>
<td></td>
<td><img src="attr.png" alt="Create subscribed library" /></td>
</tr>
<tr>
<td></td>
<td><img src="attr.png" alt="Delete library item" /></td>
</tr>
<tr>
<td></td>
<td><img src="attr.png" alt="Delete local library" /></td>
</tr>
<tr>
<td></td>
<td><img src="attr.png" alt="Delete subscribed library" /></td>
</tr>
<tr>
<td></td>
<td><img src="attr.png" alt="Download files" /></td>
</tr>
<tr>
<td></td>
<td><img src="attr.png" alt="Evict library item" /></td>
</tr>
<tr>
<td></td>
<td><img src="attr.png" alt="Evict subscribed library" /></td>
</tr>
<tr>
<td></td>
<td><img src="attr.png" alt="Probe subscription information" /></td>
</tr>
<tr>
<td></td>
<td><img src="attr.png" alt="Read storage" /></td>
</tr>
<tr>
<td></td>
<td><img src="attr.png" alt="Sync library item" /></td>
</tr>
<tr>
<td></td>
<td><img src="attr.png" alt="Sync subscribed library" /></td>
</tr>
<tr>
<td></td>
<td><img src="attr.png" alt="Type introspection" /></td>
</tr>
<tr>
<td></td>
<td><img src="attr.png" alt="Update configuration settings" /></td>
</tr>
<tr>
<td></td>
<td><img src="attr.png" alt="Update files" /></td>
</tr>
<tr>
<td></td>
<td><img src="attr.png" alt="Update library" /></td>
</tr>
<tr>
<td></td>
<td><img src="attr.png" alt="Update library item" /></td>
</tr>
<tr>
<td></td>
<td><img src="attr.png" alt="Update local library" /></td>
</tr>
<tr>
<td></td>
<td><img src="attr.png" alt="Update subscribed library" /></td>
</tr>
<tr>
<td></td>
<td><img src="attr.png" alt="View configuration settings" /></td>
</tr>
<tr>
<td>Attribute Value</td>
<td>Privilege</td>
</tr>
<tr>
<td>-------------------------</td>
<td>---------------------------------------------------------------------------</td>
</tr>
<tr>
<td>vSphere Tagging</td>
<td>- Assign or unassign vSphere tag</td>
</tr>
<tr>
<td></td>
<td>- Assign or unassign vSphere tag on object</td>
</tr>
<tr>
<td></td>
<td>- Create a vSphere tag</td>
</tr>
<tr>
<td></td>
<td>- Create a vSphere tag category</td>
</tr>
<tr>
<td></td>
<td>- Delete vSphere tag</td>
</tr>
<tr>
<td></td>
<td>- Delete vSphere tag category</td>
</tr>
<tr>
<td></td>
<td>- Edit vSphere tag</td>
</tr>
<tr>
<td></td>
<td>- Edit vSphere tag category</td>
</tr>
<tr>
<td></td>
<td>- Modify UsedBy field for category</td>
</tr>
<tr>
<td></td>
<td>- Modify UsedBy field for tag</td>
</tr>
<tr>
<td>vApp</td>
<td>- Import</td>
</tr>
<tr>
<td></td>
<td>- vApp application configuration</td>
</tr>
<tr>
<td></td>
<td>The <code>vApp.Import</code> application configuration is required for OVF templates</td>
</tr>
<tr>
<td></td>
<td>and to provision VMs from the content library.</td>
</tr>
<tr>
<td></td>
<td>The <code>vApp.vApp</code> application configuration is required when using cloud-</td>
</tr>
<tr>
<td></td>
<td>init for cloud configuration scripting. This setting allows for modifica-</td>
</tr>
<tr>
<td></td>
<td>tion of a vApp’s internal structure, such as its product information and</td>
</tr>
<tr>
<td></td>
<td>properties.</td>
</tr>
<tr>
<td>Virtual Machine - Invent</td>
<td>- Create from existing</td>
</tr>
<tr>
<td>ary</td>
<td>- Create new</td>
</tr>
<tr>
<td></td>
<td>- Move</td>
</tr>
<tr>
<td></td>
<td>- Remove</td>
</tr>
<tr>
<td>Virtual Machine - Inter</td>
<td>- Configure CD media</td>
</tr>
<tr>
<td>action</td>
<td>- Console interaction</td>
</tr>
<tr>
<td></td>
<td>- Device connection</td>
</tr>
<tr>
<td></td>
<td>- Power off</td>
</tr>
<tr>
<td></td>
<td>- Power on</td>
</tr>
<tr>
<td></td>
<td>- Reset</td>
</tr>
<tr>
<td></td>
<td>- Suspend</td>
</tr>
<tr>
<td></td>
<td>- Tools install</td>
</tr>
</tbody>
</table>

The `vApp.Import` application configuration is required for OVF templates and to provision VMs from the content library. The `vApp.vApp` application configuration is required when using cloud-init for cloud configuration scripting. This setting allows for modification of a vApp’s internal structure, such as its product information and properties.
### Table 3-12. Privileges Required for vSphere Agent to Manage vCenter Server Instance (continued)

<table>
<thead>
<tr>
<th>Attribute Value</th>
<th>Privilege</th>
</tr>
</thead>
<tbody>
<tr>
<td>Virtual Machine - Configuration</td>
<td>• Add existing disk</td>
</tr>
<tr>
<td></td>
<td>• Add new disk</td>
</tr>
<tr>
<td></td>
<td>• Remove disk</td>
</tr>
<tr>
<td></td>
<td>• Advanced</td>
</tr>
<tr>
<td></td>
<td>• Change CPU count</td>
</tr>
<tr>
<td></td>
<td>• Change resource</td>
</tr>
<tr>
<td></td>
<td>• Extend virtual disk</td>
</tr>
<tr>
<td></td>
<td>• Disk change tracking</td>
</tr>
<tr>
<td></td>
<td>• Memory</td>
</tr>
<tr>
<td></td>
<td>• Modify device settings</td>
</tr>
<tr>
<td></td>
<td>• Rename</td>
</tr>
<tr>
<td></td>
<td>• Set annotation</td>
</tr>
<tr>
<td></td>
<td>• Settings</td>
</tr>
<tr>
<td></td>
<td>• Swapfile placement</td>
</tr>
<tr>
<td>Virtual Machine - Provisioning</td>
<td>• Customize</td>
</tr>
<tr>
<td></td>
<td>• Clone template</td>
</tr>
<tr>
<td></td>
<td>• Clone virtual machine</td>
</tr>
<tr>
<td></td>
<td>• Deploy template</td>
</tr>
<tr>
<td></td>
<td>• Read customization specs</td>
</tr>
<tr>
<td>Virtual Machine - State</td>
<td>• Create snapshot</td>
</tr>
<tr>
<td></td>
<td>• Remove snapshot</td>
</tr>
<tr>
<td></td>
<td>• Revert to snapshot</td>
</tr>
</tbody>
</table>

### Configure Microsoft Azure for use with Cloud Assembly

You must gather some information and perform some configuration in order to create a Microsoft Azure cloud account in Cloud Assembly.

**Procedure**

1. Locate and record your Microsoft Azure subscription and tenant IDs.
   - Subscription ID - Click the Subscriptions icon on the left toolbar in your Azure portal to view the subscription ID.
   - Tenant ID - Click the Help icon and select Show Diagnostics in your Azure portal. Search for tenant and record the ID when you have located it.
2 You can create a new storage account and a resource group to get started. Alternatively, you can create these in blueprints later.

- Storage Account - Use the following procedure to configure an account.
  1 In your Azure portal, locate the Storage Accounts icon on the sidebar. Make sure the correct subscription is selected and click Add. You can also, search for storage account in the Azure search field.
  2 Enter the required information for the storage account. You will need your subscription ID.
  3 Select whether to use an existing resource group or create a new one. Make note of your resource group name, as you will need it later.

**Note** Save the location of your storage account as you will need it later.

3 Create a virtual network. Alternatively, if you have a suitable existing network, you can select that one.

If you are creating a network, you must select Use an Existing Resource Group and specify the group that you created in the preceding step. Also, select the same location that you specified previously. Microsoft Azure will not deploy virtual machines or other objects if the location doesn't match between all applicable components that the object will consume.

a Locate the Virtual Network icon on the left panel and click it or search for virtual network. Make sure to select the correct subscription and click Add.

b Enter a unique name for your new virtual network and record it for later.

c Enter the appropriate IP address for your virtual network in the **Address space** field.

d Ensure that the correct subscription is selected and click Add.

e Enter the remaining basic configuration information.

f You can modify the other options as necessary, but for most configurations, you can leave the defaults.

g Click **Create**.

4 Set up an Azure Active Directory application so that vRA can authenticate.

a Locate the Active Directory icon on the Azure left menu and click it.

b Click **App Registrations** and select Add.

c Type a name for your application that complies with Azure name validation.

d Leave Web app/API as the Application Type.

e The Sign-on URL can be anything that is appropriate for your usage.

f Click **Create**.
5 Create a secret key to authenticate the application in Cloud Assembly.
   a Click the name of your application in Azure.
      Make note of your Application ID for later use.
   b Click All Settings in the next pane and select Keys from the settings list.
   c Enter a description for the new key and choose a duration.
   d Click Save and make sure to copy the key value to a safe location as you will be unable to retrieve it later.
   e On the left menu, select API Permissions for the application and click Add a Permission to create a new permission.
   f Select Azure Service Management on the Select an API page.
   g Click Delegated Permissions.
   h Under Select permissions select user_impersonation and then click Add Permissions.

6 Authorize your Active Directory application to connect to your Azure subscription so that you can deploy and manage virtual machines.
   a In the left menu, click the Subscriptions icon, and select your new subscription.
      You may need to click on the text of the name to get the panel to slide over.
   b Select the Access control (IAM) option to see the permissions to your subscription.
   c Click Add under the Add a Role Assignment heading.
   d Choose Contributor from the Role drop down.
   e Leave the default selection in the Assign Access to drop down.
   f Type the name of your application in the Select box.
   g Click Save.
   h Add additional roles so that your new application has Owner, Contributor, and Reader roles.
   i Click the Save.

What to do next
You must install the Microsoft Azure command line interface tools. These tools are freely available for both Windows and Mac operating systems. See the Microsoft documentation for more information about downloading and installing these tools.

When you have the command line interface installed, you must authenticate to your new subscription.

1 Open a terminal window and type your Microsoft Azure login. You will receive a URL and a shortcode that will allow you to authenticate.
2 In a browser, enter the code that you received from the application on your device.

3 Enter your Auth Code and click **Continue**.

4 Select your Azure account and login.

   If you have multiple subscriptions, ensure that the correct one is selected using the `azure account set <subscription-name>` command.

5 Before you proceed, you must register the Microsoft.Compute provider to your new Azure subscription using the `azure provider register microsoft.compute` command.

   If the command times out and generates an error the first time you run it, run it again.

When you have completed configuration, you can use the `azure vm image list` command to retrieve available virtual machine image names. You can choose the desired image and record the URN provided for it and later use it in blueprints.

### Setting up cloud-based vendor accounts

Cloud-based vendors do not require any intermediary services or connectors. You configure a direct connection between the cloud-based target service and Cloud Assembly. The connection ensures that Cloud Assembly can collect information from the system and that you can deploy cloud templates to one or more of the account regions.

After you configure the account, **Cloud Assembly** collects the account region information. When you later configure the cloud zones, mappings, and profiles, you select the regions.

For more information about the required credentials, see [Credentials required for working with cloud accounts in vRealize Automation Cloud](#).

### Create a Microsoft Azure cloud account in vRealize Automation Cloud

As a cloud administrator, you can create a Microsoft Azure cloud account for account regions to which your team will deploy vRealize Automation Cloud cloud templates.

To view an example use case of how Microsoft Azure cloud account works in vRealize Automation Cloud see [Tutorial: Setting up and testing multi-cloud infrastructure and deployments in Cloud Assembly](#).

### Prerequisites

- Verify that you have the required administrator credentials and have enabled HTTPS access on port 443. See [Credentials required for working with cloud accounts in vRealize Automation Cloud](#).

- Verify that you have the required user role. See [What are the vRealize Automation Cloud user roles](#).

- Configure a Microsoft Azure account for use with vRealize Automation Cloud. See [Configure Microsoft Azure for use with Cloud Assembly](#).
Procedure

1. Select **Infrastructure > Connections > Cloud Accounts** and click **Add Cloud Account**.
2. Select the Microsoft Azure account type and enter credentials and other values.
3. Click **Validate**.
   
   The account regions associated with the account are collected.
4. Select the regions to which you want to provision this resource.
5. For efficiency, click **Create a Cloud zone for the selected regions**.
6. If you need to add tags to support a tagging strategy, enter capability tags. See **How do I use tags to manage Cloud Assembly resources and deployments** and **Creating a tagging strategy**.

   For more information about how capability tags and constraint tags help control deployment placements, see the **Constraint Tags and Placement** video tutorial.

7. Click **Save**.

Results

The account is added to vRealize Automation Cloud, and the selected regions are available for the specified cloud zone.

What to do next

Create infrastructure resources for this cloud account.

When you add an Azure cloud account to a cloud template, you can choose to reuse availability sets if you want. Subscriptions have a limit of 2000 availability sets and 25,000 virtual machines, so it makes sense to reuse availability sets when possible. There are two YAML properties that you can use to control how deployments use availability sets. The **availabilitySetName** property enables you to specify an availability set to use. The second property is **doNotAttachAvailabilitySet** which is set to false by default. If this property is set to true, vRealize Automation Cloud will create the deployment with no availability set.

You cannot create a deployment without an availability set if you use a load balancer attached to the virtual machine.

The following table describes how vRealize Automation Cloud behaves depending on whether a resource group and an availability set are specified in the cloud template.

An availability set cannot exist without being part of a resource group. The availability sets in a given resource group must have unique names. Availability sets can have the same name only if they are part of different resource groups.

If you do not specify a resource group name, then vRealize Automation Cloud will create a new resource group, which means that a new availability set must also be created even if a name is passed. The new set will use the name that is passed.
### Table 3-13.

<table>
<thead>
<tr>
<th>Resource Group Specified</th>
<th>Availability Set Specified</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>No</td>
<td>No</td>
<td>vRealize Automation Cloud creates a new resource group and a new availability set for the virtual machine.</td>
</tr>
<tr>
<td>Yes</td>
<td>No</td>
<td>vRealize Automation Cloud reuses the existing resource group and creates a new availability set for the virtual machine.</td>
</tr>
<tr>
<td>No</td>
<td>Yes</td>
<td>vRealize Automation Cloud creates a new resource group and a new availability set with the specified name.</td>
</tr>
<tr>
<td>Yes</td>
<td>Yes</td>
<td>vRealize Automation Cloud reuses the existing resource group. If an availability set with the specified name already exists in that group, it will also be reused. If there is no availability set with the specified name in the group, a new one is created with that name.</td>
</tr>
</tbody>
</table>

Cloud Assembly supports Azure disk snapshots for deployed virtual machines. See [Working with snapshots for Microsoft Azure virtual machine disks](#) for more information.

Cloud Assembly supports several boot diagnostics options for Azure deployments. Boot diagnostics supports debugging of Azure virtual machines and includes collection of log information and relevant screenshots. See [Using boot diagnostics and log analytics with a Microsoft Azure virtual machine](#) for more information.

### Using boot diagnostics and log analytics with a Microsoft Azure virtual machine

You can invoke and configure Microsoft Azure boot diagnostics from an Azure instance in a cloud template. In addition you can also configure log analytics for an Azure virtual machine instance. Boot diagnostics is a debugging feature for Azure virtual machines that facilitates diagnostics for virtual machine boot failures. Using boot diagnostics, a user can monitor the state of a virtual machine as it is booting up by collecting serial log information and screenshots.

### Boot Diagnostics

Boot diagnostics captures serial log information and screenshots and these needs to be saved to the disk. The disk can be of two types, Azure Managed Disk or Unmanaged Disk.

The `bootDiagnostics` YAML property is supported in Azure cloud templates. When this property is set to `true`, boot diagnostics are enabled on the applicable Azure virtual machine deployment.

The following YAML snippet shows an example of how the `bootDiagnostics` property is used.

```yaml
formatVersion: 1
inputs: {}
resources:
  Cloud_Azure_Machine_1:
    type: Cloud.Azure.Machine
metadata:
```
Boot diagnostics can also be invoked on a deployed Azure virtual machine as a day 2 operation. Navigate to the Deployments page in Cloud Assembly and select the Azure deployment. The Actions menu on this page enables you to toggle between Enable Boot Diagnostics and Disable Boot Diagnostics.

After you have deployed a cloud template with boot diagnostics enabled, the Cloud Assembly Deployments page for the deployment will indicate that boot diagnostics are enabled. If you want to disable boot diagnostics, click the Actions menu on the Deployments page and select Disable Boot Diagnostics.

**Log Analytics**

Log Analytics enables you to edit and run log queries on data collected by Azure Monitor Logs, and then interactively analyze the results. You can use Log Analytics queries to retrieve records that match specific criteria to help identify trends and patterns and provide a variety of data insights. By enabling Log Analytics on a Azure virtual machine, that machine will act as a data source.

Before you can configure log analytics in a Cloud Assembly cloud template, you must create and configure an Azure Log Analytics workspace. You can do this using the Virtual Machines option in the Azure Monitor menu. See the Microsoft Azure documentation for more information.

To configure log analytics, you must have the Azure Workspace ID and Workspace Key. You can find these on the Agent Management tab in Azure under the Log Analytics Workspace.

The following cloud template example shows how log analytics can be configured using extensions.

```json
formatVersion: 1
inputs: {}
resources:
  Cloud_Azure_Machine_1:
    type: Cloud.Azure.Machine
    properties:
      image: ubuntu
      flavor: small
      bootDiagnostics: true
    extensions:
      - autoUpgradeMinorVersion: true
        name: test-loga
        protectedSettings:
          workspaceKey: xxxxxxxxxx
          publisher: Microsoft.EnterpriseCloud.Monitoring
          settings:
        workspaceId: aaaaaaaaa
```
After you have deployed a cloud template with Log Analytics enabled, you can enable or disable it using the Actions menu options on the Cloud Assembly Deployments page for the deployment.

**Working with snapshots for Microsoft Azure virtual machine disks**

You can create full or incremental snapshots of Microsoft Azure managed disks.

The VMware Cloud Assembly Deployments page for an Azure deployment contains an Actions menu that provides several options for creating and deleting snapshots from Azure deployments on virtual machine managed disks and on independent managed disks. The following list outlines the specific snapshot functionality that is supported.

- Create a disk snapshot - Supported for both external and compute disks. You can also create snapshots for a disk in a different resource group.
- Delete a disk snapshot - Supported for external disks only
- Encrypt snapshots using an Azure disk encryption set.
- You can provide key-value pairs as tags during snapshot creation.

Snapshots on unmanaged disks are currently not supported.

If you use encryption, the current snapshot implementation supports platform-managed key encryption. By default, the network policy allows access from everywhere, so restricting access to snapshots via the network policy is not possible.

For more information about using the VMware Cloud Assembly Actions and the Deployments page, see [What actions can I run on Cloud Assembly deployments](#).


**Create an Amazon Web Services cloud account in vRealize Automation Cloud**

As a cloud administrator, you can create an Amazon Web Services (AWS) cloud account for account regions to which your team will deploy vRealize Automation Cloud cloud templates.

For authorized users, AWS cloud accounts support access to the AWS GovCloud configuration. This configuration supports most of the standard vRealize Automation Cloud cloud account functionality with regard to project configuration, tags, and infrastructure. In Cloud Assembly cloud templates, it does support use of AWS Platform as a Service (PaaS) properties.

The following procedure describes how to configure an AWS cloud account.

**Prerequisites**

- Verify that you have the required administrator credentials and have enabled HTTPS access on port 443. See [Credentials required for working with cloud accounts in vRealize Automation Cloud](#).
- Verify that you have the required user role. See What are the vRealize Automation Cloud user roles.
- Verify that you have required AWS administrator credentials.

**Procedure**

1. Select **Infrastructure > Connections > Cloud Accounts** and click **Add Cloud Account**.
2. Select the AWS account type, and enter credentials and other values.
3. Click **Validate**.
   - The account regions associated with the account are collected.
4. Select the regions to which you want to provision this resource.
5. For efficiency, click **Create a Cloud zone for the selected regions**.
6. If you need to add tags to support a tagging strategy, enter capability tags. See How do I use tags to manage Cloud Assembly resources and deployments and Creating a tagging strategy.
   - For more information about how capability tags and constraint tags help control deployment placements, see the Constraint Tags and Placement video tutorial.
7. Click **Add**.

**Results**

The account is added to vRealize Automation Cloud, and the selected regions are available for the specified cloud zone.

**What to do next**

Configure infrastructure resources for this cloud account.

**Create a Google Cloud Platform cloud account in vRealize Automation Cloud**

As a cloud administrator, you can create a Google Cloud Platform (GCP) cloud account for account regions to which your team will deploy vRealize Automation Cloud cloud templates.

**Prerequisites**

- Verify that you have the required administrator credentials and have enabled HTTPS access on port 443. See Credentials required for working with cloud accounts in vRealize Automation Cloud.
- Verify that you have the required user role. See What are the vRealize Automation Cloud user roles.
- Verify that you have access to the Google Cloud Platform JSON security key.
Verify that you have required security information for your Google Cloud Platform instance. You can obtain most of this information from your instance or from the Google documentation.

Procedure

1. In VMware Cloud Assembly, select **Infrastructure > Connections > Cloud Accounts** and click **Add Cloud Account**.

2. Select the Google Cloud Platform account type and enter the appropriate credentials and related information. Use the service account that was created when the source GCP account compute engine was initialized.

   As noted in the **Prerequisites** section above, credential requirements are available at [Credentials required for working with cloud accounts in vRealize Automation Cloud](#). To successfully create the cloud account in vRealize Automation Cloud, the source GCP account must have the compute engine service enabled.

   In vRealize Automation Cloud, the project ID is part of the Google Cloud Platform endpoint. You specify it when you create the cloud account. During data collection of project-specific private images, the vRealize Automation Cloud GCP adapter queries the Google Cloud Platform API.

3. Click **Validate**.

   The account regions associated with the account are collected.

4. Select the regions to which you want to provision this resource.

5. For efficiency, click **Create a Cloud zone for the selected regions**.

6. If you need tags to support a tagging strategy, enter capability tags. See [How do I use tags to manage Cloud Assembly resources and deployments](#) and [Creating a tagging strategy](#).

   For more information about how capability tags and constraint tags help control deployment placements, see the [Constraint Tags and Placement](#) video tutorial.

7. Click **Add**.

Results

The account is added to vRealize Automation Cloud, and the selected regions are available for the specified cloud zone.

What to do next

Create infrastructure resources for this cloud account.

The following paragraphs provide some information on deploying a Google Cloud Platform virtual machine from VMware Cloud Assembly.
When you add a Google Cloud Platform cloud account to a Cloud Assembly cloud template, you can use the `useSoleTenant` YAML property to indicate that you want to deploy a virtual machine to a sole tenant node. This configuration enables you to isolate virtual machines for security, privacy or others issues.

To facilitate this functionality, Google Cloud Platform node affinity labels are converted to tags in Cloud Assembly, and these tags are applied on relevant vRealize Automation Cloud availability zones where node groups reside. When the `useSoleTenant` property is set to true, constraint tags must be one of the node affinity labels. Also, to deploy a machine in sole tenant mode, you must include the `useSoleTenant` property in the cloud template as well as the constraint tags.

Before using this feature, you must create the appropriate node template and node affinity labels in Google Cloud Platform and then create a node group.

The following YAML example shows how the `useSoleTenant` property can be used in Cloud Assembly cloud templates. The constraint tags are the node affinity labels that were auto-collected from your Google Cloud Platform server.

```
resources:
  Cloud_GCP_Machine_1:
    type: Cloud.GCP.Machine
    properties:
      image: ubuntu
      flavor: c2-family
      name: demo-vm
      useSoleTenant: true
    constraints:
      - tag: 'env:prod'
      - tag: 'region:asia-east1'
```

Setting up cloud accounts in Cloud Assembly that require a cloud proxy

Some cloud accounts require a deployed cloud proxy to manage the connection between the vendor account and Cloud Assembly. You can create a cloud proxy to manage one or more vCenter-based cloud accounts. You can also create a cloud proxy to manage certain vCenter-based integrations, while other integrations require a cloud extensibility proxy.

If your organization is spread across multiple geographic areas, you might want to set up multiple cloud proxies to spread the workload. For example, you might consider setting up cloud proxies by geography, by project, or by deployment intent.

When you add a cloud account that is associated to a deployed cloud proxy, Cloud Assembly collects information about that account such as network and machine data.

You can configure a cloud proxy before you create a cloud account or during the process of creating a cloud account. A common practice is to first configure a cloud proxy on a particular vCenter and then select the existing cloud proxy when you create a cloud account.
for related information about cloud proxies, see Adding cloud accounts to Cloud Assembly. For details about how the cloud proxy works, see Understanding the VMware Cloud Services Cloud Proxy.

The step values that you enter in this procedure are only use case examples. You won’t be able to use them letter-by-letter in your environment unless otherwise stated in the procedure.

Add a cloud proxy to a vCenter Server in Cloud Assembly

You can install a new cloud proxy as part of the vCenter-based cloud account creation process, or you can select a previously installed cloud proxy as you create the cloud account in Cloud Assembly.

In this example, you install a cloud proxy on a target vCenter Server first, and then select it as you create a vCenter-based cloud account or integration. The installation process deploys the cloud proxy.

**Note** This procedure describes the process for downloading and deploying the cloud proxy OVA (VMware-Cloud-Services-Data-Collector.ova). Download and deployment of the similar-but-different cloud extensibility proxy OVA (VMware-Extensibility-Appliance-SAAS.ova, which is used by integration points such as IPAM and on-premises action-based extensibility, is described at Download and deploy a cloud extensibility proxy.

The supplied cloud proxy OVA contains the credentials and protocols you need to connect a proxy appliance on a host vCenter Server to vCenter-based cloud accounts and integrations in Cloud Assembly. You download the OVA and deploy it to a target vCenter Server. Once the cloud proxy is installed and running, you can associate it to one or more vCenter-based cloud accounts and integrations.

**Note** The cloud proxy OVA must be deployed on a vCenter Server. Deployment to an ESX server is not supported. For more information about cloud proxy deployment specifics, see VMware Cloud Services Cloud Proxy.

A VMware Cloud on AWS cloud account can only be associated to a cloud proxy that is deployed to a vCenter in a target VMware Cloud on AWS SDDC.

Network proxies that perform TLS termination are not supported.

**Prerequisites**

- Verify that you have cloud administrator credentials. See Credentials required for working with cloud accounts in vRealize Automation Cloud.
- Verify that you have the cloud administrator user role. See What are the vRealize Automation Cloud user roles.
- If you are creating a cloud proxy to use with a VMware Cloud on AWS cloud account, see Create and deploy a cloud proxy for a VMware Cloud on AWS cloud account in vRealize Automation Cloud.
To support the cloud proxy, whitelist access to the following domains is required. For more detailed information about cloud proxy requirements, see the Understanding the VMware Cloud services cloud proxy configuration document:

- **ci-data-collector.s3.amazonaws.com** – enables Amazon Web Services S3 access for cloud proxy OVA download.
- **symphony-docker-external.jfrog.io** - allows JFrog Artifactory to access Docker images.
- **data.mgmt.cloud.vmware.com** - enables the data pipeline service connection to VMware Cloud services for secure data communication between cloud and on-premises elements. For non-US regions, substitute the region value. For example, for the UK, use uk.data.mgmt.cloud.vmware.com and for Japan, use ja.data.mgmt.cloud.vmware.com.
- **api.mgmt.cloud.vmware.com** – enables the Web API and cloud proxy service connection to VMware Cloud services. For non-US regions, substitute the region value. For example, for the UK, use uk.api.mgmt.cloud.vmware.com and for Japan, use ja.api.mgmt.cloud.vmware.com.
- **console.cloud.vmware.com** – enables the Web API and cloud proxy service connection to the VMware Cloud service. For non-US regions, substitute the region value. For example, for the UK, use uk.console.cloud.vmware.com.

**Procedure**

1. Select **Infrastructure > Connections > Cloud Proxies** and click **New Cloud Proxy**.

2. Download the cloud proxy OVA.
3 Rename the downloaded OVA something unique, for example `vCenter1_vmc_va.ova`.

You can download and name the cloud proxy OVA to make it obvious what it is used for, for example when using a particular cloud account for a particular purpose.

4 Navigate to your vSphere Web Client data center, click the name of your vCenter Server cluster, and select **Deploy OVF Template**.

5 Enter information as prompted to install the OVA on your vCenter Server.

6 When prompted to enter the key or token, return to the **Install Cloud Proxy** page and click **Copy**.

7 (Optional) To configure additional security and force connections to pass through a proxy server, use the **Network Proxy Hostname/IP Address**, **Network Proxy Port**, **Network Proxy Username**, and **Network Proxy Password** options to configure the network proxy:

```
Deploy OVF Template

[ ] 1. Select an OVF template
[ ] 2. Select a name and folder
[ ] 3. Select a compute resource
[ ] 4. Review details
[ ] 5. License agreements
[ ] 6. Select storage
[ ] 7. Select networks
[ ] 8. Customize template
[ ] 9. Ready to complete

Cloud Proxy Display Name
Identifier for this cloud proxy on the VMware Cloud Services Portal

Network Proxy Hostname/IP Address
Leave blank if no Network Proxy being used

Network Proxy Port
Leave blank if no Network Proxy being used

Network Proxy Username
Leave blank if no Authentication required

Network Proxy Password
Leave blank if no Authentication required

Network Proxy Password

For details about configuring these cloud proxy settings, see **Understanding the VMware Cloud services cloud proxy**.

8 Return to the vSphere Web Client and paste the provided key value to install the cloud proxy’s virtual appliance.

9 In Cloud Assembly, wait for a connection to be made with your vSphere Web Client and then click **Done**.

It might take several minutes to connect.

10 (Optional) To configure a network proxy after the cloud proxy virtual appliance is deployed, modify the **configure-network-proxy** file in the cloud proxy VA by using the following procedure:

a SSH into the cloud proxy VA.

b Open the `/root/configure-network-proxy` file and provide the network proxy configuration settings.

c Save the file.
What to do next

To verify that the cloud proxy is running, see **Verify that a cloud proxy is running on a target virtual machine**.

You can now add vCenter-based cloud accounts and integrations that require the cloud proxy.

**Verify that a cloud proxy is running on a target virtual machine**

When you work in Cloud Assembly, you can check to see if an existing cloud proxy is running on a virtual machine on which it has been deployed.

A deployed cloud proxy service running on a host virtual machine is required for communication and data collection with vCenter-based cloud accounts and vCenter-based integrations when you work in Cloud Assembly. If the cloud proxy service is not running, or if the host virtual machine on which the cloud proxy service has been deployed is not running, communication cannot occur.

**Note** As an alternative to this procedure, you can test the status of a cloud proxy while working in Cloud Assembly by clicking **Infrastructure > Cloud Proxies**, selecting the cloud proxy, and selecting **Status** from the drop-down actions menu.

**Prerequisites**

- Verify that you have the cloud administrator user role. See **What are the vRealize Automation Cloud user roles**.

- If you are using VMware Cloud on AWS, verify that you have VMware Cloud on AWS CloudAdmin credentials for the target SDDC in vCenter. See **Credentials required for working with cloud accounts in vRealize Automation Cloud**.

**Procedure**

1. Log in to the virtual machine where the cloud proxy has been deployed by using any of the following methods:

   - IP address of the virtual machine
   - SSH into the virtual machine
   - Interactive window in the vSphere web client

   After you log in, URLs in the external network connectivity section are listed as **Connected**.
If a URL in the external network connectivity section is listed as Connection Failed, verify that the virtual machine is connected to the internet and that the URL is available.

- To display white-listed URLs, run the following command.
  
  ```bash
  /data-collector-status --requiredUrls
  ```

- To test if the URLs are reachable, run the following command.

  ```bash
  /data-collector-status --traceroute
  ```

2. To display the status of all containers that are running on the VA, open a command window and run the following command.

  ```bash
  /data-collector-status --containers
  ```

The output is as follows:

```
VMware Cloud Services Data Collector
Status Page
(Detailed)
-----------------------Running Containers------------------------
CONTAINER ID     IMAGE                                           NAMES                  CREATED                STATUS
5ae006245dca     symphony-docker-external.jfrog.io/vmware/tango-vro-agent:486  vro-agent     About an hour       Up About an hour
a6164d68a6c3     symphony-docker-external.jfrog.io/vmware/cloudassembly-cmx-agent:223  cloudassembly-cmx-agent 7 hours       Up 7 hours
b05cc6b7f240     symphony-docker-external.jfrog.io/symphony/codestream-lemans-agent:150  codestream-lemans-agent 29 hours      Up 29 hours
832d24d7b603     symphony-docker-external.jfrog.io/cloudassembly-sddc-agent:36eb79eb5  cloudassembly-sddc-agent 29 hours      Up 29 hours
14812ac95571     symphony-docker-external.jfrog.io/symphony/cloudassembly-blueprint-agent:14  cloudassembly-blueprint-agent 29 hours      Up 29 hours
```
To determine if the cloud proxy service on the VM is running, run the following command.

```
systemctl status rdc-proxy
```

If the service is running, your result is similar to the following.

If the service is not running, enter the following command to examine the service log.

```
vi/data1/proxylogs/va-remote-proxy.0.log
```

Look for a `startup:true` line similar to the following:

Relevant logs include the following:

- `/data1/*`
- `/usr/local/bin/proxy.properties`
- `/opt/vmware/var/log/firstboot`
- `/opt/vmware/var/log/subsequentboot`

You can restart a cloud proxy service by selecting the **Restart** option.

The restart process takes approximately 5 minutes to reestablish connectivity.

**Collect a cloud proxy log bundle in Cloud Assembly**

You can obtain a log bundle from the cloud proxy.

A log bundle can help you troubleshoot certain issues and scenarios, such as the following:

- The `cloudassembly-sddc-agent` is not present, fails to start, or is not the latest version.
- The connection is lost, as evidenced by messaging stating that the cloud proxy connection has been lost.
- The endpoint connection can't be validated, as evidenced by any of the following messages:
  - Agent not found
  - Cannot get proxy with id ......
  - Session doesn't exist ..... 
- The enumeration fails, as evidenced by the following or similar message:
The connection with vAPI endpoint cannot be established. Please make sure the vAPI endpoint is running in the vCenter. Unexpected character (’<’ (code 60)): expected a valid value (number, String, array, object, ’true’, ’false’ or ’null’).

- The enumeration never finishes.
- A VM provisioning action fails.
- A VM Day 2 post-provisioning action fails.

**Procedure**

1. Verify that you have cloud administrator credentials. See [Credentials required for working with cloud accounts in vRealize Automation Cloud](#).

2. Verify that you have the cloud administrator user role. See [What are the vRealize Automation Cloud user roles](#).

3. Verify that you have an Amazon Web Services account, and administrator credentials to access that account.

4. Generate the log bundle by using either of the following two procedures:

   - **User interface method**
     
     Use this method to view the logs. However, if you want to send the log bundle to VMware for help, use the command line method instead.

     1. In Cloud Assembly, select **Infrastructure > Connections > Cloud Proxies**.
     
     2. Select the proxy VA that is associated with the cloud account in question and click **Upload Support Bundle**.

        The logs are uploaded to a secure Amazon AWS S3 site and tagged with the cloud proxy ID. Be sure to record the cloud proxy ID so that you can use it to find the logs in AWS S3.

        The log bundle may take a few minutes to generate.

   - **Command line method**

     Use this method when sending logs to VMware for help with troubleshooting your issues.

     1. Open a command prompt window and SSH into the cloud proxy VA.
     
     2. Run the following command:

```
   sh /generate-log-bundle
```

     The zipped log file bundle is created in the `/root/log-bundle-directory` folder.

     3. Unzip the `log_bundle.tar.gz` file to display the following file structure:

```
   ├── data1
   │    ├── blueprint-agent-logs
   │    └── cmx-agent-logs
```
The vCenter, NSX, and VMware Cloud on AWS logs are in the `sddc-agent-logs` folder.

**Create a vCenter cloud account in vRealize Automation Cloud**

You can add a vCenter cloud account for the account regions to which you want to deploy vRealize Automation Cloud cloud templates.

For network and security purposes, you can associate a vCenter cloud account with an NSX-T or NSX-V cloud account.

An NSX-T cloud account can be associated to one or more vCenter cloud accounts. However, an NSX-V cloud account can only be associated to one vCenter cloud account.

**Prerequisites**

- Verify that you have the required administrator credentials and have enabled HTTPS access on port 443. See [Credentials required for working with cloud accounts in vRealize Automation Cloud](#).
- Verify that you have the cloud administrator user role. See [What are the vRealize Automation Cloud user roles](#).
- Verify that you have a cloud proxy to use with this cloud account. See [Add a cloud proxy to a vCenter Server in Cloud Assembly](#).
- Verify that the cloud proxy VM is on and that the cloud proxy service is connected and running. See [Verify that a cloud proxy is running on a target virtual machine](#).

**Procedure**

1. Select **Infrastructure > Connections > Cloud Accounts** and click **Add Cloud Account**.
2. Select the vCenter account type and enter the vCenter Server host IP address.
3 Select an existing cloud proxy from the drop-down menu.
   You can also create a new cloud proxy for this cloud account. See Add a cloud proxy to a vCenter Server in Cloud Assembly.

4 Enter your vCenter Server administrator credentials and click Validate.
   All data centers that are associated with the account are data-collected. The following elements are data-collected, as are all vSphere tags for the following elements:
   - Machines
   - Clusters and hosts
   - Port groups
   - Data stores

5 Select at least one of the available data centers on the specified vCenter Server to allow provisioning for this cloud account.

6 For efficiency, create a cloud zone for provisioning to the selected data centers.
   You can also create cloud zones as a separate step according to your organization's cloud strategy.
   For information about cloud zones, see Learn more about Cloud Assembly cloud zones.

7 Select an existing NSX cloud account.
   You can select the NSX account now, or later when you edit the cloud account.
   For information about NSX-V cloud accounts, see Create an NSX-V cloud account in vRealize Automation.
   For information about NSX-T cloud accounts, see Create an NSX-T cloud account in vRealize Automation Cloud.
   For information about making association changes after you have deployed a cloud template, see What happens if I remove an NSX cloud account association in vRealize Automation Cloud.

8 If you want to add tags to support a tagging strategy, enter capability tags.
   You can add tags now, or later when you edit the cloud account. For information about tagging, see How do I use tags to manage Cloud Assembly resources and deployments.
   For more information about how capability tags and constraint tags help control deployment placements, see the Constraint Tags and Placement video tutorial.

9 Click Save.
Results

The cloud account is added and the selected data centers are available for the specified cloud zone. Collected data such as machines, networks, storage, and volumes is listed in the Resources section of the Infrastructure tab.

What to do next

Configure remaining infrastructure resources for this cloud account. See Chapter 4 Building your Cloud Assembly resource infrastructure.

Create an NSX-V cloud account in vRealize Automation

For network and security purposes, you can create and associate an NSX-V cloud account with a vCenter cloud account.

An NSX-V cloud account can only be associated to one vCenter cloud account.

The association between NSX-V and a vCenter cloud account must be configured outside of vRealize Automation, specifically in your NSX application. vRealize Automation doesn't create the association between NSX and vCenter. In vRealize Automation, you specify an association that already exists in NSX.

Prerequisites

- Verify that you have the required administrator credentials and have enabled HTTPS access on port 443. See Credentials required for working with cloud accounts in vRealize Automation Cloud.
- Verify that you have the cloud administrator user role. See What are the vRealize Automation Cloud user roles.
- Verify that you have a cloud proxy to use with this NSX cloud account. See Add a cloud proxy to a vCenter Server in Cloud Assembly.
- Verify that you have a vCenter cloud account to use with this NSX cloud account. See Create a vCenter cloud account in vRealize Automation Cloud.
- Verify that the cloud proxy VM is on and that the cloud proxy service is connected and running. See Verify that a cloud proxy is running on a target virtual machine.

Procedure

1. Select Infrastructure > Connections > Cloud Accounts and click Add Cloud Account.
2. Select the NSX-V account type and enter the NSX-V host IP address.
3. Select an existing cloud proxy from the drop-down menu.
   - You can also create a new cloud proxy for this cloud account. See Add a cloud proxy to a vCenter Server in Cloud Assembly.
4. Enter your NSX administrator credentials and click Validate.
   - The assets associated with the account are collected.
If the NSX host IP address is not available, or if the cloud proxy is not associated with the NSX host IP address in the vCenter Server on which the cloud proxy is deployed, then validation fails.

5 If available, select the vCenter endpoint that represents the vCenter cloud account that you are associating with this NSX-V account.

Only vCenter cloud accounts that are not currently associated to an NSX-T or NSX-V cloud account are available for selection.

For information about making association changes after you have deployed a cloud template, see What happens if I remove an NSX cloud account association in vRealize Automation Cloud.

6 If you want to add tags to support a tagging strategy, enter capability tags.

You can add or remove capability tags later. See How do I use tags to manage Cloud Assembly resources and deployments.

For information about how capability tags and constraint tags help control deployment placements, see the Constraint Tags and Placement video tutorial.

7 Click Save.

What to do next

You can create or edit a vCenter cloud account to associate with this NSX cloud account. See Create a vCenter cloud account in vRealize Automation Cloud.

Create and configure one or more cloud zones for use with the data centers that are used by this cloud account. See Learn more about Cloud Assembly cloud zones.

Configure infrastructure resources for this cloud account. See Chapter 4 Building your Cloud Assembly resource infrastructure.

Create an NSX-T cloud account in vRealize Automation Cloud

For network and security purposes, you can create an NSX-T cloud account and associate it with one or more vCenter cloud accounts.

An NSX-T cloud account can be associated to one or more vCenter cloud accounts. However, an NSX-V cloud account can only be associated to one vCenter cloud account.

The association between NSX-T and one or more vCenter cloud accounts must be configured outside of vRealize Automation Cloud, specifically in your NSX application. vRealize Automation doesn’t create the association between NSX and vCenter. In vRealize Automation Cloud, you specify one or more configuration associations that already exists in NSX.

When you create an NSX-T cloud account in vRealize Automation Cloud, you specify a manager type and an NSX mode. These selections cannot be changed after you create the cloud account.
You can connect to an NSX-T Global Manager and configure an association between an NSX-T Global Manager and local managers in the context of the NSX-T federation.

For related information about NSX-T options and capabilities in general, see [NSX-T Data Center product documentation](#).

To facilitate fault tolerance and high availability in deployments, each NSX-T data center endpoint represents a cluster of three NSX Managers.

- vRealize Automation Cloud can point to one of the NSX Managers. Using this option, one NSX Manager receives the API calls from vRealize Automation Cloud.

- vRealize Automation Cloud can point to the Virtual IP of the cluster. Using this option, one NSX Manager assumes control of the VIP. That NSX Manager receives the API calls from vRealize Automation Cloud. In case of failure, another node in the cluster assumes control of the VIP and receives the API calls from vRealize Automation Cloud.

  For more information about VIP configuration for NSX, see [Configure a Virtual IP (VIP) Address for a Cluster](#) in the [*NSX-T Data Center Installation Guide*](#) at [VMware NSX-T Data Center Documentation](#).

- vRealize Automation Cloud can point to a load balancer VIP to load-balance the calls to the three NSX Managers. Using this option, all three NSX Managers receive API calls from vRealize Automation Cloud.

  You can configure the VIP on a third-party load balancer or on an NSX-T load balancer.

  For large scale environments, consider using this option to split the vRealize Automation Cloud API calls among the three NSX Managers.

### Prerequisites

- Verify that you have the required administrator credentials and have enabled HTTPS access on port 443. See [Credentials required for working with cloud accounts in vRealize Automation Cloud](#).

- Verify that you have the cloud administrator user role. See [What are the vRealize Automation Cloud user roles](#).

- Verify that you have a cloud proxy to use with this NSX cloud account. See [Add a cloud proxy to a vCenter Server in Cloud Assembly](#).

- Verify that you have a vCenter cloud account to use with this NSX cloud account. See [Create a vCenter cloud account in vRealize Automation Cloud](#).

- Verify that the cloud proxy VM is on and that the cloud proxy service is connected and running. See [Verify that a cloud proxy is running on a target virtual machine](#).

### Procedure

1. Select **Infrastructure > Connections > Cloud Accounts** and click **Add Cloud Account**.

2. Select the NSX-T account type and specify a cloud account name and description.
Enter the host IP address for the NSX-T Manager instance or VIP (see above for information about the expected behavior that pertains to the NSX Manager and VIP options).

4 Select an existing cloud proxy from the drop-down menu.

You can also create a new cloud proxy for this cloud account. See Add a cloud proxy to a vCenter Server in Cloud Assembly.

5 Enter your NSX user name and password administrator credentials.

6 For Manager type, select either Global or Local (default).
   - Global Manager
     The Global Manager setting is only available for use with the Policy NSX mode setting. It is not available when using the Manager NSX mode setting.
     The Global setting refers to the NSX-T federation capabilities, including global network segments. Only NSX-T cloud accounts with the Global setting support NSX-T federation.
     When using the Global Manager setting, you are prompted to identify a Local Manager NSX-T cloud account and an associated vCenter Server cloud account.
     You cannot associate a Global Manager NSX-T cloud account with vCenter cloud account, as you can with an Local Manager NSX-T cloud account. Similar to how a Local Manager NSX-T cloud account can be associated to multiple vCenter cloud accounts, a Global Manager NSX-T cloud account can be associated to multiple Local Manager NSX-T cloud accounts.
   - Local Manager
     Use the Local setting to define a traditional NSX-T cloud account, which can be associated to one or more vSphere cloud accounts. You can associate a Global manager NSX-T cloud account with a Local NSX-T cloud accounts. Note that this is also the setting to use if you are creating a new and empty target NSX-T cloud account for the purposes of NSX-V to NSX-T migration.
     You cannot change the Manager type setting after you create the cloud account.

7 For NSX mode, select either Policy or Manager.
   - Policy mode (default)
     The Policy mode is available for NSX-T 3.0 and NSX-T 3.1 forward. This option enables vRealize Automation Cloud to use the additional capabilities available in the NSX-T Policy API.
     If you are using NSX-T with a VMware Cloud on AWS cloud account in a cloud template, the NSX-T cloud account must use the Policy NSX mode.
     The Policy setting refers to the NSX-T Policy API form of NSX-T.
   - Manager mode
Existing NSX-T cloud accounts that were created in an earlier version of vRealize Automation Cloud are treated as Manager mode NSX-T cloud accounts.

The Manager mode is supported for NSX-T 2.4, NSX-T 3.0, and NSX-T 3.1 forward.

If you specify Manager mode, use the Manager mode option for other NSX-T cloud accounts until vRealize Automation Cloud introduces a Manager mode to Policy mode migration path.

Some vRealize Automation Cloud options for NSX-T require NSX-T 3.0 or greater, including adding tags to virtual machine NIC components in the cloud template.

The Manager setting refers to the NSX-T Manager API form of NSX-T.

If you have existing NSX-T cloud accounts that were created prior to the introduction of the Policy method in vRealize Automation Cloud August 2020, they use the Manager method. It is recommended that you wait until the Manager mode to Policy mode migration capability is made available in vRealize Automation Cloud. If you prefer not to wait, you should replace your existing NSX-T cloud accounts with new NSX-T cloud accounts that specify the Policy method.

You cannot change the NSX mode value after you create the cloud account.

8 Click Validate to confirm the credentials in relation to the selected NSX Manager type and NSX mode.

The assets associated with the account are collected.

If the NSX host IP address is not available, or if the cloud proxy is not associated with the NSX host IP address in the vCenter Server on which the cloud proxy is deployed, validation fails.

9 In Associations, add one or more vCenter cloud accounts to associate with this NSX-T cloud account. You can also remove existing vCenter cloud account associations.

Only vCenter cloud accounts that are not currently associated in vRealize Automation Cloud to an NSX-T or NSX-V cloud account are available for selection.

See What can I do with NSX-T mapping to multiple vCenters in vRealize Automation Cloud.

For information about making association changes after you have deployed a cloud template, or about deleting the cloud account after you have deployed a cloud template, see What happens if I remove an NSX cloud account association in vRealize Automation Cloud.

10 If you want to add tags to support a tagging strategy, enter capability tags.

You can add or remove capability tags later. See How do I use tags to manage Cloud Assembly resources and deployments.

For more information about how capability tags and constraint tags help control deployment placements, see the Constraint Tags and Placement video tutorial.

11 Click Save.
What to do next

You can create or edit a vCenter cloud account to associate with this NSX cloud account. See Create a vCenter cloud account in vRealize Automation Cloud.

Create and configure one or more cloud zones for use with the data centers that are used by this cloud account. See Learn more about Cloud Assembly cloud zones.

Configure infrastructure resources for this cloud account. See Chapter 4 Building your Cloud Assembly resource infrastructure.

For samples of using NSX-T options in vRealize Automation Cloud cloud templates, see Networks, security resources, and load balancers in vRealize Automation Cloud.

Create a VMware Cloud on AWS cloud account in vRealize Automation Cloud

As a cloud administrator, you can create a VMware Cloud on AWS cloud account for account regions to which your team will deploy vRealize Automation Cloud cloud templates.

VMware Cloud on AWS requires some unique configuration procedures in vRealize Automation Cloud. To properly configure vRealize Automation Cloud for VMware Cloud on AWS, including setting an API token values for the cloud account and setting gateway firewall rules for its cloud proxy, see the Tutorial: Configuring VMware Cloud on AWS for vRealize Automation Cloud workflow.

Prerequisites

- Verify that you have the required VMware Cloud on AWS administrator credentials, including VMware Cloud on AWS CloudAdmin credentials for the target SDDC in vCenter and that you have enabled HTTPS access on port 443. See Credentials required for working with cloud accounts in vRealize Automation Cloud.

- Verify that you have the cloud administrator user role. See What are the vRealize Automation Cloud user roles.

- Verify that you have installed a cloud proxy on the vCenter Server that is included in your VMware Cloud on AWS SDDC. See Create and deploy a cloud proxy for a VMware Cloud on AWS cloud account in vRealize Automation Cloud.

- Verify that the cloud proxy VM is on and that the cloud proxy service is connected and running. See Verify that a cloud proxy is running on a target virtual machine.

Procedure

1. Select Infrastructure > Connections > Cloud Accounts, click Add Cloud Account and select the VMware Cloud on AWS account type.

2. Add the VMC API token for your organization to access the available SDDCs.

   You can create a new token or use an existing token for your organization on the linked API Tokens page. For details, see Create a VMware Cloud on AWS cloud account in vRealize Automation Cloud within a sample workflow.
3 Select the SDDC to be available for deployments.
   NSX-V SDDCs are not supported and do not appear in the list.
   The vCenter and NSX-T Manager IP address/FQDN values are automatically populated based on the SDDC.
   If you have already installed a cloud proxy on the vCenter, the cloud proxy value is also specified. If you need to create a cloud proxy, see Create and deploy a cloud proxy for a VMware Cloud on AWS cloud account in vRealize Automation Cloud.

4 Enter your vCenter user name and password for the specified SDDC if other than the default value of cloudadmin@vmc.local.

5 Click Validate to confirm your access rights to the specified vCenter and check that the vCenter is running.
   The data centers associated with the account are collected.

6 For efficiency, create a cloud zone for provisioning to the selected SDDC.
   You can also create cloud zones as a separate step according to your organization's cloud strategy.

7 If you want to add tags to support a tagging strategy, enter capability tags.
   You can add or remove capability tags later. See How do I use tags to manage Cloud Assembly resources and deployments.

   For more information about how capability tags and constraint tags help control deployment placements, see the Constraint Tags and Placement video tutorial.

   As with VMs deployed to vSphere, you can configure machine tags for a VM to be deployed on VMware Cloud on AWS. You can also update the machine tag after initial deployment. These machine tags allow vRealize Automation Cloud to dynamically assign a VM to an appropriate NSX-T security group during deployment. For related information, see More about security group and tag resources in vRealize Automation Cloud cloud templates.

8 Click Save.

Results
The cloud account is added and the selected SDDC is available for the specified cloud zone.

What to do next
To properly configure vRealize Automation Cloud for VMware Cloud on AWS, see Tutorial: Configuring VMware Cloud on AWS for vRealize Automation Cloud.

For related information about VMware Cloud on AWS outside of vRealize Automation Cloud, see VMware Cloud on AWS documentation.
Integrating vRealize Automation Cloud with other applications

Integrations enable you to add external systems to vRealize Automation Cloud.

Integrations include vRealize Orchestrator, configuration management and other external systems such as GitHub, Ansible, Puppet, and external IPAM providers such as Infoblox.

Setting up cloud-based integrations

You configure a direct connection between Cloud Assembly and the target integration application.

How do I use Git integration in Cloud Assembly

Cloud Assembly supports integration with various flavors of Git repositories so that you can manage VMware cloud templates and action scripts under source control. This functionality facilitates auditing and accountability of processes around deployment.

Cloud Assembly supports different flavors of Git integration as described in the following list. Each of these options is a separate integration.

- GitHub cloud
- GitLab cloud

You must have an appropriate local Git repository configured with access for all designated users in order to set up Git integration with Cloud Assembly. Also, you must save your cloud templates in a specific structure in order for them to be detected by Git. To create an integration with GitLab or GitHub, select Infrastructure > Connections > Integrations in Cloud Assembly and then make the appropriate selection. You will need the url and token for the target repository.

When Git integration is configured with an existing repository, all cloud templates associated with selected projects become available to qualified users. You can use these templates with an existing deployment or as the basis of a new deployment. When you add a project, you must select some properties regarding where and how it is stored in Git.

You can save actions to a Git repository directly from Cloud Assembly. You can version action scripts either directly to Git, or you can create versions in Cloud Assembly. If you create a version of an action in Cloud Assembly, then it is automatically saved to Git as a version. Cloud templates are a bit more complicated, because you cannot directly add them to a Git integration from Cloud Assembly. You must save them directly to a Git instance, and then you can retrieve them from Git when working with the cloud template management page in Cloud Assembly.
Before you Begin

You must create and save your cloud templates in a specific structure in order for them to be detected by GitLab or GitHub.

- Configure and store cloud templates to be integrated with GitLab correctly. Only valid templates are imported into GitLab.
  - Create one or more designated folders for the cloud templates.
  - All cloud templates must be stored within `blueprint.yaml` files.
  - Ensure that the top of your templates include the `name:` and `version:` properties.
- Extract an API key for the applicable repository. In your Git account, select your login in the upper right corner, and navigate to the Settings menu. Select **Access Tokens**, then name your token, set an expiration date. Then, select API and create the token. Copy the resulting value and save it.

The following guidelines must be observed for all cloud templates used with Git integration.

- Each cloud template must reside in a separate folder.
- All cloud templates must be named `blueprint.yaml`.
- All cloud template YAML files must use `name` and `version` fields.
- Only valid cloud templates are imported.
- If you update a draft cloud template imported from Git, and its content differs from that in the top version, the draft will not be updated in subsequent syncs and a new version is created. If you want to update a template and also allow further sync's from Git, then you must create a new version after final changes.

**Configure GitLab cloud template integration in Cloud Assembly**

This procedure demonstrates configuring GitLab integration in Cloud Assembly so that you can work with cloud templates in the repository and automatically download saved templates that are associated with designated projects. To use cloud templates with GitLab, you must create a connection to an appropriate GitLab instance, and then save the desired templates to that instance.

**Configure GitHub integration in Cloud Assembly**

You can integrate the GitHub cloud-based repository hosting service in Cloud Assembly

**Configure GitLab cloud template integration in Cloud Assembly**

This procedure demonstrates configuring GitLab integration in Cloud Assembly so that you can work with cloud templates in the repository and automatically download saved templates that are associated with designated projects. To use cloud templates with GitLab, you must create a connection to an appropriate GitLab instance, and then save the desired templates to that instance.
When GitLab integration is configured with an existing repository, all cloud templates associated with selected projects become available to qualified users. You can use these templates with an existing deployment or as the basis of a new deployment. When you add a project, you must select some properties regarding where and how it is stored in GitLab.

**Note** You cannot push new or updated cloud templates to the Git repository from Cloud Assembly. Also, you cannot push new templates to the repository from Cloud Assembly. To add cloud templates to a repository, developers must use the Git interface.

If you update a draft cloud template imported from Git, and its content differs from that in the top version, the draft will not be updated in subsequent syncs and a new version is created. If you want to update a cloud template and also allow further sync's from Git, then you must create a new version after final changes.

After you set up your cloud templates for use with GitLab and collect required information, you must set up integration with your GitLab instance. Then, you can import the designated cloud templates into GitLab. You can view a video demonstration of this procedure at [https://www.youtube.com/watch?v=h0vqo63Sdgg](https://www.youtube.com/watch?v=h0vqo63Sdgg).

**Prerequisites**

- Extract an API key for the applicable repository. In your GitLab account, select your login in the upper right corner, and navigate to the Settings menu. Select Access Tokens, then name your token, set an expiration date. Then, select API and create the token. Copy the resulting value and save it.

You must have an appropriate local Git repository configured with access for all designated users in order to set up Git integration with Cloud Assembly. Also, you must create and save your cloud templates in a specific structure in order for them to be detected by GitLab.

- Configure and store cloud templates to be integrated with GitLab correctly. Only valid templates are imported into GitLab. See [How do I use Git integration in Cloud Assembly](#).

**Procedure**

1. Set up integration with your GitLab environment in Cloud Assembly.
   a. Select **Infrastructure > Integrations > Add New** and choose GitLab.
   b. Enter the **URL** for your GitLab instance. For a software as a service GitLab instance, in most cases, it will be gitlab.com.
   c. Enter the **Token**, also known as an API key, for the specified GitLab instance. See the prerequisites above for information about extracting the token from your GitLab instance.
   d. Add an appropriate **Name and Description**.
   e. Click **Validate** to verify the connection.
f  Add capability tags if desired. See Using capability tags in Cloud Assembly for more information.

g  Click Add.

2 Configure the GitLab connection to accept cloud templates in an appropriate repository.

a  Select Infrastructure > Integrations and choose the appropriate GitLab integration.

b  Select Projects.

c  Select New Project and create a name for the project.

d  Enter the Repository path within GitLab. Typically, this is the user name of the main account appended to the repository name.

e  Enter the appropriate GitLab Branch that you want to use.

f  If applicable, enter a Folder name. If left blank, all folders are available.

g  Enter an appropriate Type. If applicable, enter a folder name. If left blank, all folders are available.

h  Click Next to finish adding the repository.

   When you click Next, an automated synchronization task is initiated that imports cloud templates into the platform.

   When the synchronization tasks are complete, a message indicates that the cloud templates have been imported.

Results

You can now retrieve cloud templates from GitLab.

Configure GitHub integration in Cloud Assembly

You can integrate the GitHub cloud-based repository hosting service in Cloud Assembly.

You need a valid GitHub token to configure GitHub integration in Cloud Assembly. See the GitHub documentation for information about creating and locating your token.

Prerequisites

- You must have access to GitHub.
- Configure and store cloud templates to be integrated with GitHub correctly. Only valid cloud templates are imported into GitHub. See How do I use Git integration in Cloud Assembly.

Procedure

1  Select Infrastructure > Connections > Integrations and click Add Integration.

2  Select GitHub.

3  Enter the required information on the GitHub configuration page.

4  Click Validate to check the integration.
5 If you need to add tags to support a tagging strategy, enter capability tags. See How do I use tags to manage Cloud Assembly resources and deployments and Creating a tagging strategy.

6 Click Add.

7 Configure the GitLab connection to accept cloud templates in an appropriate repository.
   a Select Infrastructure > Integrations and choose the appropriate GitHub integration.
   b Select Projects.
   c Select New Project and create a name for the project.
   d Enter the Repository path within GitHub. Typically, this is the user name of the main account appended to the repository name.
   e Enter the appropriate GitHub Branch that you want to use.
   f If applicable, enter a Folder name. If left blank, all folders are available.
   g Enter an appropriate Type.
   h Click Next to finish adding the repository.

   An automated synchronization task is initiated that imports cloud templates into the platform.

   When the synchronization tasks are complete, a message indicates that the cloud templates have been imported.

Results

GitHub is available for use in Cloud Assembly blueprints.

What to do next

You can now retrieve cloud templates from GitHub.

Configure GitHub or GitLab integration for extensibility actions in Cloud Assembly

You can use Cloud Assembly to enable remote version control of extensibility actions.

By creating an integration with a remote GitHub or GitLab repository, you can manage the version history of your Cloud Assembly extensibility actions.

Also, you can use Git to manage Cloud Assembly cloud templates. For more information on using Git integration and Cloud Assembly cloud templates, see Configure GitLab cloud template integration in Cloud Assembly and Configure GitHub integration in Cloud Assembly.

Prerequisites

- An existing GitHub or GitLab repository.
- Extract an applicable access token for your GitHub or GitLab instance. For information on generating a GitHub access token, see the GitHub documentation. For information on generating a GitLab access token, see the GitLab documentation.

**Procedure**

1. Create a Git integration in Cloud Assembly.
   a. Navigate to **Infrastructure > Connections > Integrations > Add Integration**.
   b. Select **GitHub** or **GitLab**.
   c. Enter a name and description for the integration.
   d. Enter the server URL.
   e. Enter the access token for your Git instance.
   f. Click **Validate**.
   g. (Optional) Add capability tags. For more information see, Using capability tags in Cloud Assembly.
   h. Click **Add**.

2. Add your repository to the Git integration.
   a. Navigate to **Infrastructure > Connections > Integrations** and select the previously created Git integration.
   b. Select the **Projects** tab.
   c. Click **Add Project**.
   d. Select your extensibility project and click **Next**.
   e. Enter the path to your Git repository. Typically, the path is the user name of the main account appended to the repository name.
   f. Enter the branch of your Git repository that you want to use.
   g. (Optional) Enter an appropriate folder. If this text box is left blank, all folders are available.
   h. Under the **Type** drop-down menu, select **Action Based Scripts**.
   i. To finish adding the repository, click **Next**.

**Results**

Changes to the extensibility actions in your project are saved in the integrated Git repository.

**Note** Any changes to extensibility actions or actions scripts pushed directly to the integrated Git repository are picked up automatically by the extensibility action version history in Cloud Assembly.
Configure external IPAM integration in vRealize Automation Cloud using Amazon Web Services or Microsoft Azure

The procedure for defining an external IPAM integration point that is associated to a cloud extensibility proxy and one that is not associated to a cloud extensibility proxy are virtually identical.

To create an external IPAM integration point that uses an Amazon Web Services or a Microsoft Azure cloud account as its running environment, see How to configure an external IPAM integration in vRealize Automation Cloud.

Configure MyVMware integration in Cloud Assembly

You can integrate MyVMware with Cloud Assembly to support VMware related actions and capabilities, such as accessing the VMware Marketplace for cloud templates.

You can create only one My VMware integration for each organization.

Prerequisites

You must have a user account with the appropriate permissions for MyVMware.

- For information about inviting a user to a MyVMware account, see KB 2070555.
- For information about assigning user permissions in a My VMware account, see KB 2006977.

Procedure

1. Select Infrastructure > Connections > Integrations and click Add Integration.
2. Select My VMware.
3. Enter the required information on the MyVMware configuration page.
4. If you require tags to support a tagging strategy, enter capability tags. See How do I use tags to manage Cloud Assembly resources and deployments and Creating a tagging strategy.
5. Click Add.

Results

My VMware is available for use with cloud templates.

What to do next

Add a My VMware component to the desired cloud templates.

Request a SaltStack Config cloud integration

If you have an active vRealize Automation Cloud cloud license, you are eligible for a SaltStack Config cloud integration. You can request a SaltStack Config cloud integration using the VMware Cloud Services Console.

Note The SaltStack Config cloud deployment does not include SSH access, but you can use curl commands or deploy Salt states to make changes to your Salt master.
Requesting a SaltStack Config cloud integration

1. Log in to the VMware Cloud Services Console at https://cloud.vmware.com/.

2. Open a new Support Request. See How Do I Get Support for more detailed instructions on opening a support request.

3. In the support request, include the following information:
   - Your organization ID
   - Your contact email
   - IP addresses of the devices that need to access SaltStack Config
   - IP addresses of the public IP CIDR range for the SDDC (software-defined datacenter) where your Salt master is located

After submitting this support request, a VMware team deploys your SaltStack Config cloud instance within 24 hours. When the deployment is complete, you will receive a confirmation email containing the details for your new SaltStack Config cloud instance.

Requesting SaltStack Config configuration changes

Because your SaltStack Config cloud instance does not include SSH access, all configuration changes require another support request using the same process explained in the previous section.

In your support request, include:
   - Your organization ID
   - Any state files that need to be applied
   - A description of the requested changes

For faster service, indicate the path to the file you would like to change and list the changes in key-value pairs. For example, if you needed to change the RaaS configuration file to update the sseapi_command_age_limit and sseapi_ssl_validate_cert configuration settings, your request should be formatted similar to the following example:

```
File: /etc/salt/master.d/raas.conf

Values that need to be changed
sseapi_command_age_limit: 86400
sseapi_ssl_validate_cert: False
```

Troubleshooting cloud integrations

If your new deployment takes longer than 24 hours to complete or if some elements of your deployment is not working as expected, you can escalate the request to Global Support.

Create a SaltStack Config integration in vRealize Automation Cloud

You can create a SaltStack Config integration to access the SaltStack Config service and use SaltStack Config objects and actions in vRealize Automation Cloud.
With vRealize Automation SaltStack Config, you can provision, configure, and deploy software to your virtual machines at any scale using event-driven automation. You can also use SaltStack Config to define and enforce optimal, compliant software states across your entire environment.

**Considerations**

Integrated vRealize Automation SaltStack Config is available for vRealize Automation Cloud with the following conditions:

- The SaltStack Config integration is associated to a specific host during install.
- vRealize Automation Cloud does not support multi-tenancy for SaltStack Config currently.
- The vRealize Automation Cloud tenant can support one SaltStack Config integration and one Salt master. The Salt master can support multiple minions.
- Deleting a SaltStack Config integration in vRealize Automation Cloud does not check if there are existing deployments that use the SaltStack Config integration.

**Prerequisites**

- Verify that you have the SaltStack Config service administrator role in vRealize Automation Cloud. See What are the vRealize Automation Cloud user roles.
- Verify that you have the Cloud Assembly service administrator role in vRealize Automation Cloud. See Organization and service user roles in vRealize Automation Cloud.

**Configure a SaltStack Config integration in vRealize Automation Cloud**

Once you have installed SaltStack Config for vRealize Automation Cloud, you can open it from either the Infrastructure > Integrations page in Cloud Assembly or as a service from the vRealize Automation Cloud Service Console.

**Note** Unlike other vRealize Automation Cloud integration types, you cannot add a SaltStack Config integration by selecting Infrastructure > Connections > Integrations > Add Integration. The SaltStack Config integration is not available until it has been installed by using either of the two installation methods described above.

1. In Cloud Assembly, select Infrastructure > Connections > Integrations, select the available SaltStack Config integration tile, and click Open.
2  Complete the form.

a  Enter or edit the description.

The integration name is created when SaltStack Config is installed.

b  Enter the SaltStack Config administrator user name and password used to access the specified host.

The host name value is specified during SaltStack Config installation as the master address value. It cannot be changed after installation.

c  Click **Validate** to confirm your administrator access to the SaltStack Config integration host.

d  (Optional) Enter capability tags. For information about tagging, see **Using constraint tags in Cloud Assembly**.

e  Click **Save**.

After you save the SaltStack Config integration point, you can then open the SaltStack Config integration service itself from either the **Integrations** page or from the vRealize Automation Cloud Service Console.
3 Click the saved integration point to open the SaltStack Config integrated service and access the host.

4 When prompted to log in to SaltStack Config, enter your SaltStack Config administrator user name and password.

A sample SaltStack Config page is shown below.

---

**Setting up integrations that use a cloud proxy**

You configure a cloud proxy to communicate with some integrations, just as you do for cloud accounts. The cloud proxy is deployed on a target vCenter Server and manages the information between Cloud Assembly and the integration service or application.

For related information about cloud proxies, see Add a cloud proxy to a vCenter Server in Cloud Assembly. For details about how the cloud proxy works, see Understanding the VMware Cloud Services Cloud Proxy.
Configure a vRealize Orchestrator integration in Cloud Assembly

You can configure one or more vRealize Orchestrator integrations, so that you can use workflows as part of extensibility and cloud templates.

With the vRealize Orchestrator integration, you deploy a cloud extensibility proxy instance, that can be used for extensibility subscriptions and XaaS (Anything as a Service) operations used for cloud templates. This deployment includes a preconfigured vRealize Orchestrator 8.x instance that is created after you deploy and power on your cloud extensibility proxy. You cannot integrate external vRealize Orchestrator instances in vRealize Automation Cloud.

Important Integration of new SaaS-enabled vRealize Orchestrator 7.6 instances is no longer supported. Existing vRealize Orchestrator 7.6 SaaS integrations will continue to operate, but you cannot update the configuration of these integrations. To migrate these vRealize Orchestrator 7.6 SaaS integrations to your new vRealize Orchestrator 8.x integration, see Migrating a vRealize Orchestrator 7.6 SaaS instance to the cloud extensibility proxy.

The vRealize Orchestrator integration in vRealize Automation Cloud uses a Cloud Services Platform (CSP) authentication that provides the same features as external vRealize Orchestrator deployments that use a vRealize Automation license.

Note vRealize Orchestrator integration in vRealize Automation Cloud does not support changing the license provider for the integration.

For more information on license-based feature availability in vRealize Orchestrator, see vRealize Orchestrator Feature Enablement with Licenses in Installing and Configuring VMware vRealize Orchestrator.

Prerequisites

- Download and deploy a cloud extensibility proxy. See Download and deploy a cloud extensibility proxy.
- Verify that your organization has access to the Cloud Assembly service.
- Verify that you have cloud administrator credentials. See Credentials required for working with cloud accounts in vRealize Automation Cloud.
- Verify that the users you want to add to the vRealize Orchestrator integration have a Cloud Assembly Administrator or Cloud Assembly User service role. User roles in the vRealize Orchestrator integration are based on Cloud Assembly service roles. For a user to have administrator rights in the vRealize Orchestrator integration, they need the Cloud Assembly Administrator role. For a user to have workflow developer rights in the vRealize Orchestrator integration, they need the Cloud Assembly User role.

Procedure

1. Select Infrastructure > Connections > Integrations.
2. Click Add integration.
3 Select vRealize Orchestrator.

4 Enter a name for the vRealize Orchestrator integration.

5 (Optional) Enter a description for the vRealize Orchestrator integration.

6 Under **vRealize Orchestrator URL**, enter the fully qualified domain name (FQDN) of your cloud extensibility proxy.

   For example, https://my_extensibility_va_FQDN.com:443.

7 Under **Cloud extensibility proxy**, add your Cloud Assembly cloud extensibility proxy.

   Multiple vRealize Orchestrator integrations cannot share a cloud extensibility proxy.

8 (Optional) Enter your vRealize Automation Cloud refresh token.

   **Note** You can generate a new refresh token from the API Tokens tab of the My Account page. By entering a refresh token, you can access the vRealize Automation Cloud services from the client of the integrated vRealize Orchestrator deployment.

9 To validate the integration, click **Validate**.

10 (Optional) If prompted to do so, review the certificate information, and click **Accept**.

11 (Optional) Add capability tags. For more information on capability tags, see Using capability tags in Cloud Assembly.

   **Note** Capability tags can be used to manage multiple vRealize Orchestrator integrations. See Managing multiple vRealize Orchestrator integrations with project constraints.

12 Click **Add**.

   The vRealize Orchestrator integration is saved.

13 To verify that the integration is configured and that the workflows are added, select **Extensibility > Library > Workflows**.

**Results**

You have created a vRealize Orchestrator integration. After the integration, the vRealize Orchestrator service performs periodic subscription checks that confirm that your organization has access to the Cloud Assembly service. If your subscription to the Cloud Assembly service expires, you will be unable to access the vRealize Orchestrator integration.

**What to do next**

To create and edit vRealize Orchestrator content in your new integration, navigate to https://my_extensibility_va_FQDN.com:443.

**Migrating a vRealize Orchestrator 7.6 SaaS instance to the cloud extensibility proxy**

You can migrate your vRealize Orchestrator 7.6 SaaS instance to the cloud extensibility proxy so you can use the new vRealize Orchestrator 8.x functionality.
Cloud Assembly now supports the integration of vRealize Orchestrator 8.x instances through the cloud extensibility proxy. While you can continue using your existing vRealize Orchestrator 7.6 SaaS integration as it is, you cannot update its configuration. To overcome this limitation, you can migrate the vRealize Orchestrator 7.6 SaaS configuration to the cloud extensibility proxy.

The migration transfers the vRealize Orchestrator 7.6 SaaS configuration to your current vRealize Orchestrator 8.x instance, overwriting all existing elements such as workflows, actions, configuration and resource elements, including secure strings in workflows and configuration elements, packages, tasks, policies, certificates and trusted certificates, plug-ins, custom records in the js-io-rights.conf file, and Control Center system properties.

**Note** Following the migration, all currently running entities on the vRealize Orchestrator 7.6 SaaS instance, such as workflow runs, policy runs, scheduled tasks, and workflows waiting for user interaction, are in a failed state in the vRealize Orchestrator 8.x instance.

**Prerequisites**

Download and deploy a cloud extensibility proxy. See Download and deploy a cloud extensibility proxy.

**Procedure**

1. Log in to the command line of your cloud extensibility proxy over SSH as root.
2. To start the migration, run the vro-migrate command.
3. Follow the command prompts to provide the fully qualified domain name (FQDN) and credentials of the vRealize Orchestrator 7.6 SaaS instance.
4. (Optional) To follow the migration progress, access the migration log:
   a. Log in to the command line of your cloud extensibility proxy over a separate SSH session as root.
   b. Run the tail -f /var/log/vro-migration.log command.

**Results**

The migration process begins. You receive a notification on the cloud extensibility proxy when the migration finishes.

**Managing multiple vRealize Orchestrator integrations with project constraints**

You can use project constraints to manage what vRealize Orchestrator integrations are used in workflow subscriptions.

Cloud Assembly supports the integration of multiple vRealize Orchestrator servers that can be used in workflow subscriptions. You can manage what vRealize Orchestrator integrations are used in cloud templates provisioned by your project with soft or hard project constraints. For more information on project constraints, see Using Cloud Assembly project tags and custom properties.
Prerequisites

- Verify that you have cloud administrator credentials. See What are the vRealize Automation Cloud user roles.
- Configure two or more vRealize Orchestrator integrations in Cloud Assembly. See Configure a vRealize Orchestrator integration in Cloud Assembly.
- Add capability tags to your vRealize Orchestrator integrations. See Using capability tags in Cloud Assembly.

Procedure

1. Navigate to Infrastructure > Administration > Projects and select your project.
2. Select the Provisioning tab.
3. Enter the capability tags of your vRealize Orchestrator integrations in the Extensibility constraints text box and set them as soft or hard project constraints.
4. Click Save.

Results

When you deploy a cloud template, Cloud Assembly uses the project constraints to manage what vRealize Orchestrator integrations are used in workflow subscriptions.

What to do next

Alternatively, you can use capability tags to manage multiple vRealize Orchestrator integrations on a cloud account level. For more information, see Managing multiple vRealize Orchestrator integrations with cloud account capability tags.

Managing multiple vRealize Orchestrator integrations with cloud account capability tags

You can use capability tags to manage what vRealize Orchestrator integrations are used in workflow subscriptions.

Cloud Assembly supports the integration of multiple vRealize Orchestrator servers that can be used in workflow subscriptions. You can manage what vRealize Orchestrator integrations are used in workflow subscriptions by adding capability tags to your cloud account.

Prerequisites

- Verify that you have cloud administrator credentials. See What are the vRealize Automation Cloud user roles.
- Configure two or more vRealize Orchestrator integrations in Cloud Assembly. For more information, see Configure a vRealize Orchestrator integration in Cloud Assembly.
- Add capability tags to your vRealize Orchestrator integrations. See Using capability tags in Cloud Assembly.
Procedure

1. Navigate to **Infrastructure > Connections > Cloud Accounts**.
2. Select your cloud account.
3. Enter the capability tags of the vRealize Orchestrator integrations you want to use.
   The capability tags are automatically converted into soft constraints. To use hard constraints in managing your integrations, you must use project constraints. For more information, see *Managing multiple vRealize Orchestrator integrations with project constraints*.
4. Click **Save**.

Results

When you deploy a cloud template, Cloud Assembly uses the tagging in the associated cloud account to manage what vRealize Orchestrator integrations are used in workflow subscriptions.

Data collection for vRealize Orchestrator integrations

vRealize Automation Cloud performs periodic data collection for your vRealize Orchestrator integrations.

Data collection events for vRealize Orchestrator integrations are triggered every 10 minutes. The data collection gathers data about the workflows included in the library of each vRealize Orchestrator integration.

**Important**  
Verify that you version up a workflow when you are finished editing it. Changes to non-versioned up workflows are not picked up by the data collector.

You can find information about the last data collection performed on a vRealize Orchestrator integration by navigating **Infrastructure > Connections > Integrations** and selecting the specific integration. You can also trigger a manual data collection event by clicking **Start Data Collection**.

For more information on vRealize Automation Cloud data collection, see *How does data collection work in vRealize Automation Cloud*.

Configure an on-premises action-based extensibility integration in Cloud Assembly

You can configure your Cloud Assembly project to use an on-premises FaaS provider for your extensibility actions.

Using an on-premises FaaS provider for your extensibility actions provides several benefits:

- Enables the use of on-premises services such as LDAP, CMDB, and vCenter data centers as part of Cloud Assembly Extensibility.
- Enables the implementation of cross-cloud extensibility action flows. For example, you can create action flows that use Microsoft Azure or Amazon Web Services based extensibility actions alongside on-premises based extensibility actions.
The configuration of the on-premises integration requires the use of a cloud extensibility proxy. A non-cloud extensibility cloud proxy cannot be used when configuring the on-premises integration. For example, a cloud proxy created for the Cloud Assembly integration of services like Ansible or Puppet cannot be used for the on-premises extensibility action integration.

**Prerequisites**

Verify that you have cloud administrator credentials. See *What are the vRealize Automation Cloud user roles.*

**Procedure**

1. Configure the on-premises extensibility action integration.
   a. Select *Infrastructure > Connections > Integrations.*
   b. Click *Add Integration* and select *Extensibility Actions On Prem.*
   c. Create a cloud extensibility proxy or add an existing cloud extensibility proxy. See *Download and deploy a cloud extensibility proxy.*
   d. Click *Validate.*
   e. Enter a name for the integration.
   f. Add capability tags. See *Using capability tags in Cloud Assembly.*
   g. Click *Add.*

2. Add the capability tags of the on-premises integration to your project as project constraints.
   a. Select *Infrastructure > Administration > Projects.*
   b. Select your extensibility project and click the *Provisioning* tab.
   c. Enter the capability tags of your on-premises integration as soft or hard project constraints in the *Extensibility constraints* text box. See *Using Cloud Assembly project tags and custom properties.*
   d. Click *Save.*

**Results**

Extensibility subscriptions provisioned by your project can now use on-premises as the FaaS provider for your extensibility actions. See *Configure on-premises extensibility actions.*

**Download and deploy a cloud extensibility proxy**

Before you can create an on-premises extensibility action integration or vRealize Orchestrator 8.x integration for vRealize Automation Cloud, you must deploy a cloud extensibility proxy on your vCenter Server or VMware Cloud on AWS (VMC) library.

The cloud extensibility proxy is a virtual appliance (VA) used in the configuration of the on-premises extensibility action integrations and vRealize Orchestrator 8.x integrations in Cloud Assembly.
Prerequisites

- Verify that you have cloud administrator credentials. See Credentials required for working with cloud accounts in vRealize Automation Cloud.
- Verify that you have the cloud administrator user role. See What are the vRealize Automation Cloud user roles.

Procedure

1. Select **Infrastructure > Connections > Integrations**.
2. Click **Add Integration**.
3. Depending on the integration type, select **Extensibility Actions On Prem** or **vRealize Orchestrator**.

   The **New Integration** page opens.

4. Select **New Cloud Extensibility Proxy**.
5. Download the cloud extensibility proxy OVA or copy the provided link.

   **Note** Keep the **Install Cloud Extensibility Proxy** page in Cloud Assembly open. You must return to this page for a later step in the cloud extensibility proxy deployment.

6. Navigate to your vSphere Client data center, click the name of your vCenter Server cluster, and select **Deploy OVF Template**.

   **Note** You can also deploy the cloud extensibility proxy OVA on VMC. See Deploy a VM from an OVF Template in a Content Library in Managing the VMware Cloud™ on AWS Data Center.

7. Enter the deployment information as prompted.
8. Configure the network settings of the cloud extensibility proxy.
   a. For your network configuration, you must use a fully qualified domain name (FQDN) that can be resolved in reverse and forward DNS lookups.

      **Note** If the host name displayed in the shell of the deployed cloud extensibility proxy is *photon-machine*, the network is not configured correctly.

   b. (Optional) Enter the IP address/hostname, port, and credentials of your network proxy.

   c. Enter a display name for the remote data collector (RDC).

      **Note** The display name you enter for your RDC is used to identify the cloud extensibility proxy in the **Cloud Proxies** page of Cloud Assembly.

   d. Return to the **Install Cloud Extensibility Proxy** page in Cloud Assembly and copy the key provided in step 3.
Return to the vSphere Client and paste the key in the **VMware Cloud Services One Time Key (OTK)** text box.

Click **Next**.

Review the deployment information and click **Finish**.

After the deployment finishes, power on the cloud extensibility proxy. The cloud extensibility proxy automatically downloads and configures the components needed for your on-premises extensibility action integration or vRealize Orchestrator 8.x integration. This process can take up to 30 minutes to finish.

In **Cloud Assembly**, navigate to **Infrastructure > Connections > Cloud Proxies**.

Click the name of your cloud extensibility proxy.

Validate that the status of the **vro-agent** and **cloudassembly-abx-agent** service proxies is **RUNNING**.

The deployment of the cloud extensibility proxy can fail, because of an error in obtaining a valid IP address for the OpenFaaS gateway service. See **Troubleshooting the OpenFaaS service of the cloud extensibility proxy**.

For vRealize Orchestrator integrations, verify that the components are up and running.

Navigate to **https://my_extensibility_va_FQDN.com/vco-controlcenter**.

Log in as **root**.

Select **Validate Configuration**, and confirm that the vRealize Orchestrator components are up and running.

You can now use the deployed cloud extensibility proxy in configuring your on-premises extensibility action integration or vRealize Orchestrator 8.x integration.

**Troubleshooting the OpenFaaS service of the cloud extensibility proxy**

The deployment of the cloud extensibility proxy can fail because of a problem with the IP address configuration of the OpenFaaS gateway service.

The cloud extensibility proxy used for on-premises extensibility actions and vRealize Orchestrator 8.x integrations includes several infrastructure and application services that run in Kubernetes. Part of the cloud extensibility proxy deployment includes the assignment of IP addresses to these services. During deployment, one of the other Kubernetes services can obtain the IP address of the OpenFaaS gateway service. This results in the cloud extensibility proxy not deploying properly.
If the OpenFaaS IP configuration error occurs, you must delete the cloud extensibility proxy and redeploy it before you can begin creating and using on-premises extensibility actions or vRealize Orchestrator workflows.

**Note** You can view a detailed log of the cloud extensibility proxy deployment in the `deployExtensibility.log` file stored in the `/var/log` directory of the cloud extensibility proxy virtual appliance.

### Prerequisites

Download and deploy a cloud extensibility proxy in vCenter Server. See Download and deploy a cloud extensibility proxy.

### Procedure

1. **Verify if the OpenFaaS gateway service IP address is configured properly.**
   a. Log in to the cloud extensibility proxy command line over SSH as `root`. You can log in to the command line by using a SSH-enabled shell and entering the IP address of the cloud extensibility proxy.
   b. Verify if the OpenFaaS gateway service is deployed properly and has a valid IP address by running the `kubectl get services` command.

   ```bash
   kubectl get services --all-namespaces | grep openfaas
   ```

   If the OpenFaaS gateway service does not have an assigned IP address, you must redeploy the cloud extensibility proxy.

2. **Redeploy the cloud extensibility proxy.**
   a. Log in to your vSphere Client as an **administrator**.
   b. Select the host used to deploy the cloud extensibility proxy OVA.
   c. Power off the cloud extensibility proxy virtual machine and delete it from the disk.
   d. Redeploy the cloud extensibility proxy. See Download and deploy a cloud extensibility proxy.

   **Note** When redeploying the cloud extensibility proxy, you must generate a new authorization key in Cloud Assembly.

### Upgrading a cloud extensibility proxy

You can upgrade your deployed cloud extensibility proxy, so you can access the latest functionality for your vRealize Orchestrator and on-premises extensibility action integrations.

Upgrading the cloud extensibility proxy is only supported if you are using a proxy downloaded and deployed during the vRealize Automation Cloud December 2020 release or later.
Prerequisites

1. Download the upgrade ISO image by navigating to the MyVMware Patch Downloads page and selecting vRealize Automation Cloud Extensibility Proxy from the list of available product patches. For more information on accessing the MyVMware Patch Downloads page, see KB 80305.

2. Connect the CD-ROM drive of the cloud extensibility proxy virtual machine in vSphere. See the vSphere Virtual Machine Administration documentation.

   Note  After connecting the CD-ROM drive, navigate to your cloud extensibility proxy VM settings page and verify that Connect At Power On is enabled.

3. Mount the ISO image to the CD-ROM drive of the cloud extensibility proxy virtual machine in vSphere. See the vSphere Virtual Machine Administration documentation.

Procedure

1. Log in to the cloud extensibility proxy command line as root.

2. Run the blkid command, and note the device name for the cloud extensibility proxy CD-ROM drive.

3. Mount the CD-ROM drive.

   mount /dev/xxx /mnt/cdrom

4. Back up your cloud extensibility proxy by taking a virtual machine (VM) snapshot. See Take a Snapshot of a Virtual Machine.

   Caution  The cloud extensibility proxy does not currently support memory snapshots. Before taking the snapshot of your cloud extensibility proxy, verify that the Snapshot the virtual machine's memory option is disabled.

5. To initiate the upgrade, run the vracli upgrade exec -y --repo cdrom:// command.

   During the upgrade, you are automatically logged out of your terminal, because the cloud extensibility proxy reboots.

6. Log in to the cloud extensibility proxy command line as root and follow the upgrade progress by running the vracli upgrade status --follow command.

   Caution  The vracli upgrade status --follow command can occasionally display a false error message that indicates that the upgrade has failed. To troubleshoot this problem, see step 7.
7 (Optional) If you receive an error message while running the \texttt{vracli upgrade status \--follow} command, follow these steps:

a Verify that you receive the following error message:

```
Running health check after upgrade for nodes and pods.
Health check after upgrade for nodes and pods failed.
... Upgrade terminated due to critical error. Follow the upgrade guide to recover the system. ...
```

b Navigate to the \texttt{/var/log/vmware/prelude/upgrade-report-latest} and confirm that you receive the following error:

```
Pod: vco-app-xxxx is not in Ready or Completed state. All pods must be in either of these states
```

c Run the \texttt{kubectl get pods -n prelude \-w \| grep -E 'vco|orchestration-ui'} command and verify that the status of all 3 vco-app pods and the orchestration-up-app pod is \textbf{RUNNING}.

\textbf{Note} It can take up to 5-10 minutes after receiving the error message for all pods to enter the \textbf{RUNNING} state.

```
orchestration-ui-app-xxxx 1/1 Running 0 5h42m
vco-app-xxxx 3/3 Running 0 5h47m
```

d Run the \texttt{curl -k https://<cloud extensibility proxy FQDN>/vco/api/healthstatus} command and verify that the health check is returning a \textbf{RUNNING} status.

```
{"state":"RUNNING","health-status":{"state":"OK","time":1615296823325},"instance-id":"cloud extensibility proxy FQDN"}
```

\textbf{Important} The preceding command must run in an environment different from the cloud extensibility proxy command line. You can run the command from the command line of a different virtual machine. You can also view the health status information in a browser by navigating directly to \texttt{https://<cloud extensibility proxy FQDN>/vco/api/healthstatus}.

Results

You have successfully upgraded your cloud extensibility proxy to the latest version.

\textbf{How to configure an external IPAM integration in vRealize Automation Cloud}

You can create a provider-specific external IPAM integration point to manage the IP addresses used in your cloud template deployments. When using an external IPAM integration point, IP addresses are obtained from and managed by the designated IPAM provider rather than from vRealize Automation Cloud.
You can create a provider-specific IPAM integration point to manage IP addresses and DNS settings for cloud template deployments and VMs in vRealize Automation Cloud.

For information about how to configure the prerequisites, and an example of how to create a provider-specific external IPAM integration point within the context of a sample workflow, see Add an external IPAM integration for Infoblox in vRealize Automation Cloud. Note that this workflow is for an Infoblox IPAM integration but can be used as reference for any external IPAM vendor.

For information about how to create the needed assets to enable external IPAM partners and vendors to integrate their IPAM solution with vRealize Automation Cloud, see How do I use the IPAM SDK to create a provider-specific external IPAM integration package for vRealize Automation Cloud.

**Prerequisites**

- Verify that you have cloud administrator credentials. See Credentials required for working with cloud accounts in vRealize Automation Cloud.
- Verify that you have the cloud administrator user role. See What are the vRealize Automation Cloud user roles.
- Verify that you have an account with the external IPAM provider, for example Infoblox or Bluecat, and that you have the correct access credentials to your organization's account with the IPAM provider.
- Verify that you have access to a deployed integration package for the IPAM provider, such as Infoblox or BlueCat. The deployed package is initially obtained as a .zip download from your IPAM provider or from the vRealize Automation Cloud Marketplace and then deployed to vRealize Automation Cloud.
- Verify that you have access to a configured running environment for the IPAM provider.
- Verify that you have the required user credentials to access and use the IPAM vendor product. See your integration vendor's product documentation for information about required user permissions.

**Procedure**

1. Select **Infrastructure > Connections > Integrations** and click **Add Integration**.
2. Click **IPAM**.
3. In the **Provider** drop-down, select a configured IPAM provider package from the list. If the list is empty, click **Import Provider Package**, navigate to an existing provider package .zip file, and select it. If you do not have the .zip file, you can obtain it from your provider's web site or from the vRealize Automation Cloud **Marketplace** tab.
4. Enter your administrator user name and password credentials for your account with the external IPAM provider, along with all other (if any) mandatory fields, such as the host name of your provider.
5 In the **Running Environment** drop-down list, select an existing running environment, such as on-premises actions-based extensibility integration point.

The running environment supports communication between vRealize Automation Cloud and the IPAM provider.

The IPAM framework supports any of the following three types of running environments:
- an on-premises actions-based extensibility (ABX) integration point
- an Amazon Web Services (FaaS) cloud account
- a Microsoft Azure (FaaS) cloud account

**Note** If you use an Amazon Web Services or Microsoft Azure cloud account as the integration running environment, be sure that the IPAM provider appliance is accessible from the Internet and is not behind a NAT or firewall and that it has a publicly resolvable DNS name. If the IPAM provider is not accessible, the Amazon Web Services Lambda or Microsoft Azure Functions cannot connect to it and the integration will fail.

6 Click **Validate**.

7 When prompted to trust the self-signed certificate from the external IPAM provider, click **Accept**.

   After you accept the self-signed certificate, the validation action can continue to completion.

8 Enter a name for this IPAM integration point and click **Add** to save the new IPAM integration point.

   A data collection action is imitated. Networks and IP addresses are data-collected from the external IPAM provider.

### How to upgrade to a newer external IPAM integration package in vRealize Automation Cloud

You can upgrade an existing external IPAM integration point to source a more recent version of the vendor-specific IPAM integration package.

An external IPAM provider or VMware may upgrade a source IPAM integration package for a particular vendor. For example, the external IPAM integration package for Infoblox has been upgraded several times. To preserve any existing vRealize Automation Cloud infrastructure settings that use a named IPAM integration point, you can edit an IPAM integration point to source the updated IPAM integration package, rather than create a new IPAM integration point.

**Prerequisites**

This procedure assumes that you have already created an external IPAM integration point and want to upgrade that integration point to use a more recent version of the vendor's IPAM integration package.
For information about how to create an external IPAM integration point, see Add an external IPAM integration for Infoblox in vRealize Automation Cloud.

- Verify that you have cloud administrator credentials. See Credentials required for working with cloud accounts in vRealize Automation Cloud.
- Verify that you have the cloud administrator user role. See What are the vRealize Automation Cloud user roles.
- Verify that you have an account with the external IPAM provider and that you have the correct access credentials to your organization's account with that IPAM provider.
- Verify that you have access to a deployed integration package for your IPAM provider. The deployed package is initially obtained as a .zip download from your IPAM provider website or from the vRealize Automation Cloud Marketplace and then deployed to vRealize Automation Cloud.

For information about how to download and deploy the provider package .zip file and make it available as a Provider value on the IPAM Integration page, see Download and deploy an external IPAM provider package for use in vRealize Automation Cloud.

- Verify that you have access to a configured running environment for the IPAM provider. The running environment is typically a cloud account or integration point that contains an associated cloud extensibility endpoint that has been deployed to the target vCenter.

For information about running environment characteristics, see Create a running environment for an IPAM integration point in vRealize Automation Cloud.

Procedure

1. Select Infrastructure > Connections > Integrations IPAM and open the existing IPAM integration point.
2. Click Manage Providers.
3. Navigate to and import the updated IPAM integration package.
4. Click Validate and click Save.

How do I create an Active Directory integration in Cloud Assembly

Cloud Assembly supports integration with Active Directory servers to provide out of the box creation of computer accounts in a specified Organizational Unit (OU) within an Active Directory server prior to provisioning a virtual machine. Active Directory supports an LDAP connection to the Active Directory server.

An Active Directory policy that is associated with a project is applied to all virtual machines provisioned within the scope of that project. Users can specify one or more tags to selectively apply the policy to virtual machines that are provisioned to the cloud zones with matching capability tags.
For on-premises deployments, Active Directory integration enables you to set up a health check feature that shows the status of the integration and the underlying ABX integration on which it relies, including the required extensibility cloud proxy. Prior to applying an Active Directory policy, Cloud Assembly checks the status of the underlying integrations. If the integration is healthy, Cloud Assembly creates the deployed computer objects in the specified Active Directory. If the integration is unhealthy, the deploy operation skips the Active Directory phase during provisioning.

**Prerequisites**

- Active Directory integration requires an LDAP connection to the Active Directory server.
- If you are configuring an Active Directory integration with vCenter on-premises, you must configure an ABX integration with an extensibility cloud proxy. Select **Extensibility > Activity > Integrations** and choose **Extensibility Actions On Prem**.
- If you are configuring an integration with Active Directory in the cloud, you must have a Microsoft Azure or Amazon Web Services account.
- You must have a project configured with appropriate cloud zones, and image and flavor mappings to use with the Active Directory integration.
- If you are running an on-premises Active Directory, verify that you have installed and configured a cloud extensibility proxy. See [Download and deploy a cloud extensibility proxy](#).
- The desired OU on your Active Directory must be pre-created before you associated your Active Directory integration with a project.

**Procedure**

1. Select **Infrastructure > Connections > Integrations** and then **New Integration**.
2. Click **Active Directory**.
3. On the **Summary** tab, enter the appropriate LDAP host and environment names.
   
   If you are running an on-premises Active Directory, the running environment must be the name of your extensibility proxy. If you are running Active Directory on Microsoft Azure or Amazon Web Services in the cloud, enter the account name.
   
   The specified LDAP host is used to validate the Active Directory integration, and it is also used for subsequent deployments if no alternative hosts are specified and invoked due to errors or unavailability.
4. Enter the name and password for the LDAP server.
5. Enter the appropriate Base DN that specifies the root for the desired Active Directory resources.

   **Note** You can specify only one DN per Active Directory integration.
6. Click **Validate** to ensure that the integration is functional.
7. Enter a Name and Description of this integration.
8 Click **Save**.

9 Click the **Project** tab to add a project to the Active Directory integration.

   On the **Add Projects** dialog, you must select a project name and a relative DN, which is a DN that exists within the Base DN specified on the Summary tab.

10 Under the Extended Options selection, provide a comma separated list of **Alternate Hosts** that will be used if the initially selected server is unavailable during deployment. The primary server is always used for initial validation of the integration.

   **Note** If the format of the primary host is LDAP, LDAPS is not supported for alternative hosts.

11 Enter the time in seconds to wait for the initial server to respond before trying an alternate server in the **Connection Timeout** box.

12 Click **Save**.

**Results**

You can now associate the project with Active Directory integration to a cloud template. When a machine is provisioned using this cloud template, it is pre-staged in the specified Active Directory and Organizational Unit.

Initially, Active Directory integrations are deployed to a default OU with little user restrictions. The OU is set by default when you map an Active Directory integration to a project. You can add a property called **FinalRelativeDN** to blueprints to change the OU for Active Directory deployments. This property enables you to specify the OU to use with an Active Directory deployment.

```yaml
formatVersion: 1
inputs: {}
resources:
  Cloud_vSphere_Machine_1:
    type: Cloud.vSphere.Machine
    properties:
      image: CentOS8
      flavor: tiny
      activeDirectory:
        finalRelativeDN: ou=test
        securityGroup: TestSecurityGroup
```

As shown in the preceding YAML example, users can add a property to an Active Directory integration deployment that adds a computer account to the security group so that appropriate permissions are assigned to access the shared resource over a network. The Active Directory virtual machine is initially deployed to a fixed OU but when the machine is ready to release, it is moved to a different OU with the appropriate policy as applicable for users.

If a computer account is moved to a different OU after deployment, Cloud Assembly attempts to delete the accounts on the initial OU. Deletion of computer accounts succeeds only in the case of virtual machines moved to a different OU within the same domain.
You can also implement a tag-based health check for on-premises Active Directory integrations as follows.

1. Create an Active Directory integration as described in the preceding steps.
2. Click the **Project** tab to add a project to the Active Directory integration.
3. Select a project name and a relative DN on the Add Projects dialog. The relative DN must exist within the specified base DN.

   There are two switches on this dialog that enable you to control Active Directory configuration from cloud templates. Both switches are off by default.

   - **Override** - This switch enables you to override Active Directory properties, specifically the relative DN in cloud templates. When switched on, you can change the OU specified in the `relativeDN` property in the cloud template. When provisioned the machine will be added to the OU specified in the `relativeDN` property in the cloud template. The following example shows the cloud template hierarchy in which this property appears.

     ```
     activeDirectory:
     relativeDN: OU=ad_integration_machine_override
     ```

   - **Ignore** - This switch enables you to ignore the Active Directory configuration for the project. When switched on, it adds a property to the cloud template called `ignoreActiveDirectory` for the associated virtual machine. When this property is set to true means that the machine is not added to the Active Directory when deployed.

4. Add appropriate tags. These tags are applicable to the cloud zone to which the Active Directory policy may apply.
5. Click Save.

The Status of the Active Directory integration is displayed for each integration on the **Infrastructure > Connections > Integrations** page in Cloud Assembly.

You can associate the project with Active Directory integration with a cloud template. When a machine is provisioned using this template, it is pre-staged in the specified Active Directory and OU.

**How do I work with Kubernetes in Cloud Assembly**

Cloud Assembly offers several options for managing and deploying Kubernetes resources.

There are two primary options to working with Kubernetes resources in Cloud Assembly. You can integrate VMware Tanzu Kubernetes Grid Integrated Edition (TKGI), formerly PKS, or Red Hat OpenShift with Cloud Assembly to configure, manage and deploy Kubernetes resources. With the second option, you can leverage a vCenter cloud account to access supervisor namespaces to work with vSphere Project Pacific Kubernetes-based functionality. You can also integrate external Kubernetes resources in Cloud Assembly.
Working with VMware Tanzu Kubernetes Grid Integrated Edition (TKGI) or Openshift Integrations

For TKGI, external clusters, or OpenShift configurations, Cloud Assembly provides a Kubeconfig that enables users to access applicable Kubernetes clusters.

After you create a TKGI or OpenShift integration, applicable Kubernetes clusters become available in Cloud Assembly and you can add and create Kubernetes components to Cloud Assembly to support management of cluster and container applications. These applications form the basis of self-service deployments that are available from the Service Broker catalog.

Working with vSphere Project Pacific Kubernetes Clusters

Project Pacific is a vSphere enhancement that uses Kubernetes as its control plane. It enables you to manage both virtual machines and containers from one interface. Cloud Assembly enables users to leverage the Pacific Kubernetes capabilities that have been embedded within vSphere. You can access Pacific functionality by creating an integration with a vCenter deployment with a vSphere implementation that contains supervisor clusters. Pacific enables you to manage both conventional virtual machines and Kubernetes clusters from vCenter.

For Pacific-based supervisor namespaces, users must have access to an applicable vSphere SSO so that they can log in to a provided link to the supervisor namespace details. Then, they can download a customized Kubectl with vSphere authentication so they can use their supervisor namespace.

To use this functionality, you must have a vCenter with vSphere cloud account that has supervisor namespaces configured. After a user has logged in they can begin working with applicable namespaces.

- **Configure VMware Tanzu Kubernetes Grid Integrated Edition Integration in Cloud Assembly**
  You can configure a VMware Tanzu Kubernetes Grid Integrated Edition (TKGI), formerly PKS, resource connection on premises and in the cloud to support Kubernetes integration and management capabilities in Cloud Assembly.

- **Configure Red Hat OpenShift Integration in Cloud Assembly**
  You can configure a Red Hat OpenShift resource connection on premises and in the cloud to support enterprise-level Kubernetes integration and management capabilities in Cloud Assembly.

- **Configure a Kubernetes Zone in Cloud Assembly**
  Kubernetes zones enable cloud administrators to define policy-based placement of Kubernetes clusters and namespaces and supervisor namespaces used in Cloud Assembly deployments. An administrator can use this page to specify what clusters are available for provisioning of Kubernetes namespaces and what properties are acceptable for clusters.

- **Use Pacific supervisor clusters and namespaces with Cloud Assembly**
  Administrators can configure Cloud Assembly to use supervisor namespaces from an existing Pacific-enabled vSphere integration so that users can deploy namespaces in cloud templates and request them in the Service Broker catalog.
Add an external Kubernetes cluster to Cloud Assembly
You can add external Kubernetes clusters, not associated with a PKS instance, to Cloud Assembly for use in deployments.

Adding Kubernetes components to cloud templates in Cloud Assembly
When adding Kubernetes components to a Cloud Assembly cloud template, you can choose to add clusters or enable users to create namespaces in various configurations. Typically, this choice depends on your access control requirements, how you have configured your Kubernetes components, and your deployment requirements.

Using Cloud Assembly extensibility with Kubernetes
Cloud Assembly provides a set of event topics that correspond to typical actions related to Kubernetes cluster and namespace deployment. Users can subscribe to these topics as desired, and they will run at the appropriate time. Users receive notification when the event related to the subscribed topic occurs. You can also configure vRO workflows to run based on event notifications.

Working with Kubernetes clusters and namespaces in Cloud Assembly
Cloud administrators can add, view, and manage the configuration of deployed Kubernetes clusters and namespaces, both generic and Pacific-based, in Cloud Assembly.

Configure VMware Tanzu Kubernetes Grid Integrated Edition Integration in Cloud Assembly
You can configure a VMware Tanzu Kubernetes Grid Integrated Edition (TKGI), formerly PKS, resource connection on premises and in the cloud to support Kubernetes integration and management capabilities in Cloud Assembly.

TKGI integrations enable you to manage TKGI instances on premises and in the cloud and Kubernetes clusters provisioned on TKGI and external clusters. You must create a Kubernetes profile and associate it with a project to support policy-based placement of resources.

Prerequisites
- You must have an appropriately configured TKGI server set up with UAA authentication.
- You must have an existing cloud proxy or configure a new one. For details about how the cloud proxy works, see Understanding the VMware Cloud Services Cloud Proxy.
- Verify that you have cloud administrator credentials. For more information, see What are the vRealize Automation Cloud user roles.

Procedure
1. Select Infrastructure > Connections > Integrations and click Add Integration.
3. Enter the IP address or FQDN, and TKGI address for the TKGI cloud account you are creating.
   - The IP address is the FQDN or IP address of the TKGI user authentication server.
   - The TKGI address is the FQDN or IP address for the main TKGI server.
4 Select whether this TKGI server is local or located in the public cloud or on a private cloud.
   If the TKGI server is located on a private cloud, you must configure a cloud proxy to access it. Click **New Cloud Proxy**.
5 Enter an appropriate **Username** and **Password** for the TKGI server and other related information.
6 If you use tags to support a tagging strategy, enter capability tags. See *How do I use tags to manage Cloud Assembly resources and deployments* and *Creating a tagging strategy*.
7 Click **Add**.

**Results**

You can create Kubernetes zones and assign them to a project, or you can discover external Kubernetes clusters and assign those clusters to projects. In addition, you can add or create Kubernetes namespaces that facilitate management of clusters among large groups and organizations.

**What to do next**

Create or select the appropriate Kubernetes zones, then select one or more clusters or namespaces, and assign them to a project. After that, you can create and publish cloud templates to enable users to generate self-service deployments that use Kubernetes.

**Configure Red Hat OpenShift Integration in Cloud Assembly**

You can configure a Red Hat OpenShift resource connection on premises and in the cloud to support enterprise-level Kubernetes integration and management capabilities in Cloud Assembly.

Cloud Assembly supports integration with OpenShift versions 3.x.

**Prerequisites**

- You must have an appropriately configured Red Hat OpenShift implementation.
- You must have an existing cloud proxy or configure a new one if the target server is located on a private cloud. For details about how the cloud proxy works, see *Understanding the VMware Cloud Services Cloud Proxy*.
- Verify that you have cloud administrator credentials. For more information, see *What are the vRealize Automation Cloud user roles*.
- VMware supplies resources you can use to create an OpenShift cluster with a cloud template at the following location: [https://flings.vmware.com/enterprise-openshift-as-a-service-on-cloud-automation-services](https://flings.vmware.com/enterprise-openshift-as-a-service-on-cloud-automation-services). You can use clusters created with these resources as global clusters in the Kubernetes zones to create self-service namespaces.

**Procedure**

1 Select **Infrastructure > Connections > Integrations** and click **Add Integration**.
2 Select Red Hat OpenShift.
Enter the **Address** and **Location** for the OpenShift server.

If the OpenShift server is located on a private cloud, you must configure a cloud proxy to access it. Click **New Cloud Proxy**.

Select the appropriate **Credential Type** and enter the appropriate credentials.

OpenShift integration supports either OAuth username/password, public key, or bearer token authentication.

Enter an appropriate **Name** and **Description** for the OpenShift integration.

If you use tags to support a tagging strategy, enter the appropriate capability tags. See *How do I use tags to manage Cloud Assembly resources and deployments* and *Creating a tagging strategy*.

Click **Add**.

**Results**

When an integration is created, new Kubernetes clusters appear in the relevant section of the Kubernetes page. You can create Kubernetes zones and assign them to a project. In addition, you can configure Kubernetes namespaces that facilitate management of clusters among large groups and organizations.

**What to do next**

Create or select the appropriate Kubernetes zones, then select one or more clusters or namespaces, and assign them to a project. After that, you can create and publish cloud templates to enable users to generate self-service deployments that use Kubernetes.

**Configure a Kubernetes Zone in Cloud Assembly**

Kubernetes zones enable cloud administrators to define policy-based placement of Kubernetes clusters and namespaces and supervisor namespaces used in Cloud Assembly deployments. An administrator can use this page to specify what clusters are available for provisioning of Kubernetes namespaces and what properties are acceptable for clusters.

Cloud administrators can associate Kubernetes zones with TKGI cloud accounts configured for Cloud Assembly or with external Kubernetes clusters that are not associated with a project.

When you create a Kubernetes zone, you can assign multiple provider-specific resources to the zone, and these resources will dictate what properties can be set for the newly provisioned clusters in terms of the number of workers, masters, available CPU, memory, and other configuration settings. For TKGI providers, these correspond to TKGI plans. An administrator can also assign multiple clusters to a Kubernetes zone that will be used for placement of newly provisioned Kubernetes namespaces. The administrator can only assign clusters that are not onboarded, or not managed by CMX, and are provisioned via the preselected cluster provider. The administrator can assign multiple Kubernetes zones to a single project, thus making them all available for placement operations that happen within this project.
A cloud administrator can assign priorities on multiple levels.

- Kubernetes zone priority within a project.
- Resource priority within a Kubernetes zone.
- Cluster priority within a Kubernetes zone.

The cloud administrator can also assign tags on multiple levels:

- Capability tags per Kubernetes zone.
- Tags per resource assignment.
- Tags per cluster assignment.

You can create Kubernetes zones with supervisor namespaces on vSphere in the same way that you work with generic Kubernetes namespaces. To add a supervisor namespace to a Kubernetes zone, you must associate the zone with a vSphere 7 endpoint that contains the desired Pacific namespace resources.

Service Broker contains a version of the Kubernetes Zone page to enable Service Broker administrators to access existing Kubernetes zones so they can create placement policies for Kubernetes namespaces and clusters provisioned from the catalog.

**Prerequisites**


**Procedure**

1. Select **Infrastructure > Configure > Kubernetes Zone** and click **New Kubernetes Zone**.
2. Enter the TKGI integration **Account** name to which you want this zone to apply.
   - This defines the cloud account or endpoint that is associated with the zone. You can assign only one endpoint to each zone. If you are working with Supervisor Namespace on vSphere, you can only select vSphere instances here that contain Supervisor namespaces.
3. Add a **Name** and **Description** for the Kubernetes Zone.
4. Add capability tags if appropriate. See Using capability tags in Cloud Assembly for more information.
5. Click **Save**.
6. Click the On-demand tab and add TKGI plans as appropriate for the zone to use for cluster provisioning.
   - You can select one or more plans and assign priorities to them. Lower numbers equal higher priority. Priority assignments are secondary to tag based selection.
7 Click the Cluster tab and then click the **Add Compute** button to add Kubernetes or supervisor clusters to the zone. If you are working with an external cluster, it is automatically onboarded to Cloud Assembly when you select it.

You can add Kubernetes namespaces to the cluster on the Kubernetes Clusters page in Cloud Assembly.

**Results**

Kubernetes zones are configured for use with Cloud Assembly deployments.

**What to do next**

Assign the Kubernetes zone to a project.

1. Select **Infrastructure > Administration > Projects** and then select the project that you want to associate with your Kubernetes zone.

2. Click the Kubernetes Provisioning tab on the Project page.

3. Click **Add Kubernetes Zone** and add the zone that you just created. You can multiple zones if applicable, and you also set the priority on the zones.

4. Click **Save**.

The Kubernetes Provisioning tab of the Cloud Assembly Project page enables you to set limits on the type and number of namespaces users can provision to a Kubernetes zone. You can also select the type of namespaces that can be provisioned to a zone, either regular namespaces or supervisor namespaces. The Kubernetes Zones table on the Kubernetes Provisioning tab contains columns that show the current limit settings. To set limits, click the applicable zone on the table to open a dialog that enables you to choose namespace and supervisor namespace limits.

Click within the Supports column on the Kubernetes Zones table to select what type of namespace can be provisioned to the zone.

After you assign a Kubernetes zone to a project, you can use the Cloud Templates page under the Cloud Assembly Design tab to provision a deployment based on the Kubernetes zone and project configuration. This Cloud Templates page includes options to add a K8S Cluster, K8S Namespace and Supervisor Namespace. Select the appropriate option for the Kubernetes resource you are working with.

**Use Pacific supervisor clusters and namespaces with Cloud Assembly**

Administrators can configure Cloud Assembly to use supervisor namespaces from an existing Pacific-enabled vSphere integration so that users can deploy namespaces in cloud templates and request them in the Service Broker catalog.

This task describes how to add supervisor clusters with Cloud Assembly for use in deployments and how to create or add namespaces that define what Cloud Assembly projects and users can access particular Kubernetes resources. This functionality relies on a suitable vSphere cloud account rather than an integration such as VMware Tanzu Kubernetes Grid Integrated Edition.
(TKGI) or Openshift. Supervisor clusters are customized Kubernetes clusters associated with vSphere. They expose Kubernetes APIs to end users, and they use ESXi as a platform for worker nodes rather than Linux. Supervisor namespaces facilitate access control to Kubernetes resources, because it is typically easier to apply policies to namespaces than to individual virtual machines. You can create multiple namespaces for each supervisor cluster.

When used with Pacific enabled vSpehere instances, Kubernetes zones define which supervisor clusters are available for provisioning with a supervisor namespace. Supervisor namespaces are specific to Pacific enabled vSphere instances. You cannot provision a generic Kubernetes resource to a Pacific enabled vSphere instance.

Cloud Assembly users designated as project viewers have view only access to namespaces, while project members can edit them.

You can configure the supervisor clusters associated with namespaces if desired.

**Prerequisites**

- To use Pacific namespaces with Cloud Assembly, you must have a vSphere 7.x endpoint configured. vSphere is installed as part of a vCenter cloud account. See [Create a vCenter cloud account in vRealize Automation Cloud](#).
- Project Pacific must be enabled on the vSphere cloud account, and it must contain appropriate supervisor namespaces.
- Both your vCenter and your vRealize Automation Cloud deployment should use the same Active Directory for users to be synched. Though provisioning will still function if this is not the case, vRealize Automation Cloud users will not get automatic access to the namespace.

**Procedure**

1. Select **Infrastructure > Configure > Kubernetes Zone** in Cloud Assembly.
   
   This page shows managed clusters that are available for use, and enables you to add additional clusters. You can click on any of the clusters to view their details.

2. Select **New Kubernetes Zone**.

3. Specify the **Account** details for the target vSphere cloud account.

4. Click the Search icon in the text box to either view all vSphere accounts or search for an account by name.

5. Type a **Name** and **Description** for the new zone.

6. Add capability tags if appropriate. See [Using capability tags in Cloud Assembly](#) for more information.

7. Click the Provisioning tab to select the supervisor cluster that will be associated with the namespaces.

8. Click **Add Compute** to view and select the available supervisor clusters.

9. Click **Add**.
10 Select **Infrastructure > Administration > Projects** and then select the project that you want to associate with your Kubernetes zone.

11 Click the Kubernetes Provisioning tab on the Project page.

12 Click **Add Kubernetes Zone** and add the zone that you just created. You can multiple zones if applicable, and you also set the priority on the zones.

13 Click **Save**.

**What to do next**

After a namespace is configured, the **Infrastructure > Resources > Kubernetes** page in Cloud Assembly for applicable users displays the namespace. Users can click the Address link on the Summary tab to open the vSphere Kubernetes CLI Tools to manage the namespace. Users must be a cloud administrator or a member of the namespace for the designated project to access a link to the Supervisor namespace details. Also users can download a customized Kubectl to use the Supervisor namespace. Users can log in to the supervisor namespace and use it as they would any other namespace, and then create cloud templates and deploy applications.

To add the namespace to a cloud template select **Design > Cloud Template** and select an existing cloud template or create a new one. Then you can select the Supervisor namespace item on the left menu and drag it to the canvas.

You can assign storage policies to a supervisor namespace using tags. You can add tags, such as `location:local` to specify the kubernetes zone you want to use with the deployment and other tags on your storage profiles such as `speed:fast` and `speed:slow`.

```yaml
formatVersion: 1
resources:
  Cloud_SV.Namespace_1:
    type: Cloud.SV.Namespace
    properties:
      name: 'a'
      storage:
        - profile:
            constraints:
              - tag: 'speed:fast'
        - profile:
            limitMB: 1000
            constraints:
              - tag: 'speed:slow'
```

This cloud template requests a supervisor namespace with no constrains, and specifies two storage profiles with it.

After you deploy cloud templates containing a supervisor namespace, users can also request supervisor namespaces from the Service Broker catalog. Also, you can click on the Deployments page in Cloud Assembly to view information about the deployment and access a link that contains the command to run the kubectl for the namespace on vSphere.
Add an external Kubernetes cluster to Cloud Assembly

You can add external Kubernetes clusters, not associated with a PKS instance, to Cloud Assembly for use in deployments.

An external Kubernetes cluster is one that is not associated with a PKS instance.

As an alternative to the procedure herein, you can also add external clusters using the Cloud Assembly API. See Add an external Kubernetes cluster using the Cloud Assembly API.

The following procedure describes how to add an external cluster from the Cloud Assembly user interface.

Procedure

1. In Cloud Assembly, select **Infrastructure > Resources > Kubernetes** and click **Add External**.
   
   The Add External Cluster page appears.

2. Type a name and description for the cluster.

3. Select a Sharing configuration for the cluster. Global cluster are available for all cloud templates. If you choose Project, the cluster is available only for the project that you select.

4. Enter the address for the cluster and select whether it is located on a public or private cloud.

   If you select Private Cloud, you must configure a cloud proxy for the cluster.

What to do next

Add the cluster to a Kubernetes zone.

Add an external Kubernetes cluster using the Cloud Assembly API

You can add external Kubernetes clusters using the Cloud Assembly API as an alternative to the user interface.

API calls can be executed from a remote physical machine with connectivity to Cloud Assembly. Here is an example of an API call to add an external cluster.

```json
{
    "customProperties":
    {
        "dcId":
        "cbd683f6-3993-4e6d-a53c-a020bb45c08c"
    },
    "credentials":
    {
        "type":
        "Password",
        "userEmail":
        "kube-admin",
        "password":
        "secure_password"
    }
}
```
"privateKey": "test"
},
"name": "test-cluster",

"address": "https://10.1.39.36:6443",
"caCertificate": "...",

"clusterType": "EXTERNAL"
}

Note that docId is the ID of the data collector for accessing the on-prem resource. You can fetch all data collectors from the IaaS API at /iaas/api/data-collectors. The caCertificate property is the issuing certificate signed on the cluster's api certificate in PEM format.

After registering the cluster, you can list the namespaces and download the kube config file for them.

**Adding Kubernetes components to cloud templates in Cloud Assembly**

When adding Kubernetes components to a Cloud Assembly cloud template, you can choose to add clusters or enable users to create namespaces in various configurations. Typically, this choice depends on your access control requirements, how you have configured your Kubernetes components, and your deployment requirements.

To add a Kubernetes component to a cloud template in Cloud Assembly, select **Design > Cloud Templates**, click **New**, and then locate and expand the Kubernetes option on the left menu. Then, make the desired selection, either Cluster or KBS Namespace by dragging it to the canvas.

Adding a Kubernetes cluster that is associated with a project to a cloud template is the most straightforward method of making Kubernetes resources available to valid users. You can use tags on clusters to control where they are deployed just as you do with other Cloud Assembly resources. You can use tags to select a zone and a VMware Tanzu Kubernetes Grid Integrated Edition (TKGI) plan during the allocation phase of cluster deployment.

Once you add a cluster in this way, it is automatically available to all valid users.
Cloud template examples

The first cloud template example shows a template for a simple Kubernetes deployment that is controlled by tagging. A Kubernetes zone was created with two deployment plans, configured on the New Kubernetes Zone page. In this case, a tag called `placement:tag` was added as a capability on the zone, and it was used to match the analogous constraint on the cloud template. If there were more than one zone configured with the tag, the one with the lowest priority number would be selected.

```yaml
formatVersion: 1
inputs: {}
resources:
  Cluster_provisioned_from_tag:
    type: Cloud.K8S.Cluster
    properties:
      hostname: 109.129.209.125
      constraints:
      - tag: 'placement tag'
      port: 7003
      workers: 1
      connectBy: hostname
```

The second cloud template examples shows how to set up a template with a variable called `${input.hostname}` so that users can input the desired cluster hostname when requesting a deployment. Tags can also be used to select a zone and a TKGI plan during the resource allocation phase of cluster deployment.

```yaml
formatVersion: 1
inputs:
  hostname:
    type: string
    title: Cluster hostname
resources:
  Cloud_K8S_Cluster_1:
    type: Cloud.K8S.Cluster
    properties:
      hostname: ${input.hostname}
      port: 8443
      connectBy: hostname
      workers: 1
```

If you want to use namespaces to manage cluster usage, you can set up a variable in the cloud template called `name: ${input.name}` to substitute for the namespace name which a user enters when requesting a deployment. For this sort of deployment, you would create a template something like the following example:

```yaml
1 formatVersion: 1
2 inputs:
3  name:
4    type: string
5    title: "Namespace name"
6 resources:
```
Users can manage deployed clusters via kubeconfig files that are accessible from the **Infrastructure > Resources > Kubernetes Clusters** page. Locate the card on the page for the desired cluster and click **Kubeconfig**.

**Supervisor Namespaces in VMware Cloud Templates**

The following is the schema for a basic Supervisor namespace in a Cloud Assembly cloud template.

```json
{
  "title": "Supervisor namespace schema",
  "description": "Request schema for provisioning of Supervisor namespace resource",
  "type": "object",
  "properties": {
    "name": {
      "title": "Name",
      "description": "Alphabetic (a-z and 0-9) string with maximum length of 63 characters. The character '-' is allowed anywhere except the first or last position of the identifier.",
      "type": "string",
      "pattern": "^.*\$\{.*\}.*$|^((?!-)[a-z0-9-]{1,63}(?<!-))$",
      "ignoreOnUpdate": true
    },
    "description": {
      "title": "Description",
      "description": "An optional description of this Supervisor namespace.",
      "type": "string",
      "ignoreOnUpdate": true
    },
    "constraints": {
      "title": "Constraints",
      "description": "To target the correct resources, blueprint constraints are matched against infrastructure capability tags. Constraints must include the key name. Options include value, negative [!], and hard or soft requirement.",
      "type": "array",
      "recreateOnUpdate": true,
      "items": {
        "type": "object",
        "properties": {
          "tag": {
            "title": "Tag",
Examples:
```
!location:eu:hard
 location:us:soft
pci
```
          "type": "string",
          "recreateOnUpdate": true
        }
      }
    }
  },
  "limits": {
    "title": "Limits",
    "description": "Limit constraints for Supervisor namespace resources.",
    "type": "object",
    "properties": {
      "items": {
        "type": "object",
        "properties": {
          "key": {
            "title": "Key",
            "description": "A user-provided key for the namespace constraint.",
            "type": "string",
            "ignoreOnUpdate": true
          },
          "value": {
            "title": "Value",
            "description": "A user-provided value for the namespace constraint.",
            "type": "string",
            "ignoreOnUpdate": true
          }
        }
      }
    }
  }
}
```
"description": "Defines namespace resource limits such as pods, services, etc.",
"type": "array",
"recreateOnUpdate": false,
"items": {
  "type": "object",
  "properties": {
    "stateful_set_count": {
      "title": "stateful_set_count",
      "description": "This represents the new value for 'statefulSetCount' option which is the maximum number of StatefulSets in the namespace.",
      "type": "integer",
      "recreateOnUpdate": false
    },
    "deployment_count": {
      "title": "deployment_count",
      "description": "This represents the new value for 'deploymentCount' option which is the maximum number of deployments in the namespace.",
      "type": "integer",
      "recreateOnUpdate": false
    },
    "cpu_limit_default": {
      "title": "cpu_limit_default",
      "description": "This represents the new value for the default CPU limit (in Mhz) for containers in the pod. If specified, this limit should be at least 10 MHz.",
      "type": "integer",
      "recreateOnUpdate": false
    },
    "config_map_count": {
      "title": "config_map_count",
      "description": "This represents the new value for 'configMapCount' option which is the maximum number of ConfigMaps in the namespace.",
      "type": "integer",
      "recreateOnUpdate": false
    },
    "pod_count": {
      "title": "pod_count",
      "description": "This represents the new value for 'podCount' option which is the maximum number of pods in the namespace.",
      "type": "integer",
      "recreateOnUpdate": false
    },
    "job_count": {
      "title": "job_count",
      "description": "This represents the new value for 'jobCount' option which is the maximum number of jobs in the namespace.",
      "type": "integer",
      "recreateOnUpdate": false
    },
    "secret_count": {
      "title": "secret_count",
      "description": "This represents the new value for 'secretCount' option which is the maximum number of secrets in the namespace.",
      "type": "integer",
      "recreateOnUpdate": false
    }
  }
}
"cpu_limit": {
  "title": "cpu_limit",
  "description": "This represents the new value for 'limits.cpu' option which is equivalent to the maximum CPU limit (in MHz) across all pods in the namespace.",
  "type": "integer",
  "recreateOnUpdate": false
},
"cpu_request_default": {
  "title": "cpu_request_default",
  "description": "This represents the new value for the default CPU request (in MHz) for containers in the pod. If specified, this field should be at least 10 MHz.",
  "type": "integer",
  "recreateOnUpdate": false
},
"memory_limit_default": {
  "title": "memory_limit_default",
  "description": "This represents the new value for the default memory limit (in mebibytes) for containers in the pod.",
  "type": "integer",
  "recreateOnUpdate": false
},
"memory_limit": {
  "title": "memory_limit",
  "description": "This represents the new value for 'limits.memory' option which is equivalent to the maximum memory limit (in mebibytes) across all pods in the namespace.",
  "type": "integer",
  "recreateOnUpdate": false
},
"memory_request_default": {
  "title": "memory_request_default",
  "description": "This represents the new value for the default memory request (in mebibytes) for containers in the pod.",
  "type": "integer",
  "recreateOnUpdate": false
},
"service_count": {
  "title": "service_count",
  "description": "This represents the new value for 'serviceCount' option which is the maximum number of services in the namespace.",
  "type": "integer",
  "recreateOnUpdate": false
},
"replica_set_count": {
  "title": "replica_set_count",
  "description": "This represents the new value for 'replicaSetCount' option which is the maximum number of ReplicaSets in the namespace.",
  "type": "integer",
  "recreateOnUpdate": false
},
"replication_controller_count": {
  "title": "replication_controller_count",
  "description": "This represents the new value for 'replicationControllerCount' option which is the maximum number of ReplicationControllers in the namespace.",
  "type": "integer",
  "recreateOnUpdate": false
}
VMware cloud templates support the use of limits with supervisor namespaces. Limits enable you to control resource usage for CPUs and memory as well as the maximum number of pods allowed in the namespace by deployed machines.

```json
formatVersion: 1
inputs: {}
resources:
  Cloud_SV.Namespace_1:
    type: Cloud.SV.Namespace
    properties:
      name: '${env.deploymentName}'
      limits:
        - cpu_limit: 1000
        - cpu_request_default: 800
        - memory_limit: 2000
        - memory_limit_default: 1500
        - pod_count: 200
```
Using Cloud Assembly extensibility with Kubernetes

Cloud Assembly provides a set of event topics that correspond to typical actions related to Kubernetes cluster and namespace deployment. Users can subscribe to these topics as desired, and they will run at the appropriate time. Users receive notification when the event related to the subscribed topic occurs. You can also configure vRO workflows to run based on event notifications.

The following topics are available for subscription on the **Extensibility > Library > Event Topics** page in Cloud Assembly. To view these topics, search for Kubernetes in the Event Topics Search text box.

- Kubernetes cluster allocation
- Kubernetes cluster post provision
- Kubernetes cluster post removal
- Kubernetes cluster provision
- Kubernetes cluster removal
- Kubernetes namespace allocation
- Kubernetes namespace post provision
- Kubernetes namespace post removal
- Kubernetes namespace removal
- Kubernetes namespace allocation
- Kubernetes supervisor namespace allocation
- Kubernetes supervisor namespace post provision
- Kubernetes supervisor namespace post removal
- Kubernetes supervisor namespace removal
- Kubernetes supervisor namespace allocation

Click one of the topics to view the schema for that topic which shows all the information that is collected and transmitted. There are namespace topics for both Kubernetes namespaces and supervisor namespaces. You can use any of this schema information to set up various notifications and management and reporting tasks.

You can set up action scripts for CMX-related actions on the **Extensibility > Library > Actions** page. Action scripts can be used for various purposes: for example, to create a DNS record of Kubernetes cluster provisioning. If you are creating a DNS record, you can use the `masternodeips` field from the Kubernetes cluster post provision topic with a REST command in an Action script to create a DNS record.

The Subscriptions page defines the relationship between the event topics and action scripts. You can view and manage these components on the Subscriptions page in Cloud Assembly.
See the Cloud Assembly extensibility documentation at Extending and automating application life cycles with extensibility for more information.

**Working with Kubernetes clusters and namespaces in Cloud Assembly**

Cloud administrators can add, view, and manage the configuration of deployed Kubernetes clusters and namespaces, both generic and Pacific-based, in Cloud Assembly.

Users with cloud administrator privileges can view, add, and manage Kubernetes clusters and namespaces to which you are entitled access on the Infrastructure > Resources > Kubernetes page. This page contains tabs for Clusters, Namespaces, Supervisor Clusters and Supervisor Namespaces. You can select one of these tabs to view and manage the analogous resources. Most typically, this page facilitates management of deployed clusters and namespaces.

- **Cluster:** A cluster is a group of Kubernetes nodes distributed across one or more physical machines. This page shows provisioned and undeployed clusters that have been configured for use on your Cloud Assembly instance. You can click on a cluster to view information about its current status. When you deploy a cluster, it includes a link to a Kubconfig file that is accessible only for cloud administrators. This file grants full admin privileges over the cluster including a list of namespaces.

  Supervisor clusters are unique to vSphere instances and use ESXI as their worker nodes instead of Linux.

- **Namespaces:** Namespaces are virtual clusters that provide administrators with a way to segregate cluster resources. They facilitate management of resources among large groups of users and organizations. As a form of role-based access control, a cloud administrator can enable users to add namespaces to a project when they request a deployment and then later manage those namespaces from the Kubernetes Clusters page. When you deploy a namespace, it includes a link to a kubeconfig file that enables valid users, such as developers, to view and manage some aspects of that namespace.

  Supervisor clusters and supervisor namespaces exist only on vSphere instances and provide Kubernetes-like access to vSphere objects.

A cloud administrator can change the project associated with a Kubernetes namespace or cluster on this page so that the administrator can provision Kubernetes resources from cloud templates and Service Broker and then assign them to specific projects for consumption. The administrator can change the scope of a cluster to make it global or project specific. Global clusters appear in the Clusters tab for all Kubernetes zones and are available for selection and provisioning. If a cluster is global, it can be added to a Kubernetes zone and then used to provision namespaces from the catalog.

If you are configuring new or existing cluster, you must select whether to connect with a master IP address or a master hostname.

**Working with generic Kubernetes Clusters in Cloud Assembly**

You can add new, existing, or external clusters to Cloud Assembly using the options on this page.

1. Select Infrastructure > Resources > Kubernetes and confirm that the Clusters tab is active.
If there are any clusters currently configured for your Cloud Assembly instance, they appear on this page.

2 If you are adding a new or existing cluster, or deploying a cluster, select the appropriate option according to the following table.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deploy</td>
<td>Add new clusters to Cloud Assembly</td>
<td>You must specify the TKGI cloud account that to which this cluster will be deployed as well as the desired plan and the number of nodes.</td>
</tr>
<tr>
<td>Add Existing</td>
<td>Configure an existing cluster to work with your project.</td>
<td>You must specify the TKGI cloud account, the cluster to use, and the appropriate project for the targeted developer. Also, you need to specify the sharing scope. If you want to share globally, you must configure your Kubernetes zones and namespaces appropriately.</td>
</tr>
<tr>
<td>Add External</td>
<td>Add a vanilla Kubernetes cluster, that might not be associated with TKGI, to Cloud Assembly.</td>
<td>You must designate a project to which the cluster is associated, enter the IP address for the desired cluster and select a cloud proxy and certificate information required to connect to this cluster.</td>
</tr>
</tbody>
</table>

3 Click **Add** to make the cluster available within Cloud Assembly.

**Working with Kubernetes Namespaces in Cloud Assembly**

If you are a cloud administrator, namespaces help you group and manage Kubernetes cluster resources. If you are a user, namespaces are the area in Kubernetes clusters for your deployments. Administrators and users can access namespaces using the Namespaces tab located on the **Infrastructure > Resources > Kubernetes** page.

There are several ways to add Kubernetes namespaces to resources in Cloud Assembly. The following procedure outlines one typical method.

1 Select **Infrastructure > Resources > Kubernetes** and click the Namespaces tab.

2 To add a new namespace, click **New Namespace**. To add an existing namespace click **Add Namespace**.

3 Enter a **Name** and **Description** for the namespace.

   At this point you have added a namespace for use with Kubernetes resources, but it is not associated with anything in particular.

4 Specify the **Cluster** that you want to associate with this namespace.

5 Click **Create** to add the namespace to Cloud Assembly.
You can add custom properties on Kubernetes namespaces to support extensibility in several different ways. You add custom properties when you provision a namespace by creating a Cloud Assembly cloud template. When you specify a Kubernetes namespace in a cloud template you can add properties to the namespace. First, you can right click on the properties in the template to access the default properties that are part of the cloud template schema. As a second option, you can add user-defined properties in the properties section of the namespace in the cloud template.

After deployment, these custom properties appear on the Deployments page in Cloud Assembly for the applicable deployment.

Finally, you can also add custom properties to a namespace using actions configured on the Extensibility > Actions page in Cloud Assembly.

Working with Supervisor clusters and Supervisor namespaces
Cloud administrators can view and change the configuration of supervisor clusters and namespaces on the Kubernetes page in Cloud Assembly.

1. Select Infrastructure > Resources > Kubernetes in Cloud Assembly.
2. Select Add Supervisor Cluster.
3. Specify the Account details for the target vSphere cloud account.
4. Click the Search icon in the Supervisor cluster text box to either view all supervisor clusters or search for a cluster by name.
5. Select the desired cluster and click Add.
6. Select the Supervisor Namespaces tab and click the New Supervisor Namespace button to add a new namespace.
7. Select the Supervisor Namespaces tab and click the New Supervisor Namespace button to add a new namespace.
   a. If you are creating a new namespace, add a Name and Description.
   b. Select the appropriate cloud Account to associate with the namespace.
   c. Select the Supervisor cluster to associate with this namespace.
   d. Select the Project to associate with the namespace.
   e. Use the Available storage policies selection to add storage policies for use with the namespace.

You can add all available storage policies or select specific policies for use with the supervisor namespace. Also, you can optionally set a limit on the storage size available with each available storage policy.

f. Click Create.
8. Review the relevant details for the new namespace. You can change the storage policy configuration if needed.
Users and groups that currently have access to the namespace in vSphere are listed on the Users tab. If new users or groups are added to the project, click the Update Users button on this tab to update the list. The list is not updated automatically, so you must use the button to update.

**Note**  Synchronization of users makes sense only if Cloud Assembly and vCenter are configured with a common Active Directory/LDAP service.

After a cluster or namespace is configured, the Infrastructure > Resources > Kubernetes page in Cloud Assembly displays the clusters and namespaces available to the user. You can click an individual namespace or cluster to open a page that contains a number or tabs that show statistics and other information for the resource, and enables you to configure various options.

The Summary tab for clusters on the Kubernetes page enables administrators to view and, in some cases, update configuration of a cluster including changing the scope. The Sharing radio buttons enable you to select either Global (sharable via Kubernetes Zone) or Project (access limited to a single project). If you select Project, you must also specific the applicable project in the Project selection below.

**Note**  Changing the sharing configuration may affect the namespaces available on the cluster.

Users can click the Address link on the Summary tab to open the vSphere Kubernetes CLI Tools to manage the namespace. Users must be a cloud administrator or a member of the namespace for the designated project to access a link to the Supervisor namespace details. Also users can download a customized Kubectl to use the Supervisor namespace. Users can log in to the supervisor namespace and use it as they would any other namespace, and then create cloud templates and deploy applications.

**What Is configuration management in Cloud Assembly**

Cloud Assembly supports integration with Puppet Enterprise, Ansible Open Source, and Ansible Tower so that you can manage deployments for configuration and drift.

**Puppet Integration**

To integrate Puppet-based configuration management, you must have a valid instance of Puppet Enterprise installed on a public or private cloud with a vSphere workload. You must establish a connection between this external system and your Cloud Assembly instance. Then you can make Puppet configuration management available to Cloud Assembly by adding it to appropriate blueprints.

The Cloud Assembly blueprint service Puppet provider installs, configures, and runs the Puppet agent on a deployed compute resource. The Puppet provider supports both SSH and WinRM connections with the following prerequisites:

- **SSH connections:**
  - The user name must be either a super user or a user with sudo permissions to run commands with `NOPASSWD`. 
- Disable `requiretty` for the given user.
- cURL must be available on the deployment compute resource.
- WinRM connections:
  - PowerShell 2.0 must be available on the deployment compute resource.
  - Configure the Windows template as described in the vRealize Orchestrator documentation.

The DevOps administrator is responsible for managing the connections to a Puppet master and for applying Puppets roles, or configuration rules, to specific deployments. Following deployment, virtual machines configured to support configuration management are registered with the designated Puppet Master.

When virtual machines are deployed, users can add or delete a Puppet Master as an external system or update projects assigned to the Puppet Master. Finally, appropriate users can de-register deployed virtual machines from the Puppet Master when the machines are decommissioned.

**Ansible Open Source Integration**

When setting up an Ansible integration, install Ansible Open Source in accordance with the Ansible installation instructions. See the Ansible documentation for more information about installation.

Ansible enables host key checking by default. If a host is reinstalled with a different key in the `known_hosts` file, an error message appears. If a host is not listed in the `known_hosts` file, you must supply the key on start-up. You can disable host key checking with the following setting in the `/etc/ansible/ansible.cfg` or `~/.ansible.cfg` file:

```
[defaults]
host_key_checking = False
localhost_warning = False

[paramiko_connection]
record_host_keys = False

[ssh_connection]
#ssh_args = -C -o ControlMaster=auto -o ControlPersist=60s
ssh_args = -o UserKnownHostsFile=/dev/null
```

To avoid the host key checking errors, set `host_key_checking` and `record_host_keys` to `False` including adding an extra option `UserKnownHostsFile=/dev/null` set in `ssh_args`. In addition, if the inventory is empty initially, Ansible warns that the host list is empty. This causes the playbook syntax check to fail.

Ansible vault enables you to store sensitive information, such as passwords or keys, in encrypted files rather than as plain text. Vault is encrypted with a password. In Cloud Assembly, Ansible uses Vault to encrypt data such as ssh passwords for host machines. It assumes that the path to the Vault password has been set.
You can modify the `ansible.cfg` file to specify the location of the password file using the following format.

```
vault_password_file = /path to/file.txt
```

You can also set the `ANSIBLE_VAULT_PASSWORD_FILE` environment variable so that Ansible automatically searches for the password. For example,

```
ANSIBLE_VAULT_PASSWORD_FILE=~/.vault_pass.txt
```

Cloud Assembly manages the Ansible inventory file, so you must ensure that the Cloud Assembly user has `rwx` access on the inventory file.

```
cat ~/var/tmp/vmware/provider/user_defined_script/$(ls -t ~/var/tmp/vmware/provider/user_defined_script/ | head -1)/log.txt
```

If you want to use a non-root user with Cloud Assembly open-source integration, the users require a set of permissions to run the commands used by the Cloud Assembly open-source provider. The following commands must be set in the user's `sudoers` file.

```
Defaults:myuser !requiretty
myuser ALL=(ALL) NOPASSWD: ALL
```

If you encounter errors or other problems when setting up Ansible integration, refer to the `log.txt` file at `cat~/var/tmp/vmware/provider/user_defined_script/$(ls -t ~/var/tmp/vmware/provider/user_defined_script/ | head -1)` on the Ansible Control Machine.

**Ansible Tower Integration**

**Supported Operating System Types**

- Red Hat Enterprise Linux 8.0 or later 64-bit (x86), supports only Ansible Tower 3.5 and greater.
- Red Hat Enterprise Linux 7.4 or later 64-bit (x86).
- CentOS 7.4 or later 64-bit (x86).

The following is a sample inventory file, which is generated during an Ansible Tower installation. You may need to modify it for Cloud Assembly integration uses.

```
[root@cava-env8-dev-001359 ansible-tower-setup-bundle-3.5.2-1.el8]# pwd
/root/ansible-tower-install/ansible-tower-setup-bundle-3.5.2-1.el8

[root@cava-env8-dev-001359 ansible-tower-setup-bundle-3.5.2-1.el8]# cat inventory
[tower]
localhost ansible_connection=local

[database]

[all:vars]

admin_password='VMware1!'

pg_host=''

pg_port=''

pg_database='awx'

pg_username='awx'

pg_password='VMware1!'

rabbitmq_port=5672

rabbitmq_vhost=tower

rabbitmq_username=tower

rabbitmq_password='VMware1!'

rabbitmq_cookie=cookiemonster

# Needs to be true for fqdns and ip addresses

rabbitmq_use_long_name=false

# Isolated Tower nodes automatically generate an RSA key for authentication;
Configure Puppet Enterprise integration in Cloud Assembly

Cloud Assembly supports integration with Puppet Enterprise configuration management. When you add Puppet Enterprise to Cloud Assembly as an external system, by default it is available on all projects. You can restrict it to specific projects.

To add a Puppet Enterprise integration, you must have the Puppet master name and the hostname or IP address of the master.

You can find Puppet logs at the following location in case you need to check them for errors or information purposes.

<table>
<thead>
<tr>
<th>Description</th>
<th>Log Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>Log for create and install related events</td>
<td>Logs are on the deployed machine at `~/var/tmp/vmware/provider/user_defined_script/$(ls -t ~/var/tmp/vmware/provider/user_defined_script/</td>
</tr>
<tr>
<td>Log for Puppet delete and run related tasks</td>
<td>Logs are on the PE at `~/var/tmp/vmware/provider/user_defined_script/$(ls -t ~/var/tmp/vmware/provider/user_defined_script/</td>
</tr>
</tbody>
</table>

Procedure

1. Select **Infrastructure > Connections > Integrations** and click **Add Integration**.
2. Select Puppet.
3. Enter the required information on the Puppet configuration page.

   For Puppet integration to work properly, the provided credentials must be valid for both the SSH and the API account. Also, the specified OS and application user accounts must have the same username and password.

4. If you need a cloud proxy, click **NEW CLOUD PROXY** and enter the required information.

   Typically, a cloud proxy is required only if you are using vSphere. See Add a cloud proxy to a vCenter Server in Cloud Assembly.

5. Click **Validate** to check the integration.
6. Click **Add**.

Results

Puppet is available for use with cloud templates.
What to do next

Add Puppet components to the desired cloud templates.

1. Under Cloud Templates in Cloud Assembly, select Puppet under the Content Management heading on the cloud template menu and drag the Puppet component to the canvas.

2. Enter Puppet Properties on the pane to the right.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Master</td>
<td>Enter the name of the Puppet primary machine to be used with this cloud template.</td>
</tr>
<tr>
<td>Environment</td>
<td>Select the environment for the Puppet primary machine.</td>
</tr>
<tr>
<td>Role</td>
<td>Select the Puppet role to be used with this cloud template.</td>
</tr>
<tr>
<td>Agent Run Interval</td>
<td>The frequency at which you want the Puppet agent to poll the Puppet primary machine for configuration details to be applied to deployed virtual machines related to this cloud template.</td>
</tr>
</tbody>
</table>

3. Click the Code tab on the right pane to view the YAML code for the Puppet configuration properties.

When you add a Puppet component to a cloud template, you can add the `installMaster` property to the YAML file to point to a Puppet install master, also known as a compile master. The value of this property can be the IP address or the hostname of the Puppet compile master. Using this property provides access to enhanced capabilities for deployed Puppet virtual machines and also supports additional day two actions.

```yaml
Puppet_Agent:
  type: Cloud.Puppet
  properties:
    account: PEIntegrationAccount
    environment: production
    role: 'role::linux_webserver'
    host: '${CentOS-Puppet.*}'
    username: root
    password: password123!
    agentConfiguration:
      installMaster: my-pe-compile-master.example.com
      agentName: ${CentOS-Puppet.address}
      osType: linux
      count: 1
```

By default vRealize Automation Cloud passes some machine related information to Puppet virtual machines as facts. For Windows machines, this information is very limited. On Linux machines more information is passed by default, and users can pass additional information using custom properties.
There are some limitations on what is passed to Puppet machines under Linux. Custom properties on host resources and on the Puppet agent are passed to Puppet virtual machines. Custom properties on network resources are not passed to the virtual machine. Items passed include simple properties, boolean properties as well as custom named and complex types such as nested maps with arrays.

The following example shows how various custom resources can be called on host resources:

```yaml
resources:
  Puppet-Host:
    type: Cloud.AWS.EC2.Instance
    properties:
      customer_specified_property_on_ec2_resource: "property"

  customer_specified_property_on_network_resource_that_should_also_be_a_fact_and_is_boolean:
    true
      CustomerNameStuff: "zone A"
    try_map:
      key: value
      keytwo: value
    nested_array:
      - one
      - two
      - true
    try_array:
      - one
      - two
      - three:
        inner_key: value
```

If a Puppet purge command results in errors, in most cases, vRealize Automation Cloud will ignore purge errors for nodes and proceed with deletion of the node. Even if a certificate is not found for a specific node, vRealize Automation Cloud will proceed with deletion. If vRealize Automation Cloud cannot proceed with the node deletion for some reason, you can click Delete on the Deployments page Actions menu to open a dialog that will enable you to proceed with the node deletion. A similar workflow is executed when you remove a Puppet integration from a cloud template and then apply the template to the deployment. This workflow triggers a node purge operation that is handled as described above.

Integration with Puppet Enterprise requires a public IP address. If there is no public IP address configured for the Puppet Enterprise machine, the IP address of the first NIC is used.

If the NIC of a Puppet provisioned machine has multiple IP addresses, you can use the `primaryAddress` YAML property in cloud templates to specify which IP address to use for connections. When the `primaryAddress` property is assigned to a NIC, then the IP address of this NIC is used by Puppet. Only one NIC can be designated as primary. See the following YAML snippet for an example of how the `primaryAddress` property is used.

```yaml
BaseVM:
  type: Cloud.vSphere.Machine
  properties:
```
image: photon
count: 2
customizationSpec: Linux
cpuCount: 1
totalMemoryMB: 1024
networks:
  - network: '${resource.dev.id}'
    deviceIndex: 0
    primaryAddress: true
    assignment: static
  - network: '${resource.prod.id}'
    deviceIndex: 1
    assignment: static

If the `primaryAddress` property is not set for any virtual machine NIC, the cloud template logic will default to the current behavior for IP address selection.

**Configure Ansible Open Source integration in Cloud Assembly**

Cloud Assembly supports integration with Ansible Open Source configuration management. After configuring integration, you can add Ansible components to new or existing deployments.

When you integrate Ansible Open Source with Cloud Assembly, you can configure it to run one or more Ansible playbooks in a given order when a new machine is provisioned to automate configuration management. You specify the desired playbooks in the cloud template for a deployment.

When setting up an Ansible integration, you must specify the Ansible Open Source host machine as well as the inventory file path that defines information for managing resources. In addition, you must provide a name and password to access the Ansible Open Source instance. Later, when you add an Ansible component to a deployment, you can update the connection to use key-based authentication.

By default, Ansible uses ssh to connect to the physical machines. If you are using Windows machines as specified in the cloud template with the `osType` Windows property, the `connection_type` variable is automatically set to `winrm`.

Initially, Ansible integration uses the user/password or user/key credentials provided in the integration to connect to the Ansible Control Machine. Once the connection is successful, the provided playbooks in the cloud template are validated for syntax.

If the validation is successful, then an execution folder is created on the Ansible Control Machine at `~/.var/tmp/vmware/provider/user_defined_script/`. This is the location from which scripts run to add the host to the inventory, create the host vars files including setting up the authentication mode to connect to the host, and finally run the playbooks. At this point, the credentials provided in the cloud template are used to connect to the host from the Ansible Control Machine.
Ansible integration supports physical machines that do not use an IP address. For machines provisioned on public clouds such as AWS, Azure, and GCP, the address property in the created resource is populated with the machine's public IP address only when the machine is connected to a public network. For machines not connected to a public network, the Ansible integration looks for the IP address from the network attached to the machine. If there are multiple networks attached, Ansible integration looks for the network with the least deviceIndex; that is, the index of the Network Interface Card (NIC) attached to the machine. If the deviceIndex property is not specified in the blueprint, the integration uses the first network attached.

See What Is configuration management in Cloud Assembly for more details on configuring Ansible Open Source for integration in Cloud Assembly.

In addition, you must set up a cloud proxy if you are using vSphere.

Prerequisites

- The Ansible control machine must use an Ansible version. See the vRealize Automation Support Matrix for information about supported versions.
- Ansible log verbosity must be set to default of zero.
- If applicable, ensure that the cloudassembly-blueprint-agent version in cloud proxy is 20 or newer.
- The user must have read/write access to the directory where the Ansible inventory file is located. In addition, the user must have read/write access to the inventory file, if it exists already.
- If you are using a non-root user with the sudo option, ensure that the following is set in the sudoers file:
  
  Defaults:user_name !requiretty

  and

  username ALL=(ALL) NOPASSWD: ALL

- Ensure that host key checking is disabled by setting host_key_checking = False at /etc/ansible/ansible.cfg or ~/.ansible.cfg.

- Ensure that the vault password is set by adding the following line to the /etc/ansible/ansible.cfg or ~/.ansible.cfg file:

  vault password_file = /path/to/password_file
The vault password file contains the password in plain text and is used only when cloud templates or deployments provide the username and password combination to use between ACM and the node as show in the following example.

```
echo 'myStr0ng9@88w0rd' > ~/.ansible_vault_password.txt
echo 'ANSIBLE_VAULT_PASSWORD_FILE=~/.ansible_vault_password.txt' > ~/.profile
```

```
# Instead of this way, you can also set it setting
'vault_password_file=~/.ansible_vault_password.txt' in either /etc/ansible/ansible.cfg or ~/.ansible.cfg
```

To avoid host key failures while trying to run playbooks, it is recommended that you include the following settings in /etc/ansible/ansible_config.

```
[paramiko_connection]
record_host_keys = False

[ssh_connection]
#ssh_args = -C -o ControlMaster=auto -o ControlPersist=60s
ssh_args = -o UserKnownHostsFile=/dev/null                  # If you already have any
options set for ssh_args, just add the additional option shown here at the end.
```

**Procedure**

1. Select **Infrastructure > Connections > Integrations** and click **Add Integration**.
2. Click Ansible.
   
The Ansible configuration page appears.
3. Enter the Hostname, Inventory File Path and other required information for the Ansible Open Source instance.
4. If you need a cloud proxy, click **NEW CLOUD PROXY** and enter the required information. Typically, a cloud proxy is required only if you are using vSphere. See Add a cloud proxy to a vCenter Server in Cloud Assembly.
5. Click **Validate** to check the integration.
6. Click **Add**.

**Results**

Ansible is available for use with cloud templates.

**What to do next**

Add Ansible components to the desired cloud templates.

1. On the cloud template canvas page, select Ansible under the Configuration Management heading on the cloud template options menu and drag the Ansible component to the canvas.
2. Use the panel on the right to configure the appropriate Ansible properties such as specifying the playbooks to run.
In Ansible, users can assign a variable to a single host, and then use it later in playbooks. Ansible Open Source integration enables you to specify these host variable in cloud templates. The `hostVariables` property must be in proper YAML format, as expected by the Ansible control machine, and this content will be placed at the following location:

```
parent_directory_of_inventory_file/host_vars/host_ip_address/vra_user_host_vars.yml
```

The default location of the Ansible inventory file is defined in the Ansible account as added on the Integrations page in Cloud Assembly. The Ansible integration will not validate the `hostVariable` YAML syntax in the cloud template, but the Ansible Control Machine will throw an error when you run a playbook in the case of incorrect format or syntax.

The following cloud template YAML snippet shows an example usage of the `hostVariables` property.

```
Cloud_Ansible_1:
  type: Cloud.Ansible
  properties:
    host: '${resource.AnsibleLinuxVM.*}'
    osType: linux
    account: ansible-CAVA
    username: ${input.username}
    password: ${input.password}
    maxConnectionRetries: 20
    groups:
      - linux_vms
    playbooks:
      provision:
        - /root/ansible-playbooks/install_web_server.yml
    hostVariables: |
      message: Hello ${env.requestedBy}
      project: ${env.projectName}
```

Ansible integrations expect authentication credentials to be present in a cloud template in one of the following ways:

- User name and password in the Ansible resource.
- User name and privateKeyFile in the Ansible resource.
- Username in Ansible resource and privatekey in the compute resource by specifying `remoteAccess` to `generatedPublicPrivateKey`.

When you create an Ansible Open Source integration, you must provide login information for the integration user to connect with the Ansible control machine using SSH. To run playbooks with an integration, you can specify a different user in the integration YAML code. If no user is specified, vRealize Automation Cloud uses the integration user by default.

If you specify a different user, that user should have write access to the Ansible hosts file and should have permission to create private key files.
When you add an Ansible Open Source tile to a cloud template, vRealize Automation Cloud creates the host entry for the attached virtual machine. By default, vRealize Automation Cloud will use the virtual machine’s resource name to create the host entry, but you can specify any name using the `hostName` property in the blueprint YAML. In order to communicate with the machine, vRealize Automation Cloud will create the host variable `ansible_host: IP Address` for the host entry. You can override the default behaviour to configure communication using FQDN, by specifying the keyword `ansible_host` under `hostVariables` and providing FQDN as its value. The following YAML code snippet shows an example of how hostname and FQDN communication can be configured:

```yaml
Cloud_Ansible:
  type: Cloud Ansible
  properties:
    osType: linux
    username: ubuntu
    groups:
      - sample
    hostName: resource name
    host: name of host
    account: name of account
    hostVariables:
      ansible_host: Host FQDN
```

In this example you override the default `ansible_host` value by providing the FQDN. This may be useful for users who want Ansible Open Source to connect to the host machine using the FQDN.

The default value of `hostVariables` in the YAML will be `ansible_host:IP_address` and the IP address is used to communicate with the server.

If the YAML count property is more than 1 for Ansible Open Source, the hostname could be mapped to any of the respective virtual machine’s properties. The following example shows mapping for a virtual machine resource named Ubuntu-VM if we want its address property to be mapped to the hostname.

```
hostname: '${resource.Ubuntu-VM.address[count.index]}'
```

In cloud templates, ensure that the path to the Ansible playbook is accessible to the user specified in the integration account. You can use an absolute path to specify the playbook location, but it is not necessary. An absolute path to the user’s home folder is recommended so that the path remains valid even if the Ansible integration credentials change over time.

**Configure Ansible Tower Integration in Cloud Assembly**

You can integrate Ansible Tower with Cloud Assembly to support configuration management of deployed resources. After configuring integration, you can add Ansible Tower virtual components to new or existing deployments from the cloud template editor.

You must set up a cloud proxy if you are using vSphere.
Prerequisites

- Grant non-administrator users the appropriate permissions to access Ansible Tower. There are two options that work for most configurations. Choose the one that is most appropriate for your configuration.
  - Grant users Inventory Administrator and Job Template Administrator roles at the organization level.
  - Grant users Administrator permission for a particular inventory and the Execute role for all job templates used for provisioning.
- You must configure the appropriate credentials and templates in Ansible Tower for use with your deployments. Templates can be job templates or workflow templates. Job templates define the inventory and playbook for use with a deployment. There is a 1:1 mapping between a job template and a playbook. Playbooks use a YAML-like syntax to define tasks that are associated with the template. For most typical deployments, use machine credentials for authentication.
  - You can view the execution of the Job templates or workflow templates invoked from Cloud Assembly on the Ansible Tower Jobs tab.

Procedure

1. Select **Infrastructure > Connections > Integrations** and click **Add Integration**.
2 Click Ansible Tower.

The Ansible configuration page appears.

3 Enter the **Hostname**, which can be an IP address, and other required information for the Ansible Tower instance.

4 Enter the UI-based authentication **Username** and **Password** for the applicable Ansible Tower instance.

5 If you need a cloud proxy, click **NEW CLOUD PROXY** and enter the required information. Typically, a cloud proxy is required only if you are using vSphere. See [Add a cloud proxy to a vCenter Server in Cloud Assembly](#).

6 Click **Validate** to verify the integration.

7 Type an appropriate **Name** and **Description** for the integration.

8 Click **Add**.

**Results**

Ansible Tower is available for use in cloud templates.

**What to do next**

Add Ansible Tower components to the desired cloud templates. You must specify the applicable job template with execute permission for the user specified in the integration account.

1 On the cloud template canvas page, select Ansible under the Configuration Management heading on the blueprint options menu and drag the Ansible Tower component to the canvas.

2 Use the panel on the right to configure the appropriate Ansible Tower properties such as job templates.

When you add an Ansible Tower tile to a cloud template, vRealize Automation Cloud creates the host entry for the attached virtual machine in the Ansible Tower. By default, vRealize Automation Cloud will use the virtual machine’s resource name to create the host entry, but you can specify any name using the `hostName` property in the blueprint YAML. In order to communicate with the machine, vRealize Automation Cloud will create the host variable `ansible_host: IP Address` for the host entry. You can override the default behaviour to configure communication using FQDN, by specifying the keyword `ansible_host` under `hostVariables` and providing FQDN as its value. The following YAML code snippet shows an example of how hostname and FQDN communication can be configured:

```yaml
Cloud_Ansible_Tower_1:
  type: Cloud Ansible Tower
  properties:
    host: name of host
    account: name of account
```
In this example you override the default ansible_host value by providing the FQDN. This may be useful for users who want Ansible Tower to connect to the host machine using the FQDN.

The default value of hostVariables in the YAML will be ansible_host:IP_address and the IP address is used to communicate with the server.

If the YAML count property is more than 1 for Ansible Tower, the hostname could be mapped to any of the respective virtual machine's properties. The following example shows mapping for a virtual machine resource named Ubuntu-VM if we want its address property to be mapped to the hostname.

```yaml
hostname: '${resource.Ubuntu-VM.address[count.index]}'
```

When you add an Ansible Tower component to a cloud template, and you can specify the job template to call in the cloud template YAML. You can also specify workflow templates or a combination of job templates and workflow templates. If you don't specify the template type, by default vRealize Automation Cloud assumes that you are calling a job template.

The following YAML snippet shows an example of how a combination of job and workflow templates can be called in an Ansible Tower cloud template.

```yaml
Cloud_Ansible_1:
  type: Cloud.Ansible.Tower
  properties:
    host: '${resource.CentOS_Machine.*}'
    account:
      maxConnectionRetries: 2
      maxJobRetries: 2
  templates:
    provision:
      - name: My workflow
        type: workflow
      - name: My job template

We added the maxConnectionRetries and maxJobRetries to handle Ansible related failures. The cloud templates accepts the custom value and, in case no value is provided, it uses the default value. For maxConnectionRetries, the default value is 10, and for maxJobRetries the default value is 3.

**Note** Earlier versions of vRealize Automation Cloud supported the execution of job templates only using the jobTemplate schema in the cloud template. jobTemplate is now deprecated and might be removed in future releases. For now, using the jobTemplate property will continue to work as expected. To run workflow templates and use additional features, it is recommended to use the templates schema.
Cloud Assembly cloud templates for Ansible Tower integrations include the `useDefaultLimit` property with a true or false value to define where Ansible templates are executed. Ansible templates can be job templates or workflow templates. If this value is set to true, the specified templates are run against the machine specified in the Limit box on the Ansible Templates page. If the value is set to false, the templates are run against the provisioned machine, but users should check the Prompt on Launch checkbox on the Ansible Tower Templates page. By default, the value of this property is false. The following YAML example shows how the `useDefaultLimit` property appears in cloud templates.

```yaml
templates:
  provision:
    - name: ping aws_credentials
      type: job
      useDefaultLimit: false
      extraVars: '{"rubiconSurveyJob": "checkSurvey"}'
```

In addition, as the preceding example shows, you can use the `extraVars` property to specify extra variables or survey variables. This capability can be useful for running templates that require input. If a user has maintained the survey variable, then you must pass the variable in the `extraVars` section of the cloud template to avoid errors.

**Configure CloudHealth Integration in Cloud Assembly**

You can configure one or more CloudHealth integrations, so that you can provide and use public cloud costs for deployments and projects.

You can only create one CloudHealth integration per organization in vRealize Automation.

CloudHealth provides two user roles: Administrator and Power User, which also work with the integration to Cloud Assembly. The user that owns the API key used in creating the integration must belong to the Global organization in CloudHealth to have visibility to all public cloud accounts.

When a CloudHealth integration is created, vRealize Automation creates perspective groups for each public deployment hosted in vRealize Automation.

The perspective is synced every 12 hours to sync any new vRealize Automation deployments with CloudHealth. Once synced, CloudHealth generates the cost for all perspective groups and sends the data to vRealize Automation. For more information on costs for public cloud deployments, see [How to view pricing for AWS and Azure](#).

Prices are synced every 6 hours. You can also sync the perspective and prices on demand by clicking **Sync Perspective** and **Sync Prices** from the integration page.
Prerequisites

- You must have at least one perspective available for reporting in CloudHealth. If you plan to deploy many public cloud deployments, you might need to increase the perspective group limit. By default, CloudHealth has a limit of 200 perspective groups for each perspective. If you plan to have more deployments, contact CloudHealth to request a limit increase.

- The API key must have:
  - Perspective CRUD Privileges
  - AWS and Azure cost current and history report read privileges

Procedure

1. Select **Infrastructure > Connections > Integrations**.
2. Click **Add Integration**.
4. Enter a name and (optional) description.
5. Click **Display pricing information** to enable price syncing from CloudHealth to Cloud Assembly.
6. Enter the API key of the CloudHealth user and click **Validate**.
7. **(Optional)** Add capability tags. For more information on capability tags, see Using capability tags in vRealize Automation Cloud Assembly.
8. Click **Save**.

The CloudHealth integration is saved.

Results

The CloudHealth integration is created. A vRealize Automation perspective is created in CloudHealth.

**Note**  Do not attempt to modify or edit the vRealize Automation perspective in CloudHealth.

What to do next

Verify that the integration was created successfully by navigating to a public cloud deployment and clicking **Price**. If no price is shown, the perspective might not have synced yet. Navigate to the CloudHealth integration and click **Sync Perspective** and **Sync Prices**.

Integrating with vRealize Operations Cloud

vRealize Automation Cloud can work with vRealize Operations Cloud to perform advanced workload placement, provide deployment health and virtual machine metrics, and display pricing.
Number and type of integrations
Integration between the two products must be cloud to cloud, not a mix of cloud and on-premises.

vRealize Automation Cloud automatically discovers and adds an integration for a vRealize Operations Cloud instance when that vRealize Operations Cloud is linked to the same organization. In other words, integration between vRealize Automation Cloud and vRealize Operations Cloud is one-to-one per organization.

Basic requirements for integration
To integrate with vRealize Operations Cloud, make sure that vRealize Operations Cloud is one of the service cards to which you have access under My Services. In addition, vRealize Automation Cloud and vRealize Operations Cloud need to manage the same vSphere endpoint.

See the following sections for details. For pricing information, see How to use Pricing Cards.

Advanced workload placement using vRealize Operations Cloud
vRealize Automation Cloud and vRealize Operations Cloud can work together to optimally place deployment workloads.

You enable workload placement at the vSphere based cloud zone level. Only Distributed Resource Scheduler (DRS) enabled clusters of a cloud zone are eligible for advanced placement using vRealize Operations Cloud.

- vRealize Automation Cloud placement—The vRealize Automation Cloud placement engine is application intent based. It considers tag-based constraints, project membership and the associated cloud zones, and affinity filters related to network, storage, and compute. Resource placement depends on all of these factors plus the presence of other, related target resources in the same deployment.

- vRealize Operations Cloud placement—vRealize Operations Cloud considers operational intent for optimal placement. Operational intent can take past workloads and future, what-if predictions into account.

  When using advanced workload placement, you must apply vRealize Automation Cloud tagging in order to implement business intent decisions, instead of using the vRealize Operations Cloud business intent options.

When integrating with vRealize Operations Cloud, vRealize Automation Cloud continues to follow its application intent model and its related constraints to filter for target placement. Then, from within those results, it uses the vRealize Operations Cloud recommendation to further refine placement.

In the absence of a recommendation
If you enable advanced workload placement, and vRealize Operations Cloud analysis returns no recommendations, you may configure vRealize Automation Cloud to fall back to its default, application intent placement.
Limitations on workload placement

Certain limitations apply when using vRealize Operations Cloud to place workloads.

- vRealize Operations Cloud does not support workload placement on resource pools in vCenter Server.
- If vRealize Operations Cloud is down, the timeout used for workload placement to call vRealize Operations Cloud might expire.
- Placement doesn't cross multiple cloud zones. vRealize Automation Cloud sends one cloud zone to vRealize Operations Cloud for placement recommendations within that single cloud zone.

How to enable workload placement

To enable workload placement, there are steps to take for vSphere, vRealize Operations Cloud, and vRealize Automation Cloud.

1. Make sure that vRealize Operations Cloud is one of the service cards to which you have access under My Services.

   ![My Services](image)

   When the card is present, integration is automatic, but you can verify that it exists in Cloud Assembly under Infrastructure > Connections > Integrations.

   2. Make sure that vRealize Operations Cloud can access the same cloud account as Cloud Assembly.

      If necessary, import Cloud Assembly cloud accounts. In vRealize Operations Cloud, go to Administration > Solutions > Cloud Accounts, and click Import Accounts.

   3. In Cloud Assembly, connect to your vCenter Server cloud account.

      The options are under Infrastructure > Connections > Cloud Accounts.

   4. In vCenter Server, verify that DRS enabled clusters exist and are set to fully automated.

   5. In vRealize Operations Cloud, verify that the same vCenter Server is being managed.

   6. In Cloud Assembly, create a cloud zone for the vCenter Server account.

      The options are under Infrastructure > Configure > Cloud Zones.

   7. Under the cloud zone Summary tab, set the Placement Policy to ADVANCED.

   8. Under the Placement Policy, select whether to have vRealize Automation Cloud fall back to its default placement if vRealize Operations Cloud returns no recommendations.
Troubleshooting workload placement

If vRealize Operations Cloud isn’t recommending workload placements the way that you expect, review the deployment request details in Cloud Assembly or VMware Service Broker.

1. Go to **Infrastructure > Activity > Requests**, and click the request.
2. In Request Details, look at the allocation phases.
   - Look for targets that were successfully or unsuccessfully identified.
3. In Request Details, at the upper right, enable Dev Mode.
4. Follow the request path to locate filter blocks.
5. Click a filter block, and review the following section.

```
filterName: ComputePlacementPolicyAffinityHostFilter
  v computeLinksBefore
  v computeLinksAfter
  v filteredOutHostsReasons
```

<table>
<thead>
<tr>
<th>Entry</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>computeLinksBefore</td>
<td>List of potential placement hosts based on vRealize Automation Cloud algorithms.</td>
</tr>
<tr>
<td>computeLinksAfter</td>
<td>Selected placement host.</td>
</tr>
<tr>
<td>filteredOutHostsReasons</td>
<td>Messages describing why a host was selected or rejected.</td>
</tr>
<tr>
<td>advance policy filter: Filtered hosts based on recommendation from vROPS.</td>
<td></td>
</tr>
</tbody>
</table>

Deployment monitoring based on vRealize Operations Cloud

vRealize Automation Cloud can show vRealize Operations Cloud data about your deployments.

Reviewing the filtered set of metrics directly in vRealize Automation Cloud saves you the task of accessing or searching vRealize Operations Cloud. Although you cannot launch in context to vRealize Operations Cloud, you are of course free to log in and use vRealize Operations Cloud for additional data as needed.

Enabling vRealize Operations Cloud data

For vRealize Automation Cloud to work with vRealize Operations Cloud, you have an automatic vRealize Operations Cloud account endpoint integration.

The automatic vRealize Operations Cloud account endpoint

vRealize Automation Cloud automatically discovers and adds an integration for a vRealize Operations Cloud instance when that vRealize Operations Cloud is linked to the same organization.

For the automatic endpoint, a discovery process runs every 15 minutes and refreshes vRealize Automation Cloud with the following:

- Any new vRealize Operations Cloud being associated with the organization.
- Any updates to a current vRealize Operations Cloud associated with the organization.
- Any new vCenters or cloud accounts monitored by the linked vRealize Operations Cloud.
- Any removed vCenters or cloud accounts that were monitored by the linked vRealize Operations Cloud.

The automatic endpoint uses internal system client credentials to access vRealize Operations Cloud. You can't update the endpoint through user credentials, in either the user interface or by REST API.

vRealize Automation Cloud won't create an endpoint for an on-premises vRealize Operations associated with the organization. vRealize Automation Cloud only supports vRealize Operations Cloud.

**Health and alerts provided by vRealize Operations Cloud**

When monitoring is enabled, vRealize Automation Cloud retrieves vRealize Operations Cloud Health and associated alerts about your deployments.

To access monitoring, click a deployment and select the **Monitor** tab. If the tab is missing, see [Enabling vRealize Operations Cloud data](#).

To see alerts, highlight the deployment name at the top of the component tree in the left panel.

- You can review the severity and text of the alerts.
- To focus on areas of concern, filter and sort on data in the columns.
- Only Health badges and Health alerts appear. Other alert types such as Efficiency or Risk are not supported.

**Metrics provided by vRealize Operations Cloud**

When monitoring is enabled, vRealize Automation Cloud retrieves vRealize Operations Cloud metrics about your deployments.

To access monitoring, click a deployment and select the **Monitor** tab.

To see metrics, expand the component tree on the left, and highlight a virtual machine.

- Metrics are not cached. They come directly from vRealize Operations Cloud and might take a few moments to load.
- Only virtual machine metrics appear. Metrics from other components such as vCloud Director, Software, or XaaS are not supported.
- Only vSphere virtual machine metrics appear. Other cloud providers such as AWS or Azure are not supported.

Metrics appear as timeline graphs that show highs and lows for the following measures.

- CPU
- Memory
- Storage IOPS
- Network MBPS

To reveal the specific metric name, click the blue information icon at the upper left corner of the timeline.

**Acting on data provided by vRealize Operations Cloud**

When metrics provided by vRealize Operations Cloud expose a problem, you can identify trouble areas directly in vRealize Automation Cloud.

To see metrics provided by vRealize Operations Cloud, click a deployment and select the **Monitor** tab.

Metrics for the past day, week, or month are available. To zoom in on an area of concern, select a small area in the lower, shaded part under any metric timeline:

**Resource management and deployment optimization using vRealize Operations Cloud metrics in vRealize Automation Cloud**

In an integrated vRealize Automation Cloud and vRealize Operations Cloud environment, you can access insights and alerts for vRealize Automation Cloud objects that are monitored by vRealize Operations Cloud.
The **Insights** dashboard and **Alerts** tab pages provide the real-time capacity and related awareness information that you need to make management decisions in vRealize Automation Cloud without needing to open vRealize Operations Cloud. The information is supplied by the associated vRealize Operations Cloud application.

**Working with the insights dashboard and with resource alerts**

The **Insights** dashboard conveys information about capacity consumption across all computes within the cloud zone and grouped by projects. It can also show project deployments that are in need of optimization.

The **Alerts** pages displays potential capacity and performance concerns for objects such as cloud zones, projects, deployments, and virtual machines. They also contain information for project owners as to which of their deployments can be optimized. Each deployment link opens the **Optimize** tab in the deployment, where specific guidance is provided.

The following diagram illustrates the relationship between vRealize Automation Cloud resources and deployments, and the data that the associated vRealize Operations Cloud application provides in vRealize Automation Cloud.
Working with the insights dashboard

The Insights dashboard, which is available on each cloud zone page, provides the following vRealize Operations Cloud metrics:

- CPU, memory, and storage utilization usage as a percentage of capacity
- Capability consumption summary
- CPU and memory demand and usage history
- Consumption across projects
- Reclaimable resource capacity, with cost savings, for deployments and projects in a cloud zone

It also provides an option to alert project owners of deployments that can be optimized.

The Insights dashboard is available for vSphere and VMware Cloud on AWS cloud zones, provided that the cloud accounts are configured in both vRealize Automation Cloud and vRealize Operations Cloud and are being monitored in vRealize Operations Cloud.

For details, see: How to use the Insights dashboard to monitor resource capacity and notify project owners in vRealize Automation Cloud.

Working with alerts

The Alerts pages provide the following filtering categories. Filtering categories are supplied by the associated vRealize Operations Cloud application.

- Severity
- Status
- Impact
- Type
- Subtype
- Resource
Each filter can be further refined using quick filters. For example, the resource filter can be further refined by its quick filter types of cloud zone, virtual machine, deployment, and project resource.

You use combinations of filters and quick filters to control which alerts are available for display.

Some Alerts provide information about, and a link to, deployments that can be optimized. An individual alert can provide the option to contact the project owner, examine an Insights dashboard, or take possible actions.
Alerts are available for vSphere and VMware Cloud on AWS resource objects.

For details about how to configure and use integrated alerts, see How to use Alerts to manage resource capacity, performance, and availability in vRealize Automation Cloud and How to use Alerts to optimize deployments in vRealize Automation Cloud.

What are onboarding plans in Cloud Assembly

You use a workload onboarding plan to identify machines that have been data-collected from a cloud account type in a target region or data center but that are not yet managed by a Cloud Assembly project.

When you add a cloud account that contains machines that were deployed outside of Cloud Assembly, the machines are not managed by Cloud Assembly until you onboard them. Use an onboarding plan to bring unmanaged machines into Cloud Assembly management. You create a plan, populate it with machines, and then run the plan to import the machines. Using the onboarding plan, you can create a cloud template and can also create one or many deployments.

You can onboard one or many unmanaged machines in a single plan by selecting machines manually.
Machines that are available for workload onboarding are listed on the Resources > Machines page relative to a specific cloud account type and region and labeled as Discovered in the Origin column. Only machines that have been data-collected are listed. After you onboard the machines, they appear in the Origin column as Deployed.

The person who runs the workload onboarding plan is automatically assigned as the machine owner.

Onboarding also supports onboarding custom properties, attached disks, changing deployment owners, and vSphere networks.

- Custom properties - you can set custom properties at the plan and at the individual machine levels. A custom property set at the machine level overrides the same property on the plan level.

- Attached disks - If a machine has any non-bootable disks, they are automatically onboarded with the parent machine. To view non-bootable disks, click the machine name in the plan, and then navigate to the Storage tab.

- Deployment ownership - Onboarding allows you to change the default deployment owner. To change the owner, select a deployment from the Deployment tab, click Edit owner, and select the desired user associated with the project.

Onboarding examples

For examples of onboarding techniques, see Example: Onboard selected machines as a single deployment in Cloud Assembly.

Onboarding event subscriptions

A Deployment Onboarded event is created when you run the plan. Using Extensibility tab options, you can subscribe to these deployment events and perform actions on them.

After onboarding, you can update a project as a day 2 action for onboarded deployments. To use the change project action, the target project must use the same cloud zone resources as the deployment. You cannot run the change project action on any onboarded deployments where you made changes after onboarding.

Example: Onboard selected machines as a single deployment in Cloud Assembly

In this example, you onboard two unmanaged machines as a single Cloud Assembly deployment and create a single cloud template for all machines in the plan.

When you create a cloud account, all machines that are associated to it are data-collected and then displayed on the Infrastructure > Resources > Machines page. If the cloud account has machines that were deployed outside of Cloud Assembly, you can use an onboarding plan to allow Cloud Assembly to manage the machine deployments.

Note You can only rename deployments before they are onboarded. After onboarding, the Rename option is disabled.
Prerequisites

- Verify that you have the required user role. See What are the vRealize Automation Cloud user roles.
- Review What are onboarding plans in Cloud Assembly.
- Create and prepare a Cloud Assembly project.
  This procedure involves some of the steps from the basic Wordpress use case. See Tutorial: Setting up and testing multi-cloud infrastructure and deployments in Cloud Assembly.
  - Create a project, add users, and assign user roles in the project. See Part 2: Create the example Cloud Assembly project.
  - Create an Amazon Web Services cloud account for the project. See 1. Add cloud accounts.
    The Amazon Web Services cloud account in this procedure contains machines that were deployed before the cloud account was added to Cloud Assembly and by an application other than Cloud Assembly.
  - Verify that the Machines page contains machines to onboard. See Machine resources in vRealize Automation Cloud.

Procedure

1. Go to Infrastructure > Onboarding.
2. Click New Onboarding Plan and enter sample values.

<table>
<thead>
<tr>
<th>Setting</th>
<th>Sample Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Plan name</td>
<td>VC-sqa-deployments</td>
</tr>
<tr>
<td>Description</td>
<td>Sample onboarding plan for AWS machine for OurCo-AWS cloud account</td>
</tr>
<tr>
<td>Cloud account</td>
<td>OurCo-AWS</td>
</tr>
<tr>
<td>Default project</td>
<td>WordPress</td>
</tr>
</tbody>
</table>

3. Click Create.
4. On the plan's Deployments tab, click Select Machines, choose one or more machines, and click OK.
5 Select **Create one deployment that contains all the machines** and click **Create**.

6 Click the check box next to the new deployment name and click **Cloud template...**.

7 Click **Create cloud template in Cloud Assembly format**.

8 Enter a cloud template name and click **Save**.

---

**Note** When your onboarding plan uses a vSphere machine, you must edit the cloud template after the onboarding process is complete. The onboarding process cannot link the source vSphere machine and its machine template, and the resultant cloud template will **contain the imageRef: "no image available" entry in the cloud template code.** The cloud template cannot be deployed until you specify the correct template name in the **imageRef:** field. To make it easier to locate and update the cloud template after the onboarding process is complete, use the **Cloud template name** option on the deployment's **Cloud template configuration** page. Record the auto-generated cloud template name or enter and record a cloud template name of your choice. When onboarding is complete, locate and open the **cloud template** and replace the "no image available" entry in the **imageRef:** field with the correct template name.

9 Click the deployment name check box, click **Run**, and then click **Run** again on the **Run plan** page.

The selected Amazon Web Services machines are onboarded as a single deployment, with an accompanying cloud template.

10 Open and examine the cloud template by clicking the **Cloud templates** tab and then clicking the cloud template name.

11 Open and examine the deployment by clicking the **Deployments** tab and then clicking the deployment name.

---

**Advanced configuration for Cloud Assembly environment**

You can configure your Cloud Assembly environment to further support project configuration, integration, and deployment.

**What can I do with NSX-T mapping to multiple vCenters in vRealize Automation Cloud**

You can associate an NSX-T cloud account to one or more vCenter cloud accounts to support various deployment objectives.

You can associate the same existing NSX-T network to network profiles for different vCenters and provision a deployment in either vCenter based on constraints. Several examples are listed below:

- Cloud templates that contain a single machine with multiple NICs that use the same network profile, where that network profile contains an NSX-T network that spans multiple vCenters.
Cloud templates that contain a machine on a *private* network that uses a network profile with subnet-based isolation and that uses an NSX-T *existing* network that spans multiple vCenters.

Cloud templates that contain a single machine on a *private* network that uses a network profile with security group-based isolation and that uses an NSX-T network that spans vCenters.

Cloud templates that contain a single machine on a *routed* network that uses a network profile that contains an NSX-T network that spans multiple vCenters.

Cloud templates that contain an on-demand load balancer that is defined in a network profile where the load balancer is applied to all the vCenter machines on the network.

Cloud templates that contain an on-demand network that is defined in a network profile where the on-demand network is used by all the vCenters that use the network profile.

Cloud templates that contain an on-demand security group that optionally contains firewall rules and where the security group is associated to all the vCenters on the network.

You can configure vRealize Automation Cloud internal or external IPAM on the NSX-T network and share the same IP address for machines that are provisioned in different vCenters.

If no network profile is defined in your system, you can provision a cloud template that contains multiple machines on different vCenters that share a single *existing* NSX-T network.

**What happens if I remove an NSX cloud account association in vRealize Automation Cloud**

If you remove an association between an NSX cloud account and a vCenter cloud account, you also need to update the related network profiles to remove the associated NSX objects.

If you remove an association between an NSX cloud account and a vCenter cloud account, the infrastructure elements are not updated automatically by vRealize Automation Cloud. You must update your existing network profiles to remove the associated NSX objects.

The user interface provides information to help highlight the impacted network profile elements as follows:

- If the network profile has an NSX existing network selected:
  - The object is marked as *invalid* and the message *Some network objects are missing or invalid.* is displayed.
  - The objects are removed when you save the network profile.

- If the network profile has app isolation configured, you must update the Isolation policy settings before the network profile can be saved.

- If the network profile has security groups or load balancers selected, the objects are removed when you save the network profile.

Existing deployments continue to work as designed for existing components, but will fail when creating new components, for example in a scale-out operation.
If you re-establish the association, the network profile is repopulated and existing deployments work as designed.

If you remove the NSX cloud account, the above behavior is the same, but network objects are marked as _missing_ rather than _invalid_.

**How do I use the IPAM SDK to create a provider-specific external IPAM integration package for vRealize Automation Cloud**

External IPAM vendors and partners can download and use the IPAM SDK to create an IPAM integration package that enables vRealize Automation Cloud to support their provider-specific IPAM solution.

The process for building and deploying a custom IPAM integration package for vRealize Automation Cloud by using the supplied IPAM SDK is described in the _Creating and Deploying a Provider-specific IPAM Integration Package for VMware Cloud Assembly_ document. As described in the document, you can download the most recent _VMware vRealize Automation Third-Party IPAM SDK_ from the _VMware code_ site. The following IPAM SDK packages are available:

- VMware vRealize Automation Third-Party IPAM SDK 1.1.0
- VMware vRealize Automation Third-Party IPAM SDK 1.0.0

Before taking the time to create a vendor-specific IPAM integration package by using the IPAM SDK, check to see if one already exists for vRealize Automation Cloud. You can check for a provider-specific IPAM integration package on the IPAM provider’s website, in the _VMware Marketplace_ and from the vRealize Automation Cloud _Marketplace_ tab.

While the _Tutorial: Configuring a provider-specific external IPAM integration for vRealize Automation Cloud_ example is vendor-specific, it also contains helpful reference information.

**Using vRealize Automation Cloud with Azure VMware Solution**

This procedure describes how to set up vRealize Automation Cloud to work with a Microsoft Azure VMware Solution self-service hybrid cloud environment so that they can use vRealize Automation Cloud workloads within this environment.
vRealize Automation Cloud supports connections with Azure VMware Solution (AVS) to move and run VMware workloads on an Azure cloud environment. AVS was created by Microsoft to support interface with VMware environments.

Use of AVS is well documented by Microsoft. You can find the documentation here:

To use AVS in vRealize Automation Cloud, you must set up both vCenter and NSX-T cloud accounts. See the following documentation for setting up these cloud accounts:
- Set up vCenter cloud account - Create a vCenter cloud account in vRealize Automation Cloud
- Create an NSX-T cloud account - Create an NSX-T cloud account in vRealize Automation Cloud

**Note** You must deploy a cloud proxy on the AVS SDDC to support the vCenter and NSX-T cloud accounts. See Setting up cloud accounts in Cloud Assembly that require a cloud proxy for more information.

The following procedure outlines the high level steps to configure your environment so that you can deploy vRealize Automation Cloud workloads on AVS.

1. Install and configure Azure VMware Solution based on the vendor instructions as appropriate for your environment.
2. Create vCenter and NSX-T cloud accounts within your vRealize Automation Cloud deployment.

**Using vRealize Automation with Google Cloud VMware Engine**

This procedure describes how to set up vRealize Automation Cloud to work with a Google Cloud VMware Solution self-service hybrid cloud environment so that you can use vRealize Automation Cloud workloads within this environment.

vRealize Automation Cloud supports connections with Google Cloud VMware Engine (GCVE) to move and run VMware workloads on Google Cloud. GCVE was created by Google to support interface with VMware environments.

Use of GCVE is well documented by Google. You can find the documentation here:
- Google Cloud VMware Engine - https://cloud.google.com/vmware-engine/docs

To use GCVE with vRealize Automation Cloud, you must set up both vCenter and NSX-T cloud accounts in vRealize Automation Cloud. See the following documentation for setting up these cloud accounts:
- Set up vCenter cloud account - Create a vCenter cloud account in vRealize Automation Cloud
Create an NSX-T cloud account - Create an NSX-T cloud account in vRealize Automation Cloud

**Note** You must deploy a cloud proxy on the GCVE SDDC to support the vCenter and NSX-T cloud accounts. See Setting up cloud accounts in Cloud Assembly that require a cloud proxy for more information.

The following procedure outlines the high level steps to configure your environment so that you can deploy vRealize Automation Cloud workloads on GCVE.

1. Install and configure Google Cloud VMware Engine based on the vendor instructions as appropriate for your environment.
2. Create vCenter and NSX-T cloud accounts within your vRealize Automation Cloud deployment.

**Using vRealize Automation with Oracle Cloud VMware Solution**

This procedure describes how to set up vRealize Automation Cloud to work with an Oracle Cloud VMware Solution self-service hybrid cloud environment so that you can use vRealize Automation Cloud workloads within this environment.

vRealize Automation Cloud supports connection with Oracle Cloud VMware Solution (OCVS) to move and run VMware workloads on Oracle Cloud. OCVS was created by Oracle to support interface with VMware environments.

Use of OCVS is well documented by Oracle. You can find the documentation here:


To use OCVS, you must set up both vCenter and NSX-T cloud accounts. See the following documentation for setting up these cloud accounts:

- Set up vCenter cloud account - Create a vCenter cloud account in vRealize Automation Cloud
- Create an NSX-T cloud account - Create an NSX-T cloud account in vRealize Automation Cloud

**Note** You must deploy a cloud proxy on the OCVS SDDC to support the vCenter and NSX-T cloud accounts. See Setting up cloud accounts in Cloud Assembly that require a cloud proxy for more information.

The following procedure outlines the high level steps to configure your environment so that you can deploy vRealize Automation Cloud workloads on OCVS.

1. Install and configure Oracle Cloud VMware Solution based on the vendor instructions as appropriate for your environment.
2. Create vCenter and NSX-T cloud accounts within your vRealize Automation Cloud deployment.
Using vRealize Automation with VMware Cloud on Dell EMC

This procedure describes how to set up vRealize Automation Cloud to work with a VMware Cloud on Dell EMC self-service hybrid cloud environment so that you can use vRealize Automation Cloud workloads within this environment.

vRealize Automation Cloud supports connection with VMware Cloud on Dell EMC to move and run VMware workloads.

See the VMware Cloud on Dell EMC documentation at https://docs.vmware.com/en/VMware-Cloud-on-Dell-EMC/index.html for more information.

To use vRealize Automation Cloud with VMware Cloud on Dell EMC, you must set up a vCenter cloud account. See the following documentation for setting up this cloud account:

- Set up vCenter cloud account - Create a vCenter cloud account in vRealize Automation Cloud

**Note** You must deploy a cloud proxy on the VMware Cloud on Dell EMC SDDC to support the vCenter cloud account. See Setting up cloud accounts in Cloud Assembly that require a cloud proxy for more information.

The following procedure outlines the high level steps to configure your environment so that you can deploy vRealize Automation Cloud workloads on VMware Cloud on Dell EMC.

1. Install and configure VMware Cloud on Dell EMC based on the vendor instructions as appropriate for your environment.

2. Create a vCenter cloud account within your vRealize Automation Cloud deployment.
Building your Cloud Assembly resource infrastructure

Cloud Assembly resource infrastructure is where you define cloud account regions as zones into which cloud templates and their workloads can be deployed.

In addition, resource infrastructure involves creation of common mappings of images and machine sizes, and profiles that define network and storage capabilities across cloud account regions or data centers.

This chapter includes the following topics:

- How to add cloud zones that define Cloud Assembly target placement regions or data centers
- How to add flavor mappings in vRealize Automation Cloud to specify common machine sizings
- How to add image mapping in vRealize Automation Cloud to access common operating systems
- How to add network profiles in vRealize Automation Cloud
- How to add Cloud Assembly storage profiles that account for different requirements
- How to use Pricing Cards
- How do I use tags to manage Cloud Assembly resources and deployments
- How to work with resources in vRealize Automation Cloud
- Configuring Multi-provider tenant resources with vRealize Automation Cloud

How to add cloud zones that define Cloud Assembly target placement regions or data centers

A Cloud Assembly cloud zone is a set of resources within a cloud account type such as AWS or vSphere.

Cloud zones in a specific account region are where your cloud templates deploy workloads. Each cloud zone is associated with a Cloud Assembly project.

Select **Infrastructure > Configure > Cloud Zones** and click **Add New Zone**.
Learn more about Cloud Assembly cloud zones

Cloud Assembly cloud zones are sections of compute resources that are specific to your cloud account type such as AWS or vSphere.

Cloud zones are specific to a region, you must assign them to a project. There is a many to many relationship between cloud zones and projects. Cloud Assembly supports deployment to the most popular public clouds including Azure, AWS and GCP as well as to vSphere. See Adding cloud accounts to Cloud Assembly.

Additional placement controls include placement policy options, capability tags, and compute tags.

- Placement policy

  Placement policy drives host selection for deployments within the specified cloud zone.

  - default - Distributes compute resources across clusters and hosts randomly. This option works at an individual machine level. For example, all machines in a particular deployment are distributed randomly across the available clusters and hosts that satisfy the requirements.

  - binpack - Places compute resources on the most loaded host that has enough available resources to run the given compute.

  - spread - Provisions compute resources, at a deployment level, to the cluster or host with the least number of virtual machines. For vSphere, Distributed Resource Scheduler (DRS) distributes the virtual machines across the hosts. For example, all requested machines in a deployment are placed on the same cluster, but the next deployment may choose another vSphere cluster depending on the current load.

  For example, let's assume you have the following configuration:

  - DRS cluster 1 with 5 virtual machines
  - DRS cluster 2 with 9 virtual machines
  - DRS cluster 3 with 6 virtual machines

  If you request a cluster of 3 virtual machines and you select a Spread policy, they should all be placed on cluster 1. The updated loads become 8 virtual machines for cluster 1, while the loads for clusters 2 and 3 remain the same at 9 and 6.

  Then, if you request an additional 2 virtual machines, they are placed on DRS cluster 3, which will now have 8 virtual machines. The load for clusters 1 and 3 remain the same at 8 and 9.

  If two cloud zones both match all the criterias needed for provisioning, then the placement logic selects the one with higher priority.

- Capability tags
Blueprints contain constraint tags to help determine deployment placement. During deployment, blueprint constraint tags are mapped to matching capability tags in cloud zones and compute resources to determine which cloud zones are available for virtual machine resource placement.

- **Computes**

You can view and manage the compute resources that are available to provision workloads, such as AWS availability zones and vCenter clusters, to this cloud zone.

**Note** Cloud zones can no longer share compute resources. With the latest update of vRealize Automation Cloud, you cannot share compute resources across different cloud zones. Legacy cloud zones that use shared compute resources are still supported, but users are prompted to update them to conform with current standards.

Cloud zones that are auto-generated during cloud account creation are associated with the underlying compute resources after data collection.

If a vCenter compute cluster is DRS-enabled, the cloud zone only displays the cluster in the list of computes and it does not display the child hosts. If a vCenter compute cluster is not DRS-enabled, the cloud zone only displays standalone ESXi hosts, if present.

Add compute resources as appropriate for the cloud zone. The Compute tab contains a filter mechanism that enables you to control how compute resources are included with cloud zones. Initially, the filter selection is Include all Compute and the list below shows available compute resources, and they are all available for use in deployments. You have two additional options for adding compute resources to a cloud zone.

- **Manually select compute** - Select this option if you want to manually select compute resources from the list below. After you select them, click Add Compute to add the resources to the zone. The selected resources are available for use in deployments.

- **Dynamically include compute by tags** - Select this option if you want to include or exclude compute resources for the zone based on tags. All compute resources are shown until you add appropriate tags that match existing tags on compute resources. After you add one or more tags, compute resources with tags that match the filter are included in the zone and are available for use in deployments, while those that don't match are excluded.

For either compute option, you can remove one or more of the compute resources shown on the page by selecting the box to the right and clicking Remove.

Compute tags help to further control placement. You can use tags to filter available compute resources to only those that match one or more tags, as shown in the following examples.

- **Computes contain no tags and no filtering is used.**
Two computes contain the same tag but no filtering is used.

Two computes contain the same tag and the tag filter matches the tag used on the two computes.
Projects

You can view which projects have been configured to support workload provisioning to this cloud zone.

After you create a cloud zone, you can validate its configuration.

Insights dashboard

If you have an associated vRealize Operations Cloud application that you have configured to work with vRealize Automation Cloud, you can access an Insights dashboard in the cloud zone. The dashboard displays capacity-related information about resources and deployments for the vSphere or VMware Cloud on AWS cloud zone, provided that the cloud accounts are configured in both vRealize Automation Cloud and vRealize Operations Cloud and are being monitored in vRealize Operations Cloud. To learn more about the Insights dashboard, see Resource management and deployment optimization using vRealize Operations Cloud metrics in vRealize Automation Cloud.

How to add flavor mappings in vRealize Automation Cloud to specify common machine sizings

A vRealize Automation Cloud flavor map is where you use natural language to define target deployment sizes for a specific cloud account/region.

Flavor maps express the deployment sizes that make sense for your environment. One example might be small for 1 CPU and 2 GB memory and large for 2 CPUs and 8 GB memory for a vCenter account in a named data center and t2.nano for an Amazon Web Services account in a named region.

Select Infrastructure > Configure > Flavor Mappings and click New Flavor Mapping.

Learn more about flavor mappings in vRealize Automation Cloud

A flavor mapping groups a set of target deployment sizings for a specific cloud account/region in vRealize Automation Cloud using natural language naming.
Flavor mapping lets you create a named mapping that contains similar flavor sizings across your account regions. For example, a flavor map named `standard_small` might contain a similar flavor sizing (such as 1 CPU, 2 GB RAM) for some or all available account/regions in your project. When you build a cloud template, you pick an available flavor that fits your needs.

Organize flavor mappings for your project by deployment intent.

To simplify cloud template creation, you can select a pre-configuration option when you add a new cloud account. When you select the pre-configuration option, your organization's most popular flavor mapping and image mapping for the specified region are selected.

With regard to image mapping in cloud templates that contain vSphere resources, if there are no flavor mappings defined for a vSphere cloud zone, you can configure unlimited memory and CPU by using vSphere-specific settings in the cloud template. If there are flavor mappings defined for a vSphere cloud zone, the flavor mapping serves as a limit for vSphere-specific configurations in the cloud template.

How to add image mapping in vRealize Automation Cloud to access common operating systems

A vRealize Automation Cloud image map is where you use natural language to define target deployment operating systems for a specific cloud account/region.

Select `Infrastructure > Configure > Image Mappings` and click `New Image Mapping`.

Learn more about image mappings in vRealize Automation Cloud

An image mapping groups a set of predefined target operating system specifications for a specific cloud account/region in vRealize Automation Cloud by using natural language naming.

Cloud vendor accounts such as Microsoft Azure and Amazon Web Services use images to group a set of target deployment conditions together, including OS and related configuration settings. vCenter and NSX-based environments, including VMware Cloud on AWS, use a similar grouping mechanism to define a set of OS deployment conditions. When you build and eventually deploy and iterate a cloud template, you pick an available image that best fits your needs.

Organize image mappings for a project by similar operating system settings, tagging strategy, and functional deployment intent.

To simplify cloud template creation, you can select a pre-configuration option when you add a new cloud account. When you select the pre-configuration option, your organization's most popular flavor mapping and image mapping for the specified region are selected.

When you add image information to a cloud template, you use either the `image` or `imageRef` entry in the `properties` section of a machine component. For example, if you want to clone from a snapshot, use the `imageRef` property.

For examples of `image` and `imageRef` entries in cloud template code, see Chapter 6 Designing your Cloud Assembly deployments.
To assign a permission on a content library, an administrator must grant the permission to the user as a global permission. For related information, see Hierarchical Inheritance of Permissions for Content Libraries in vSphere Virtual Machine Administration at VMware vSphere Documentation.

**Synchronizing images for the cloud account/region**

You can run image synchronization to ensure that the images you are adding or removing for a given cloud account/region on the Infrastructure > Configure > Image Mapping page are current.

1. Open the associated Cloud Account/Region by selecting Infrastructure > Connections > Cloud accounts. Select the existing cloud account/region.
2. Click the Sync Images button and let the action complete.
3. When the action is complete, click Infrastructure > Configure > Image Mapping. Define a new or edit an existing image mapping and select the cloud account/region from step 1.
4. Click the image synchronization icon on the Image Mapping page.

**Viewing OVF details**

You can include OVF specifications in Cloud Assembly cloud template objects, such as vCenter machine components and image maps. If your image contains an OVF file, you can discover its content without opening the file. Hover over the OVF to display OVF details, including its name and location. For more information about the OVF file format, see vcenter ovf: property.
Using shared and latest images from a Microsoft Azure image gallery

When creating image mappings for Microsoft Azure, you can select images from a shared Azure image gallery in the subscription. The images in the drop-down menu are data-collected and made available based on your selected region.

While shared image galleries can be used across multiple subscriptions, they cannot be listed in the image mapping drop-down menu across subscriptions. Only the images of a particular subscription are data-collected and listed in the image mappings list. To use an image from an image gallery in a different subscription, provide the image ID in the image mapping and use that image mapping in the cloud template.

Using constraints and tags to refine image selection

To further refine image selection in a cloud template, you can add one or more constraints to specify tag-based restrictions on the type of image that can be deployed. The supplied Constraints example that is displayed when you are creating or editing an image mapping configuration is !license:none:hard. The example illustrates a tag-based restriction where the image can only be used if the license:none tag is not present in the cloud template. If you add tags such as license:88 and license:92, the specified image can be used only if the license:88 and the license:92 tags are present in the cloud template.

Using a cloud configuration script to control deployment

You can use a cloud configuration script in an image map, cloud template, or both to define custom OS characteristics to be used in a Cloud Assembly deployment. For example, based on whether you are deploying a cloud template to a public or private cloud, you can apply specific user permissions, OS permissions, or other conditions to the image. A cloud configuration script adheres to a cloud-init format for Linux-based images or a cloudbase-init format for Windows-based images. Cloud Assembly supports the cloud-init tool for Linux systems and the cloudbase-init tool for Windows.

For Windows machines, you can use any cloud configuration script format that is supported by cloudbase-init.

The machine resource in the following sample cloud template code uses an image that contains a cloud configuration script, the content of which is seen in the image entry.

```
resources:
demo-machine:
  type: Cloud.vSphere.Machine
  properties:
    flavor: small
    image: MyUbuntu16
    https://cloud-images.ubuntu.com/releases/16.04/release-20170307/ami-ubuntu-16.04-1.10.3-00-15269239.ova
```
cloudConfig: |
  ssh_pwalth: yes
  chpasswd:
    list: |
      ${input.username}:${input.password}
      expire: false
  users:
    - default
    - name: ${input.username}
      lock_passwd: false
      sudo: ['ALL=(ALL) NOPASSWD:ALL']
      groups: [wheel, sudo, admin]
      shell: '/bin/bash'
  runcmd:
    - echo "Defaults:${input.username}  !requiretty" >> /etc/sudoers.d/${input.username}

What happens when an image mapping and a cloud template contain a cloud configuration script

When a cloud template that contains a cloud configuration script uses an image mapping that contains a cloud configuration script, both scripts are combined. The merge action processes the contents of the image mapping script first and the contents of the cloud template script second, with consideration being given to whether the scripts are in `#cloud-config` format or not.

- For scripts that are in the `#cloud-config` format, the merge combines the contents of each module (for example `runcmd`, `users`, and `write_files`) as follows:
  - For modules where the contents are a list, the lists of commands from the image mapping and from the cloud template are merged, excluding commands that are identical in both lists.
  - For modules where the contents are a dictionary, the commands are merged and the result is a combination of both dictionaries. If the same key exists in both dictionaries, the key from the image mapping script dictionary is preserved and the key from the cloud template script dictionary is ignored.
  - For modules where the contents are a string, the content values from the image mapping script are kept and the content values from the cloud template script are ignored.
  - For scripts that are in a format other than `#cloud-config` or when one script is in `#cloud-config` format and the other is not, both scripts are combined in a way that the image mapping script is run first and the cloud template script is run when the image mapping script is finished.

For related information, see Merging user-data sections.
Add an image from a vCenter content library

When a local or publisher content library resides in a vCenter that is managed by your vRealize Automation Cloud organization, content library template images appear in the image drop-down menu. The images listed include OVF and VM template images in local or publisher vCenter content libraries. Images in subscriber content libraries do not appear in the drop-down menu. The template from which a VM has been cloned is shown in the machine details section of the machine deployments user interface.

Note If the publisher content library vCenter is managed by vRealize Automation Cloud, then publisher information is displayed in the image mapping selection grid in the following format: 
*publisher_content_library_name* / *content_item_name*

To assign a permission on a content library, an administrator must grant the permission to the user as a global permission. For related information, see Hierarchical Inheritance of Permissions for Content Libraries in *vSphere Virtual Machine Administration* at VMware vSphere Documentation.

If the publisher content library vCenter is not managed by vRealize Automation Cloud, then subscriber information is displayed in the image mapping selection grid in the following format: 
*subscriber_content_library_name* / *content_item_name*

When you deploy a cloud template that contains a VM template image mapping, vRealize Automation Cloud attempts to access the mapped image in the content library that is closest to the datastore, and then closest to the host, of the machine to be provisioned. This can include a local content library as well as a publisher or subscriber content library.

When you deploy a cloud template that contains an OVF template image mapping, OVF images are accessed as specified in the image mapping row if the image is in a local content library or a local subscriber of a specified remote publisher content library.

For related information about creating and using vCenter content libraries, see Using Content Libraries in vSphere product documentation and the How to Use Content Libraries in vRealize Automation 8 and vRealize Automation Cloud blog post.

More information about configuring and using cloud configuration scripts

For more information about working with cloud configuration scripts in cloud templates, see Machine initialization in Cloud Assembly .

Also see VMware blog articles vSphere Customization with Cloud-init While Using vRealize Automation 8 or Cloud and Customizing Cloud Assembly Deployments with Cloud-Init.
How to add network profiles in vRealize Automation Cloud

A vRealize Automation Cloud network profile describes the behavior of the network to be deployed.

For example, a network might need to be Internet-facing rather than internal-only.

Networks and network profiles are cloud-specific.

Select **Infrastructure > Configure > Network Profiles** and click **New Network Profile**.

Learn more about network profiles in vRealize Automation Cloud

A network profile defines a group of networks and network settings that are available for a cloud account in a particular region or data center in vRealize Automation Cloud.

You typically define network profiles to support a target deployment environment, for example a small test environment where an existing network has outbound access only or a large load-balanced production environment that needs a set of security policies. Think of a network profile as a collection of workload-specific network characteristics.

What's in a network profile

A network profile contains specific information for a named cloud account type and region in vRealize Automation Cloud, including the following settings:

- Named cloud account/region and optional capability tags for the network profile.
- Named existing networks and their settings.
- Network policies that define on-demand and other aspects of the network profile.
- Optional inclusion of existing load balancers.
- Optional inclusion of existing security groups.

You determine the network IP management functionality based on the network profile.

Network profile capability tags are matched with constraint tags in cloud templates to help control network selection. Further, all tags that are assigned to the networks that are collected by the network profile are also matched with tags in the cloud template to help control network selection when the cloud template is deployed.

Capability tags are optional. Capability tags are applied to all networks in the network profile, but only when the networks are used as part of that network profile. For network profiles that do not contain capability tags, tag matching occurs on the network tags only. The network and security settings that are defined in the matched network profile are applied when the cloud template is deployed.
When using static IP, the address range is managed by vRealize Automation Cloud. For DHCP, the IP start and end addresses are managed by the independent DHCP server, not by vRealize Automation Cloud. When using DHCP or mixed network address allocation, the network utilization value is set to zero. An on-demand network allocated range is based on the CIDR and subnet size that is specified in the network profile. To support both static and dynamic assignment in the deployment, the allocated range is divided into two ranges - one for static allocation and another for dynamic allocation.

Networks

Networks, also referred to as subnets, are logical subdivisions of an IP network. A network groups a cloud account, IP address or range, and network tags to control how and where to provision a cloud template deployment. Network parameters in the profile define how machines in the deployment can communicate with one another over IP layer 3. Networks can have tags.

You can add networks to the network profile, edit aspects of networks that are used by the network profile, and remove networks from the network profile.

When you add a network to the network profile, you can select available networks from a filtered list of vSphere and NSX networks. If the network type is supported for the cloud account type, you can add it to the network profile.

- **Network domain** or **Transport zone**

  A network domain or transport zone is the distributed virtual switch (dvSwitch) for the vSphere vNetwork Distributed PortGroups (dvPortGroup). A transport zone is an existing NSX concept that is similar to terms like dvSwitch or dvPortGroup.

  When using an NSX cloud account, the element name on the page is **Transport zone**, otherwise it is **Network domain**.

  For standard switches, the network domain or transport zone is the same as the switch itself. The network domain or transport zone defines the boundaries of the subnets within vCenter.

  A transport zone controls which hosts an NSX logical switch can reach to. It can span one or more vSphere clusters. Transport zones control which clusters and which virtual machines can participate in the use of a particular network. Subnets that belong to the same NSX transport zone can be used for the same machine hosts.

- **Domain**

  Represents the vCenter single sign-on domain for a target virtual machine. Domains are configured by a vCenter administrator during vSphere configuration. The domain determines the local authentication space in vCenter.

- **IPv4 CIDR** and **IPv4 default gateway**
vSphere cloud accounts, and vSphere machine components in the cloud template, support dual IPv6 and IPv4 internet protocol methods. For example, 192.168.100.14/24 represents the IPv4 address 192.168.100.14 and its associated routing prefix 192.168.100.0, or equivalently, its subnet mask 255.255.255.0, which has 24 leading 1-bits. The IPv4 block 192.168.100.0/22 represents the 1024 IP addresses from 192.168.100.0 to 192.168.103.255.

- IPv6 CIDR and IPv6 default gateway

vSphere cloud accounts, and vSphere machine components in the cloud template, support dual IPv6 and IPv4 internet protocol methods. For example, 2001:db8::/48 represents the block of IPv6 addresses from 2001:db8:0:0:0:0:0:0 to 2001:db8:0:ffff:ffff:ffff:ffff:ffff.

The IPv6 format is not supported for on-demand networks.

- DNS servers and DNS search domains

- Support public IP

Select this option to flag the network as public. Network components in a cloud template that have a network type: public property are matched to networks that are flagged as public. Further matching occurs during cloud template deployment to determine network selection.

- Default for zone

Select this option to flag the network as a default for the cloud zone. During cloud template deployment, default networks are preferred over other networks.

- Origin

Identifies the network source.

- Tags

Specifies one or more tags assigned to the network. Tags are optional. Tag matching affect which networks are available for your cloud template deployments.

Network tags exist on the network item itself, irrespective of the network profile. Network tags apply to every occurrence of the network they have been added to and to all network profiles that contain that network. Networks can be instanced into any number of network profiles. Regardless of network profile residency, a network tag is associated with that network wherever the network is used.

When you deploy a cloud template, constraint tags in a cloud template's network components are matched to network tags, including network profile capability tags. For network profiles that contain capability tags, the capability tags are applied to all the networks that are available for that network profile. The network and security settings that are defined in the matched network profile are applied when the cloud template is deployed.
Network Policies

By using network profiles, you can define subnets for existing network domains that contain static, DHCP, or a mixture of static and DHCP IP address settings. You can define subnets and specify IP address settings by using the **Network Policies** tab.

When using NSX-V, NSX-T, or VMware Cloud on AWS, network policy settings are used when a cloud template requires the `networkType: outbound` or `networkType: private` or when an NSX network requires `networkType: routed`.

Depending on the associated cloud account, you can use network policies to define settings for the `outbound`, `private`, and `routed` network types and for on-demand security groups. You can also use network policies to control existing networks when there is a load balancer associated with that network.

Outbound networks allow one way access to upstream networks. Private networks do not allow any outside access. Routed networks allow East/West traffic between the routed networks. The existing and public networks in the profile are used as the underlying or upstream networks.

Options for the following on-demand selections are described in the **Network Profiles** on-screen help and summarized below:

- **Do not create an on-demand network or on-demand security group**
  
  You can use this option when specifying an existing or public network type. Cloud templates that require an `outbound`, `private`, or `routed` network are not matched to this profile.

- **Create an on-demand network**
  
  You can use this option when specifying an `outbound`, `private`, or `routed` network type.

  Amazon Web Services, Microsoft Azure, NSX, vSphere, and VMware Cloud on AWS support this option.

- **Create an on-demand security group**
  
  You can use this option when specifying an `outbound` or `private` network type.

  A new security group is created for matched cloud templates if the network type is `outbound` or `private`.

  Amazon Web Services, Microsoft Azure, NSX, and VMware Cloud on AWS support this option.

Network policy settings can be cloud account type-specific. These settings are described in the on-screen signpost help and summarized below:

- **Network domain** or **Transport zone**
  
  A network domain or transport zone is the distributed virtual switch (dvSwitch) for the vSphere vNetwork Distributed PortGroups (dvPortGroup). A `transport zone` is an existing NSX concept that is similar to terms like `dvSwitch` or `dvPortGroup`. 
When using an NSX cloud account, the element name on the page is **Transport zone**, otherwise it is **Network domain**.

For standard switches, the network domain or transport zone is the same as the switch itself. The network domain or transport zone defines the boundaries of the subnets within vCenter.

A transport zone controls which hosts an NSX logical switch can reach to. It can span one or more vSphere clusters. Transport zones control which clusters and which virtual machines can participate in the use of a particular network. Subnets that belong to the same NSX transport zone can be used for the same machine hosts.

- **External subnet**
  
  An on-demand network with outbound access requires an external subnet that has outbound access. The external subnet is used to provide outbound access if requested in the cloud template - it does not control network placement. For example, the external subnet does not affect the placing of a private network.

- **CIDR**
  
  CIDR notation is a compact representation of an IP address and its associated routing prefix. The CIDR value specifies the network address range to be used during provisioning to create subnets. This CIDR setting on the **Network Policies** tab accepts IPv4 notation ending in /nn and containing values between 0 - 32.

- **Subnet size**
  
  This option specifies the on-demand network size, using IPv4 notation, for each isolated network in a deployment that uses this network profile. The subnet size setting is available for internal or external IP address management. The IPv6 format is not supported for on-demand networks.

- **Distributed logical router**
  
  For an on-demand routed network, you must specify a distributed logical network when using an NSX-V cloud account.

  A distributed logical router (DLR) is used to route east/west traffic between on-demand routed networks on NSX-V. This option is only visible if the account/region value for the network profile is associated to an NSX-V cloud account.

- **IP range assignment**
  
  The option is available for cloud accounts that support NSX or VMware Cloud on AWS, including vSphere.

  The IP range setting is available when using an existing network with an external IPAM integration point.

  You can select one of the following three options to specify an IP range assignment type for the deployment network:

    - **Static and DHCP**
Default and recommended. This mixed option uses the allocated **CIDR** and **Subnet range** settings to configure the DHCP server pool to support half of the address space allocation using the DHCP (dynamic) method and half of the IP address space allocation using the Static method. Use this option when some of the machines that are connected to an on-demand network require assigned static IP addresses and some require dynamic IP addresses. Two IP ranges are created.

This option is most effective in deployments with machines that are connected to an on-demand network, where some of the machines are assigned static IPs and other machines have IPs dynamically assigned by an NSX DHCP server and deployments where the load balancer VIP is static.

- **DHCP (dynamic)**
  This option uses the allocated CIDR to configure an IP pool on a DHCP server. All the IP addresses for this network are dynamically assigned. A single IP range is created for each allocated CIDR.

- **Static**
  This option uses the allocated CIDR to statically allocate IP addresses. Use this option when a DHCP server is not required to be configured for this network. A single IP range is created for each allocated CIDR.

- **IP blocks**
  The IP blocks setting is available when using an on-demand network with an external IPAM integration point.

  Using the IP block setting, you can add a named IP block, or range, to the network profile from your integrated external IPAM provider. You can also remove an added IP block from the network profile. For information about how to create an external IPAM integration, see [Add an external IPAM integration for Infoblox in vRealize Automation Cloud](#).

  External IPAM is available for the following cloud account/region types:
  - vSphere
  - vSphere with NSX-T
  - vSphere with NSX-V

- **Network Resources - External network**
  External networks are also referred to as existing networks. These networks are data-collected and made available for selection.

- **Network Resources - Tier-0 logical router**
  NSX-T uses the tier-0 logical router as a gateway to networks that are external to the NSX deployment. The tier-0 logical router configures outbound access for on-demand networks.

- **Network Resources - Edge cluster**
The specified edge cluster provides routing services. The edge cluster is used to configure outbound access for on-demand networks and load balancers. It identifies the edge cluster, or resource pool, where the edge appliance is to be deployed.

- **Network Resources - Edge datastore**

  The specified edge datastore is used to provision the edge appliance. This setting applies to NSX-V only.

Tags can be used to specify which networks are available to the cloud template.

**Load Balancers**

You can add load balancers to the network profile. Listed load balancers are available based on information that is data-collected from the source cloud account.

If a tag on any of the load balancers in the network profile matches a tag in a load balancer component in the cloud template, the load balancer is considered during deployment. Load balancers in a matched network profile are used when a cloud template is deployed.

For more information, see Using load balancer settings in network profiles in vRealize Automation Cloud and Networks, security resources, and load balancers in vRealize Automation Cloud.

**Security Groups**

When a cloud template is deployed, the security groups in its network profile are applied to the machine NICs that are provisioned. For an Amazon Web Services-specific network profile, the security groups in the network profile are available in the same network domain (VPC) as the networks that are listed on the Networks tab. If the network profile has no networks listed on its Networks tab, all available security groups are displayed.

You can use a security group to further define the isolation settings for an on-demand private or outbound network. Security groups are also applied to existing networks. You can also assign global security groups.

Listed security groups are available based on information that is data-collected from the source cloud account or added as an on-demand security group in a project cloud template. For more information, see Security resources in vRealize Automation Cloud.

Security groups are applied to all the machines in the deployment that are connected to the network that matches the network profile. As there might be multiple networks in a cloud template, each matching a different network profile, you can use different security groups for different networks.

**Note** In addition to specifying a security group, you can also select NSX networks (default) or vSphere networks or both. When you deploy a cloud template, vRealize Automation Cloud adds the allocated or specified security group to machine NICs that are connected to the allocated NSX network. Only machine NICs that are connected to an NSX network can be added to an NSX security group. If the machine NIC is connected to a vSphere network, the template deployment fails.
Adding a tag to an existing security group allows you to use the security group in a cloud template Cloud.SecurityGroup component. A security group must have at least one tag or it cannot be used in a cloud template. For more information, see Security resources in vRealize Automation Cloud and Networks, security resources, and load balancers in vRealize Automation Cloud.

**More information about network profiles, networks, cloud templates, and tags**

For more information about networks, see Network resources in vRealize Automation Cloud.

For examples of sample network component code in a cloud template, see Networks, security resources, and load balancers in vRealize Automation Cloud.

For sample network automation workflows, see the following VMware blog posts:

- Network Automation with Cloud Assembly and NSX – Part 1
- Network Automation with Cloud Assembly and NSX – Part 2
- Network Automation with Cloud Assembly and NSX – Part 3
- Network Automation with Cloud Assembly and NSX – Part 4

For more information about tags and tag strategy, see How do I use tags to manage Cloud Assembly resources and deployments.

**Using network settings in network profiles and cloud templates in vRealize Automation Cloud**

You use networks and network profiles in vRealize Automation Cloud to help define the behavior of network provisioning for your deployments.

In vRealize Automation Cloud, you can define cloud-specific network profiles. See Learn more about network profiles in vRealize Automation Cloud.

Using network and network profile settings, you can control how network IP addresses are used in vRealize Automation Cloud cloud templates and deployments.

**IPv4 and IPv6 support in vRealize Automation Cloud networks**

vRealize Automation Cloud networks support pure IPv4 or dual stack IPv4 and IPv6. Pure IPv6 is not currently supported.

While pure IPv4 is supported for all cloud account and integration types, dual stack IPv4 and IPv6 is supported only for vSphere cloud accounts and their endpoints.

IPv6 is currently not supported for use with load balancers, NSX on-demand networks, or external third-party IPAM providers.
External IPAM provider support

In addition to the supplied internal IPAM support, you can use an external IPAM provider to dynamically or statically allocate IP address for networks - as IP ranges for existing networks in your cloud template designs and deployments and IP blocks for on-demand networks in your cloud template designs and deployments.

Support for external IPAM providers, such as Infoblox, is available for vendor-specific IPAM integration points that you create by using the Infrastructure > Connections > Add Integration > IPAM menu sequence.

Options for defining external IPAM provider address information is available by using the Add IPAM IP Range option on the Network Policies > Add IPAM IP Range page.

For information about how to create an external IPAM integration point, see How to configure an external IPAM integration in vRealize Automation Cloud. For an example of how to create an IPAM integration point for a specific IPAM vendor, see Tutorial: Configuring a provider-specific external IPAM integration for vRealize Automation Cloud.

Network types

A network component in a cloud template is defined as one of the following networkType types.

<table>
<thead>
<tr>
<th>Network type</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>existing</td>
<td>Selects an existing network that is configured on the underlying cloud provider, such as vCenter, Amazon Web Services, and Microsoft Azure. An existing network is required by the outbound on-demand network. You can define a range of static IP addresses on an existing network.</td>
</tr>
<tr>
<td>public</td>
<td>Machines on a public network are accessible from the internet. An IT administrator defines these networks. The definition of a public network is identical to that of an existing network for networks that allow network traffic to occur along public networks.</td>
</tr>
<tr>
<td>private</td>
<td>An on-demand network type. Limits network traffic to occur only between resources on the deployed network. It prevents inbound and outbound traffic. In NSX, it can be equated to on-demand NAT one-to-many.</td>
</tr>
<tr>
<td>Network type</td>
<td>Definition</td>
</tr>
<tr>
<td>--------------</td>
<td>------------</td>
</tr>
<tr>
<td>outbound</td>
<td>An on-demand network type. Limits network traffic to occur between the compute resources in the deployment but also allows one-way outbound network traffic. In NSX, it can be equated to on-demand NAT one-to-many with external IP.</td>
</tr>
<tr>
<td>routed</td>
<td>An on-demand network type. Routed networks contain a routable IP space divided across available subnets that are linked together. The virtual machines that are provisioned with routed networks, and that have the same routed network profile, can communicate with each other and with an existing network. Routed networks are an on-demand network type that is available for NSX-V and NSX-T networks. Microsoft Azure and Amazon Web Services provides this connectivity by default. A routed network is only available for cloud template specification in a Cloud.NSX.Network network component.</td>
</tr>
</tbody>
</table>

For more information, see [More about network resources in vRealize Automation Cloud cloud templates](#).

For examples of populated cloud templates that contain network component data, see [Networks, security resources, and load balancers in vRealize Automation Cloud](#).

**Sample network scenarios**

You can expect the following behavior when you deploy a cloud template that uses the following network profile configuration.
<table>
<thead>
<tr>
<th>Network type or scenario</th>
<th>No network profiles available for cloud zone</th>
<th>Network profiles available for cloud zone</th>
</tr>
</thead>
<tbody>
<tr>
<td>No network</td>
<td>If no network is specified in the cloud template, a random network is selected from the same provisioning region as the compute. Preference is given to networks that are labeled as default. If no networks exist in an available provisioning region, provisioning fails.</td>
<td>A network is selected from a matched network profile. Preference is given to networks that are labeled as default. If none of the network profiles meet the criteria, provisioning fails.</td>
</tr>
<tr>
<td>Existing network</td>
<td>If the network component in the cloud template contains constraint tags, those constraints are used to filter the list of available networks. Constraint tags in the cloud templates' network component are matched to network tags and, if available, network profile constraint tags. From the filtered list of networks, a single network is selected from the same provisioning region as the compute. Preference is given to networks that are labeled as default. If after filtering based on constraints there are no networks in the provisioning region, provisioning fails.</td>
<td>A network is selected from a matching network profile. Preference is given to networks that are labeled as default. If none of the network profiles meet the criteria, provisioning fails. Network constraints can be used to filter existing networks in the profile based on their pre-assigned tags.</td>
</tr>
<tr>
<td>Public network</td>
<td>If the network has constraints, those constraints are used to filter the list of available networks that have the supports public IP attribute set. From the filtered list of networks, a random network is selected from the same provisioning region as the compute. Preference is given to networks that are labeled as default. If after filtering based on constraints there are no public networks in the provisioning region, provisioning fails.</td>
<td>A network with the supports public IP attribute is selected from a matching network profile. Preference is given to networks that are labeled as default. Network constraints can be used to filter existing public networks in the profile based on their pre-assigned tags.</td>
</tr>
<tr>
<td>Private network</td>
<td>Provisioning fails because private networks require information from a network profile.</td>
<td>A new network or new security group is created based on settings in the matched network profile. Network constraint tags can be used to filter network profiles and networks.</td>
</tr>
<tr>
<td>Network type or scenario</td>
<td>No network profiles available for cloud zone</td>
<td>Network profiles available for cloud zone</td>
</tr>
<tr>
<td>-------------------------</td>
<td>---------------------------------------------</td>
<td>------------------------------------------</td>
</tr>
<tr>
<td>Outbound network</td>
<td>Provisioning fails because outbound networks require information from a network profile.</td>
<td>A new network or new security group is created based on settings in the matched network profile. Network constraint tags can be used to filter network profiles and networks.</td>
</tr>
<tr>
<td>On-demand routed network</td>
<td>Provisioning fails because routed networks require information from a network profile.</td>
<td>For NSX-V we need DLR (Distributed Logical Router) selection. For NSX-T and VMware Cloud on AWS, we require similar on-demand settings as private and outbound.</td>
</tr>
<tr>
<td>Example Wordpress use case with existing or public networks</td>
<td>Provisioning occurs as described for an existing network or public network.</td>
<td>See above descriptions for existing network and public network behavior. See Tutorial: Setting up and testing multi-cloud infrastructure and deployments in Cloud Assembly.</td>
</tr>
<tr>
<td>Example Wordpress use case with existing or public networks and private or outbound networks</td>
<td>Provisioning fails because the network requires information from a network profile.</td>
<td>See above descriptions for a private network and an outbound network. See Tutorial: Setting up and testing multi-cloud infrastructure and deployments in Cloud Assembly.</td>
</tr>
<tr>
<td>Example Wordpress use case with load balancer</td>
<td>Provisioning fails because a load balancer requires information from a network profile. Provisioning can occur when existing load balancers are present.</td>
<td>A new load balancer is created based on the network profile configuration. You can specify an existing load balancer that has been enabled in the network profile. Provisioning fails if you request an existing load balancer, but none meet the constraints in the network profile. See Tutorial: Setting up and testing multi-cloud infrastructure and deployments in Cloud Assembly.</td>
</tr>
</tbody>
</table>

**Using security group settings in network profiles and cloud template designs in vRealize Automation Cloud**

You can define and change security group settings in network profiles and in cloud template designs.

You can use security group capabilities in several ways:

- Existing security group specified in a network profile
You can add an existing security group to a network profile. When a cloud template design uses that network profile, its machines are grouped together as members of the security group. This method does not require that you add a security group resource to a cloud template design. You can also use a load balancer in this configuration. For related information, see More about load balancer resources in vRealize Automation Cloud cloud templates.

- **Security group component associated to machine resource in a cloud template design**

You can drag and drop a security group resource on to a cloud template design and bind the security group resource to a machine NIC by using constraint tags on the existing security group resource in the cloud template design and on the existing security group in the data-collected resource. You can also make this association by connecting the objects together with a connection line on the cloud template design canvas, similar to how you associate networks to machines on the design canvas.

When you drag and drop a security group resource onto the cloud template design canvas, it can be of type existing or new. If it's an existing security group type, you should add a tag constraint value as prompted. If it's a new security group type, you can configure firewall rules.

- **An existing security group allocated with tag constraints and associated with a machine NIC in the cloud template**

For example, you can associate a security group resource with a machine NIC (in a machine resource) in the cloud template design by matching tags between the two resources.

As an example for NSX-T when tags are specified in the source endpoint, you can use NSX-T tags specified in your NSX-T application. You can then use an NSX-T tag, specified as a constraint on a network resource in a cloud template design, where the network resource is connected to a machine NIC in the cloud template design. NSX-T tags enable you to dynamically group machines by using a pre-defined NSX-T tag that is data-collected from the NSX-T source endpoint. Use a logical port when you create the NSX-T tag in NSX-T.

- **Firewall rules in an on-demand security group resource in a cloud template design**

You can add firewall rules to an on-demand security group in the cloud template design. For information about available firewall rules, see More about security group and tag resources in vRealize Automation Cloud cloud templates.

**Learn more**

For information about defining security groups in network profiles, see Learn more about network profiles in vRealize Automation Cloud.

For information about viewing and changing security groups settings in infrastructure resource pages, see Security resources in vRealize Automation Cloud.

For information about defining security groups in cloud template designs, see More about security group and tag resources in vRealize Automation Cloud cloud templates.
For examples of security group resources in cloud template designs, see Networks, security resources, and load balancers in vRealize Automation Cloud.

**Using load balancer settings in network profiles in vRealize Automation Cloud**

You can configure load balancer settings in your network profile configuration.

You can add an existing load balancer to a network profile by using the Load Balancer tab.

You can add a load balancer to a cloud template design by associating it to a network profile that contains one or more load balancers or directly by using a load balancer resource in the cloud template design canvas or code.

**Examples for including a load balancer VIP based on security group use in a network profile**

There are two types of security groups that you can use in a network profile – an existing security group that you select from the Security Groups tab and an on-demand security group that you create by using an isolation policy on the Network Policies tab.

When a load balancer VIP is associated to a security group based on network profile settings, the security group configuration is supplied by the network profile.

The following table illustrates some sample scenarios.

<table>
<thead>
<tr>
<th>Cloud template design topology - associated resources</th>
<th>Network profile configuration</th>
<th>Security group membership</th>
</tr>
</thead>
<tbody>
<tr>
<td>One-armed load balancer with VIP on private network, and a machine on the same private network.</td>
<td>The selected network profile uses isolation policy defined as an on-demand security group.</td>
<td>The machine NIC and the load balancer VIP are added to the isolation security group.</td>
</tr>
<tr>
<td>One-armed load balancer with VIP on private network, and a machine on the same private network.</td>
<td>The selected network profile uses an existing security group and uses isolation policy defined as an on-demand security group.</td>
<td>The machine NIC and the load balancer VIP are added to the isolation security group and the existing security group.</td>
</tr>
<tr>
<td>Two-armed load balancer with VIP on a public network and machine on a private network.</td>
<td>The selected network profile uses an existing security group and uses isolation policy defined as an on-demand security group.</td>
<td>The machine NIC and the load balancer VIP are added to the isolation security group and the existing security group.</td>
</tr>
<tr>
<td>Two-armed load balancer with VIP on a public network and a machine on a private network.</td>
<td>The selected network profile uses an existing security group.</td>
<td>The machine NIC and the load balancer VIP are added to the existing security group.</td>
</tr>
</tbody>
</table>
| Two-armed load balancer, VIP is on network 1 and the machine is on network 2. | Two network profiles:  
- Network profile 1: Uses an existing security group 1.  
- Network profile 2: Uses an existing security group 2. | The load balancer lands on network profile 1 and the machine lands on network profile 2.  
The load balancer VIP is added to security group 1 and the machine NIC is added to security group 2. |
Learn more

For information about adding load balancer resources to a cloud template design, see More about load balancer resources in vRealize Automation Cloud cloud templates.

For examples of cloud template designs that include load balancers, see Networks, security resources, and load balancers in vRealize Automation Cloud.

How do I configure a network profile to support an on-demand network for an external IPAM integration in vRealize Automation Cloud

You can configure a network profile to support blocks of IP addresses for an on-demand network when that network profile is used in a vRealize Automation Cloud cloud template that uses external IPAM integration.

Using an existing integration for a particular external IPAM provider, you can provision on-demand network to create a new network in the external IPAM system.

Using this process, you configure a block of IP addresses instead of supplying a parent CIDR (as is done when using vRealize Automation Cloud’s internal IPAM). The IP address block is used during on-demand network provisioning to segment the new network. The IP blocks are data-collected from the external IPAM provider, provided the integration supports on-demand networking. For example, when using an Infoblox IPAM integration, IP blocks represent Infoblox network containers.

When you use an on-demand network profile and an external IPAM integration in a cloud template, the following events occur when the cloud template is deployed:

- A network is created in the external IPAM provider.
- A network is also created in vRealize Automation Cloud, reflecting the new network configuration from the IPAM provider, including settings such as CIDR and gateway properties.
- The IP address for the deployed virtual machine is fetched from the newly created network.

In this on-demand networking example, you configure a network profile to allow a cloud template deployment to provision a machine to an on-demand network in vSphere by using Infoblox as the external IPAM provider.

For related information, see How do I configure a network profile to support an existing network for an external IPAM integration in vRealize Automation Cloud. Both network configuration examples fit within the overall vendor-specific workflow for external IPAM integration at Tutorial: Configuring VMware Cloud on AWS for vRealize Automation Cloud.
Prerequisites

While the following prerequisites apply to the person who creates or edits the network profile, the network profile itself would be applicable when used by a cloud template deployment that contains an IPAM integration. To learn about vendor-specific IPAM integration points, see How to configure an external IPAM integration in vRealize Automation Cloud.

This sequence of steps is shown in the context of an IPAM provider integration workflow. See Tutorial: Configuring a provider-specific external IPAM integration for vRealize Automation Cloud.

- Verify that you have cloud administrator credentials. See Credentials required for working with cloud accounts in vRealize Automation Cloud.
- Verify that you have the cloud administrator user role. See What are the vRealize Automation Cloud user roles.
- Verify that you have an account with the external IPAM provider, for example Infoblox or Bluecat, and that you have the correct access credentials to your organization's account with the IPAM provider. In this example workflow, the IPAM provider is Infoblox.
- Verify that you have an IPAM integration point for the IPAM provider and that the IPAM package used to create the IPAM integration supports on-demand networks. See Add an external IPAM integration for Infoblox in vRealize Automation Cloud.

While the Infoblox IPAM package supports on-demand networks, if you are using an external IPAM integration for a different provider, verify that their IPAM integration package supports on-demand networks.

Procedure

1. To configure a network profile, click Infrastructure > Configure > Network Profiles.
2. Click New Network Profile.
3. Click the Summary tab and specify the following sample settings:
   - Specify a vSphere cloud account/region, for example `vSphere-IPAM-OnDemandA/Datacenter`.
     This example assumes use of a vSphere cloud account that is not associated with an NSX cloud account.
   - Name the network profile, for example `Infoblox-OnDemandNP`.
   - Add a capability tag for the network profile, for example `infoblox_ondemandA`.
     Make note of the capability tag value, as you must also use it as a cloud template constraint tag to make the network profile association to be used when provisioning the cloud template.
4. Click the Network Policies tab and specify the following sample settings:
   - From the Isolation policy drop-down menu, select On-demand network.
This option allows you to use external IPAM IP blocks. Depending on the cloud account, new options appear. For example, the following options appear when using a vSphere cloud account that is associated to an NSX cloud account:

- Transport zone
- Tier-0 logical router
- Edge cluster

For this example, the vSphere cloud account is not associated to NSX, so the **Network domain** menu option appears.

- Leave the **Network domain** option blank.

5. Click **External** as the address management **Source**.

6. Click **Add IP Block**, which opens the **Add IPAM IP Block** page.

7. From the **Provider** menu on the **Add IPAM IP Block** page, select an existing external IPAM integration. For example, select the **Infoblox_Integration** integration point from **Add an external IPAM integration for Infoblox in vRealize Automation Cloud** in the example workflow.

8. From the **Address spaces** menu, select one of the available and listed IP blocks, for example **10.23.118.0/24** and add it.

   If the IPAM provider supports address spaces, the **Address spaces** menu appears. For an Infoblox integration, address spaces are represented by Infoblox network views.

9. Select a **Subnet size**, such as **/29 (-6 IP addresses)**.

10. Click **Create**.

**Results**

A network profile is created that can be used to provision an on-demand network using the specified external IPAM integration. The following sample cloud template shows a single machine to be deployed to a network that is defined by this new network profile.

```yaml
formatVersion: 1
inputs: {}
resources:
  Cloud_Machine_1:
    type: Cloud.Machine
    properties:
      image: ubuntu
      flavor: small
      networks:
        - network: '${resource.Cloud_Network_1.id}'
          assignment: static
  Cloud_Network_1:
    type: Cloud.Network
    properties:
      networkType: private
```
constraints:
- tag: infoblox_ondemandA

**Note**  When the cloud template is deployed, the first available network in the specified IP block is fetched and considered to be the network CIDR. If you are using an NSX network in the cloud template, you can instead set the CIDR of the network manually by using the network property `networkCidr`, as shown below, to manually set a CIDR and override the settings for IP blocks and subnet size that are specified in the associated network profile.

```
Cloud_Network_1:
  type: Cloud.Network
  properties:
    networkCidr: 10.10.0.0/16
```

How do I configure a network profile to support an existing network for an external IPAM integration in vRealize Automation Cloud

You can configure a network profile to support IP address ranges for an existing network when that network profile is used in a vRealize Automation Cloud blueprint that uses external IPAM integration.

An example is provided within the context of a vendor-specific sample workflow at [Configure a network and network profile to use external IPAM for an existing network in vRealize Automation Cloud](#). The overall vendor-specific workflow for external IPAM integration is at [Tutorial: Configuring VMware Cloud on AWS for vRealize Automation Cloud](#).

For related information, see [How do I configure a network profile to support an on-demand network for an external IPAM integration in vRealize Automation Cloud](#).

How to add Cloud Assembly storage profiles that account for different requirements

A 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.

Select **Infrastructure > Configure > Storage Profiles** and click **New Storage Profile**.

Learn more about storage profiles in vRealize Automation Cloud

A cloud account region contains storage profiles that let the cloud administrator define storage for the region in vRealize Automation Cloud.
What does a storage profile do

Storage profiles include disk customizations, and a means to identify the type of storage by capability tags. Tags are then matched against provisioning service request constraints to create the desired storage at deployment time.

Storage profiles are organized under cloud-specific regions. One cloud account might have multiple regions, with multiple storage profiles under each.

Vendor-independent placement is possible. For example, you might have three different vendor accounts and a region in each. Each region includes a storage profile that is capability tagged as fast. At provisioning time, a request containing a fast hard constraint tag looks for a matching fast capability, regardless of which vendor cloud is supplying the resources. A match then applies the settings from the associated storage profile during creation of the deployed storage item.

**Note**  Different cloud storage might have different performance characteristics but still be considered the fast offering by the administrator who tagged it.

Capability tags that you add to storage profiles should not identify actual resource targets. Instead, they describe types of storage. For more about actual resources, see [Storage resources in vRealize Automation Cloud](#).

Disk allocation with machines

In a project with multiple cloud zones that belong to different cloud accounts, a disk follows the machine even if the disk isn't attached to the machine. This behavior keeps the resources together to prevent failure when you decide to attach the disk later.

For example, the following design won't work. The cloud template attempts to use location constraints to separate the disk, but the deployment returns a *No matching placement* error instead.

If you need to place a disk in a different cloud account, use a separate deployment to deploy the disk.

```yaml
resources:
 Machine1:
   type: Cloud.vSphere.Machine
   properties:
     image: ubuntu
     flavor: small
     constraints:
       - tag: 'location:siteA'

 Disk1:
   type: Cloud.vSphere.Disk
   properties:
     capacityGb: 1
     constraints:
       - tag: 'location:siteB'
```
First class and standard disks

By using the **Disk type** option on the storage profile page, or by using the vRealize Automation Cloud API, you can create a storage profile to support first class disk (FCD) or standard disk storage. In effect, the first class disk option creates a vSphere storage profile.

- **First class disk**

  First class disks can exist independent from a vSphere virtual machine. A first class disk also has life-cycle management capabilities that can operate independently of a virtual machine. First class disks are available for vSphere 6.7 Update 2 and later, and are currently implemented in vRealize Automation Cloud as an API-only feature.

  For information about FCD storage, including the capabilities that are available from the vRealize Automation Cloud API, and links to the API documentation itself, see [What can I do with first class disk storage in vRealize Automation Cloud](#).

- **Standard disk**

  Standard disk storage is created and managed as an integrated component of a virtual machine.

  For information about standard disk storage, see [What can I do with standard disk storage in vRealize Automation Cloud](#) and [What can I do with persistent disk storage in vRealize Automation Cloud](#).

**Azure server-side disk encryption**

For Azure resources, if you elect to support encryption in a managed disk storage profile, you also select disk encryption that has an associated key. Available encryption and keys correspond to the disk encryption sets configured in Azure for the location.
How to use Pricing Cards

Cloud Assembly pricing cards help cloud administrators define and assign the pricing policy for the monetary impact of your individual deployments to help you manage resources.
Pricing cards define the rates for a pricing policy. The pricing policy can then be assigned to specific projects to define a total price. After creating a vRealize Operations or CloudHealth endpoint, a predefined Default Rate Card is available with a cost equal to price configuration on the Infrastructure > Pricing Cards tab. You can create pricing cards that apply to projects only or cloud zones. By default, all new pricing cards are applied to projects.

**Note** If you change the All pricing cards are applied to setting, all existing pricing card assignments are deleted. Also, if the vRealize Operations endpoint is deleted from Cloud Assembly, all pricing cards and assignments are also deleted.

The price of a deployment over time appears on both the deployment card and Project as the month-to-date price, which resets to zero at the beginning of each month. The component cost breakdowns are available in the deployment details. Providing this information at the deployment level informs the cloud administrator, but it also helps the members understand the impact their work might have on budgets and long-term development.

You can choose to display pricing information from users in Cloud Assembly and Service Broker by selecting the Display pricing information button. If left disabled, the pricing information is hidden from Cloud Assembly and Service Broker users.

**How is price calculated**

The initial price that you see at the deployment level for your compute and storage resources are based on industry standard benchmark rates, and then calculated over time. The rate is applied to hosts and the service calculates the CPU and memory rates. The server recalculates the price every 6-hours.

New policies, assignments, and upfront pricing are priced during the next occurring data collection cycle. By default, the data collection cycle is run every 5 minutes. It can take up to 6-hours for new policies or changes to be updated in projects and deployments.

**How do I estimate the price of my deployments and projects**

Before deploying a catalog item, you can use the upfront price as a price estimate for your deployment. To view the price in VMware Cloud Assembly, you must have a vRealize Operations integration endpoint configured with pricing enabled and currency preset.
Daily Price Estimate

- **price-service-f309c00** $0.54
- **Cloud_vSphere_Machine_1** $0.53
  - Compute $0.39
  - Storage $0.03
  - Additional charges $0.11
- **Cloud_vSphere_Disk_1** $0.01
  - Storage $0.01

For an upfront price estimation, the size of boot disk per VM is always 8 GB.

The upfront price of a deployment is a daily price estimate, based on the allocation of a resource, for a given catalog item before it is deployed. After a catalog item is deployed, you can view the month-to-date price as an aggregate of the upfront price on the **Deployment** and **Infrastructure > Projects** tabs. Upfront pricing is supported for private cloud resources such as vSphere Machine and vSphere Disk, Cloud Assembly catalog items, and cloud agnostic items with vCenter configured for private cloud.

**Note** Upfront pricing is not supported for public cloud resources, or non-vSphere Machine or Disk private cloud resources.

To estimate the cost of your deployment, from the Catalog select a catalog item and click **Request > Calculate**. If the price is acceptable, click **Submit**.

For showback purposes, you can use project pricing cards to estimate the total price of all your projects.

To estimate the cost of a project, on the Infrastructure Pricing Card page, next to All pricing cards are applied to: click **Edit** and select **Projects**.

If you change the All pricing cards are applied to setting, all existing pricing card assignments are deleted. Create pricing cards and assignments using a cost-based approach.
How to create pricing cards for vSphere and VMC

You can create and assign a pricing card to projects or cloud zones, depending on the pricing strategy determined by the cloud administrator for private cloud deployments.

Pricing cards are customizable based on user-selected parameters. After configuring a pricing card, you can assign it to one or more projects and cloud zones determined by the pricing strategy.

You can manually refresh the price server at any time on the vROPs Endpoint page, Infrastructure > Integrations > vROPs Endpoint > . In the vCenter servers section, click Sync.

When manually refreshing the price server using the Sync option, the price is recalculated for all projects in the organization. Depending on how many projects your organization has this process might be intensive and take time.

To adjust your private cloud costs, select Infrastructure > Cost > Private Cloud. Select Add Overall Markup to adjust costs for all clusters or manually add markup to an individual cluster in the Add Markup % column.

After creating and assigning a pricing card, you can view the price history of your deployments and projects. To view the price history, navigate to your deployment and click Price. The price analysis provides an overview and detailed view of the deployment price along with the price month-to-date value. You can change the graphical representation to display the deployment price as daily, weekly, or monthly values. Also, you can specify an exact date range or month for the price history.

To view the price breakdown by cost component, click Details.

Prices are determined by costed component types.
Table 4-1. Costed Component Types

<table>
<thead>
<tr>
<th>Blueprint Component Type</th>
<th>Service Name/Object Type</th>
<th>Blueprint Resource Type</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cloud Agnostic</td>
<td>Machine</td>
<td>Cloud.Machine</td>
<td>If an agnostic machine is configured with vSphere, you can view deployment cost.</td>
</tr>
<tr>
<td></td>
<td>Disk</td>
<td>Cloud.Volume</td>
<td>If an agnostic disk is attached to a virtual machine that is configured with vSphere, you can view deployment cost.</td>
</tr>
<tr>
<td></td>
<td>vSphere disk</td>
<td>Cloud.vSphere.Disk</td>
<td>Deployed using a cloud-specific blueprint attached to a virtual machine.</td>
</tr>
<tr>
<td>VMware Managed Cloud (VMC)</td>
<td>vSphere machine</td>
<td>Cloud.vSphere.Machine</td>
<td>VMC only supports rate-based pricing cards (cost based pricing cards are not supported).</td>
</tr>
<tr>
<td></td>
<td>vSphere disk</td>
<td>Cloud.vSphere.Disk</td>
<td></td>
</tr>
</tbody>
</table>

Prerequisites

You must configure a vRealize Operations endpoint before you can configure pricing cards. To configure the vRealize Operations endpoint navigate to **Infrastructure > Connections > Integrations > Add Integration**.

**Note** When multiple vRealize Operations endpoints are added they must not monitor the same vCenter.

Procedure

1. Navigate to **Infrastructure > Pricing Cards > New Pricing Card**.
2. On the Summary tab, enter a name and description for the pricing card. Once the policy is defined on the pricing tab, the Overview table is populated with pricing card rates.
   
   **Note** The currency unit is determined by the valued selected in vRealize Operations.

3. Optional. Select the **Default for unassigned projects?** check box to assign this pricing card to all unassigned projects by default.
4 Click **Pricing**, and configure the details of your pricing policy.

**Table 4-2. Pricing Policy Configuration**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Basic Charges</strong></td>
<td>Enter a name and description for your policy. Select cost or rate based.</td>
</tr>
<tr>
<td></td>
<td>- <strong>Cost</strong> - The cost is defined in vRealize Operations. If selected, a multiplication factor is required. For example, if you select 1.1 as a factor, the cost is multiplied by 1.1 resulting in a 10% increase to the calculated cost. The price equation using cost is: &lt;cost&gt; x &lt;multiplication factor&gt; = Price</td>
</tr>
<tr>
<td></td>
<td>- <strong>Rate</strong> - If selected, you must use absolute values to determine the cost. The price equation using rate is: &lt;Rate&gt; = Price. Select a rate interval from the drop down list to specify how this rate is charged.</td>
</tr>
<tr>
<td></td>
<td>In the basic charges section, you define the cost or rate for CPU, memory, storage, and additional miscellaneous costs.</td>
</tr>
</tbody>
</table>

| **Guest OSes**   | You can define a Guest OS charge by clicking **Add Charge**. Enter the guest OS name and define the charging method and base rate.              |
|                  | - **Recurring** - enter a base rate and define recurring interval as the charge period. The absolute rate value is required and it is added to the overall price. |
|                  | - **One time** - define the one-time base rate charge. The absolute value is required and it is added as a one time price.                   |
|                  | - **Rate Factor** - A multiplication factor is required that is applied to the select charge category. For example, if you select CPU Charge and a rate factor of 2. The Guest OS CPU is charged as 2 times the standard cost value. |

You can add multiple Guest Oses with different rates by clicking **Add Charge** and configuring an additional charge policy.

**Note** Upfront charges for Guest Oses are not shown on the summary page, even though they are part of the policy.
Table 4-2. Pricing Policy Configuration (continued)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tags</td>
<td>You can define a Tag charge by clicking <strong>Add Charge</strong>. Select the Tag name and define the charging method and base rate.</td>
</tr>
<tr>
<td></td>
<td>- Recurring - enter a base rate and define recurring interval as the charge period. The absolute rate value is required and it is added to the overall price.</td>
</tr>
<tr>
<td></td>
<td>- One time - define the one-time base rate charge. The absolute value is required and it is added as a one time price.</td>
</tr>
<tr>
<td></td>
<td>- Rate Factor - A multiplication factor is required that is applied to the select charge category. Select how to charge the Tag based on powered on state.</td>
</tr>
<tr>
<td></td>
<td>You can add multiple Tags with different rates by clicking <strong>Add Charge</strong> and configuring an additional charge policy.</td>
</tr>
<tr>
<td></td>
<td><strong>Note</strong> Additional charges in the calculated final price include on tags on VMs and does not include tags on disks and networks.</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Custom Properties</td>
<td>You can define a Custom Property charge by clicking <strong>Add Charge</strong>. Enter the property name and value, and define the charging method and base rate.</td>
</tr>
<tr>
<td></td>
<td>- Recurring - enter a base rate and define recurring interval as the charge period. The absolute rate value is required and it is added to the overall price.</td>
</tr>
<tr>
<td></td>
<td>- One time - define the one-time base rate charge. The absolute value is required and it is added as a one time price.</td>
</tr>
<tr>
<td></td>
<td>- Rate Factor - A multiplication factor is required that is applied to the select charge category. Select how to charge the custom property based on powered on state.</td>
</tr>
<tr>
<td></td>
<td>You can add multiple custom properties with different rates by clicking <strong>Add Charge</strong> and configuring an additional charge policy.</td>
</tr>
<tr>
<td>Overall Charges</td>
<td>Define any additional charge you would like to add to the pricing policy. You can add both one time and recurring charges.</td>
</tr>
</tbody>
</table>

One time charges are not shown in the price estimate of a catalog item or on the summary tab. Only the daily price estimate for a given catalog item is shown.
5 Click the **Assignments** tab and click **Assign Projects**. Select one or more projects to assign the pricing card to.

**Note** By default pricing cards are applied to projects. On the **Infrastructure > Pricing Cards** tab, you can select to apply pricing cards to cloud zones. If cloud zones were selected, you would click **Assign Cloud Zones** on the Assignments tab.

6 Click **Create** to save and create your pricing policy.

**Results**

Your new pricing policy appears on the Pricing Cards page. To view or edit the policy details and configuration click **Open**.

**How to view pricing for AWS and Azure**

After creating a CloudHealth integration, you can view the pricing and cost information for public cloud deployments in Cloud Assembly.

Before you can view pricing for AWS and Azure in Cloud Assembly, you must have an existing CloudHealth integration endpoint. You can create a CloudHealth integration on the **Infrastructure > Integrations** page.

If you have both private cloud and public cloud pricing enabled with a vRealize Automation and vROPs integration and a CloudHealth integration, you must align the currencies across CloudHealth and vROPs to view an accurate price aggregation in Cloud Assembly. If the currencies are not aligned, Cloud Assembly only shows private cloud prices aggregated as a project price.

CloudHealth organizes assets into perspective groups. Using these groups, you can generate current and historical price reports for public cloud deployments. By default, the perspective syncs deployments every 12 hours between CloudHealth and vRealize Automation. After being synced, CloudHealth generates a cost for all perspective groups.

If a new deployment is provisioned by vRealize Automation, the associated costs are synced 12 hours later when the perspective is synced.

You can manually generate prices from CloudHealth by clicking **Infrastructure > CloudHealth Integration > Sync Prices**. Prices are generated at a deployment level only. Resource level prices for public cloud deployments is not supported.

Currently, Cloud Assembly does not support pricing for GCP deployments.
Prices are synced every 6 hours. You can view the price history in Cloud Assembly by navigating to your AWS or Azure deployment and clicking **Price**. The price analysis provides an overview and detailed view of the deployment price along with the price month-to-date value. You can change the graphical representation to display the deployment price as daily, weekly, or monthly values. Also, you can specify an exact date range or month for the price history.

You can also view the cost history for public cloud deployments directly in CloudHealth by selecting the deployment type, clicking **Cost** and selecting **Current** or **History**. The cost analysis provides an overview of perspective groups and their associated costs that you can filter and sort by length of time.
Prices are determined by these costed components types.

**Table 4-3. Costed Component Types by Platform**

<table>
<thead>
<tr>
<th>AWS</th>
<th>Azure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cloud.AWS.Volume</td>
<td>Cloud.Azure.Disk</td>
</tr>
<tr>
<td>Cloud.Service.AWS.Route53.Zone</td>
<td></td>
</tr>
</tbody>
</table>

**Cloud Agnostic Resource Types**

| Cloud.LoadBalancer                       | Cloud.Machine                             |

How do I use tags to manage Cloud Assembly resources and deployments

Tags are a critical component of Cloud Assembly that drive the placement of deployments through matching of capabilities and constraints. You must understand and implement tags effectively to make optimal use of Cloud Assembly.

Fundamentally, tags are labels that you add to Cloud Assembly items. You can create any tags that are appropriate for your organization and implementation. Tags function as much more than labels though, because they control how and where Cloud Assembly uses resources and infrastructure to build deployable services. Tags also support governance within Cloud Assembly.
Tag structure

Structurally, tags must follow the name:value pair convention, but otherwise their construction is largely free form. Throughout Cloud Assembly, all tags appear the same, and tag functionality is determined by context.

For example, tags on infrastructure resources function primarily as capability tags because Cloud Assembly uses them to match resources with deployments. Secondarily, they also identify the resources.

Tag function

The primary function of tags is to express capabilities and constraints that Cloud Assembly uses to define deployments. Context determines the function of tags. Tags placed on cloud zones, network and storage profiles, and individual infrastructure resources function as capability tags and define desired capabilities for infrastructure used in deployments. Tags placed on cloud templates function as constraints that define resources for deployments. Also, cloud administrators can place constraint tags on projects to exercise a form of governance over those projects. These constraint tags are added to other constraints expressed in cloud templates.

During provisioning, Cloud Assembly matches these capabilities with constraints, also expressed as tags, in cloud templates to define deployment configuration. This tag-based capability and constraint functionality serves as the foundation for deployment configuration in Cloud Assembly. For instance, you can use tags to make infrastructure available only on PCI resources in a particular region.
On a secondary level, tags also facilitate search and identification of storage and network items and other infrastructure resources.

For example, assume that you are setting up cloud zones and you have many compute resources available. If you have tagged your compute resources appropriately, you can use the search function on the Compute tab of the Cloud Zone page to filter the resources that are associated with that particular cloud zone.

Also, the Cloud Assembly Tag Management page and resource configuration pages contain search functions that enable you to locate items by tag names. Using logical and human readable tags for these items is key to facilitating this search and identification function.

Take a look at the following Youtube video for more information and examples of tag usage: https://youtu.be/4zNQ33RyQio
External tags

Cloud Assembly might also contain external tags. These tags are imported automatically from cloud accounts that you associate with a Cloud Assembly instance. These tags might be imported from vSphere, AWS, Azure or other external software products. When imported, these tags are available for use in the same manner as user created tags.

Managing tags

You can use the Tag Management page in Cloud Assembly to monitor and manage your tags library. You can also create tags on this page. In addition, the Tag Management page is the only page on which you can view and identify external tags.

Tag strategy

To minimize confusion, before creating tags in Cloud Assembly, devise an appropriate tag strategy and tagging conventions, so that all users who create and use tags understand what they mean and how they should be used. See Creating a tagging strategy.

Creating a tagging strategy

You must carefully plan and implement an appropriate tagging strategy based on your organization’s IT structure and goals to maximize Cloud Assembly functionality and minimize potential confusion.

While tagging serves several common purposes, your tagging strategy must be tailored to your deployment needs, structure, and goals.
Best practices for tagging

Some general characteristics of an effective tag strategy:

- Design and implement a coherent strategy for tagging that relates to the structure of your business and communicate this plan to all applicable users. A strategy must support your deployment needs, use clear human readable language, and be understandable to all applicable users.

- Use simple, clear, and meaningful names and values for tags. For instance, tag names for storage and network items should be clear and coherent so that users can readily understand what they are selecting or reviewing tag assignments for a deployed resource.

- Though you can create tags using a name with no value, as a best practice, it is more appropriate to create an applicable value for each tag name, as this makes the tag usage clear to other users.

- Avoid creating duplicate or extraneous tags. For example, only create tags on storage items that relate to storage issues.

Tagging implementation

Map out your primary considerations for a basic tagging strategy. The following list shows typical considerations to consider when mapping your strategy. Be aware that these considerations are representative rather than definitive. You might have other considerations that are highly relevant to your use cases. Your specific strategy must be appropriate for your specific use cases.

- How many different environments do you deploy to. Typically, you will create tags that represent each environment.

- How are your compute resources structured and used to support deployments.

- How many different regions or locations do you deploy to. Typically, you will create tags, at the profile level, that represents each of these different regions or locations.

- How many different storage options are available for deployments, and how do you want to characterize them. These options should be represented by tags.

- Categorize your networking options and create tags to accommodate all applicable options.

- Typical deployment variables. For example, how many different environments do you deploy to. Typically, many organizations have Test, Dev, and Production environments at a minimum. You will want to create and coordinate constraint tags and cloud zone capability tags that match so that you can easily set up deployments to one or more of these environments.

- Coordinate tags on network and storage resources so that they make logical sense in context of the network and storage profiles in which they are used. The resource tags can serve as a finer level of control over the resource deployment.
Coordinate cloud zone and network profile capability tags, and other capability tags, with constraint tags. Typically, your administrator will create capability tags for cloud zones and network profiles first, and then other users can design cloud templates with constraints that match these capability tags.

After you understand the important considerations for your organization, you can plan appropriate tag names that address these considerations in a logical manner. Then, create an outline of your strategy and make it available to all users with privileges to create or edit tags.

As a useful implementation approach, you can begin by tagging all of your compute infrastructure resources individually. As noted, use logical categories for tag names that relate to the specific resource. For instance, you might tag storage resources as tier1, tier2, etc. Also, you might tag compute resources based on their operating system, such as Windows, Linux, etc.

After you tag resources, you can then consider the approach to creating tags for cloud zone and storage and network profiles that best suits your needs.

Using capability tags in Cloud Assembly

In Cloud Assembly, capability tags enable you to define deployment capabilities for infrastructure components. Along with constraints, they function as the basis of placement logic in vRealize Automation Cloud.

You can create capability tags on compute resources, cloud zones, images and image maps, and networks and network profiles. The pages for creating these resources contain options for creating capability tags. Alternatively, you can use the Tag Management page in Cloud Assembly to create capability tags. Capability tags on cloud zones and network profiles affect all resources within those zones or profiles. Capability tags on storage or network components affect only the components on which they are applied.

Typically, capability tags might define characteristics such as location for a compute resource, adapter type for a network, or tier level for a storage resource. They can also define environment location or type and any other business considerations. As with your overall tagging strategy, you should organize your capability tags in a logical manner for your business needs.

Cloud Assembly matches capability tags from cloud zones with constraints on cloud templates at deployment time. So, when creating and using capability tags, you must understand and plan to create appropriate cloud template constraints so that matching will occur as expected.

For example, the 2. Add cloud zones topic in the Tutorial: Setting up and testing multi-cloud infrastructure and deployments in Cloud Assembly included with the documentation describes how to create dev and test tags for the OurCo-AWS-US-East and OurCo AWS-US-West cloud zones. In this tutorial, these tags indicate that the OurCo-AWS-US-East zone is a development environment, and the OurCo-AWS-US-West zone is a test environment. If you create analogous constraint tags in cloud templates, these capability tags enable you to direct deployments to the desired environments.
Tag inheritance

Cloud Assembly uses tag inheritance to selectively propagate tags on cloud accounts to other related resource. Specifically when you create tags on a cloud account, they also become effective on all storage profiles an compute resources that correspond to that cloud account.

The following example illustrates how tag inheritance works.

Compute resources
- Cluster1 with tag cluster-1
- Cluster2 with tag cluster-2
- Cluster3 with tag cluster-3

Storage profiles
- Profile 1 for Datastorecluster1 with tag cluster-1
- Profile 2 for Datastorecluster2 with tag cluster-2
- Profile 3 for Datastorecluster3 with tag cluster-3

Cloud Account
vSphere cloud account with all three of the tags: cluster-1, cluster-2, and cluster-3

While consolidating tags on storage profiles and compute resources, Cloud Assembly also takes into account the cloud account level tags. Hence, the effective tags on all the storage profiles and computes are cluster-1, cluster-2 and cluster-3 and this is why when we provide any of these tags as shown in the preceding example, all the storage profiles and computes become eligible for placement and the machine can land on any of the compute hosts.

As a best practice, to minimize unexpected results and tag clutter, any given tag should be applied only at the cloud account level if that tag is an appropriate capability for all subordinate compute and storage resources.

Using constraint tags in Cloud Assembly

Tags added to projects and cloud templates function as constraint tags when they are used to match capability tags on infrastructure resources, profiles and cloud zones. In the case of cloud templates, Cloud Assembly uses this matching functionality to allocate resources for deployments.

Cloud Assembly enables you to use constraint tags in two primary ways. The first way is when configuring projects and images. You can use tags as constraints to associate resources with the project or image. The second is in cloud templates where tags specified as constraints are used to select resources for deployments. Constraints applied in both of these ways are merged in cloud templates to form a set of deployment requirements that define resources available for a deployment.
How constraint tags work on projects

When configuring Cloud Assembly resources, cloud administrators can apply constraint tags on projects. In this way, administrators can apply governance constraints directly at the project level. All constraints added at this level are applied to every cloud template requested for the applicable project, and these constraint tags take precedence over other tags.

If constraint tags on the project conflict with constraint tags on the cloud template, the project tags take precedence, thus allowing the cloud administrator to enforce governance rules. For example, if the cloud administrators creates a `location:london` tag on the project, but a developer places a `location:boston` tag on the cloud template, the former will take precedence and the resource is deployed to infrastructure containing the `location:london` tag.

You can apply up to three constraints on projects. Project constraints can be hard or soft. By default they are hard. Hard constraints allow you to rigidly enforce deployment restrictions. If one or more hard constraints are not met, the deployment will fail. Soft constraints offer a way to express preferences that will be selected if available, but the deployment won’t fail if soft constraints are not met.

How constraint tags work in cloud templates

In cloud templates, you add constraint tags to resources as YAML code to match the appropriate capability tags that your cloud administrator created on resources, cloud zones and storage and network profiles. In addition, there are other more complex options for implementing constraint tags. For example, you can use a variable to populate one or more tags on a request. This enables you to specify one or more of the tags at request time.

Create constraint tags by using the `tag` label under a constraint heading in the cloud template YAML code. Constraint tags from projects are added to the constraint tags created in cloud templates.

Cloud Assembly supports a simple string formatting to make using constraints easier in YAML files:

```
[!]tag_key[:tag_value][:hard|:soft]
```

By default Cloud Assembly creates a positive constraint with hard enforcement. The tag value is optional, though recommended, as in the rest of the application.

The following WordPress with MySQL example shows YAML constraint tags that specific location information for compute resources.

```yaml
name: "wordpressWithMySql"
components:
  mysql:
    type: "Compute"
    data:
      name: "mysql"
      # ... skipped lines ...
  wordpress:
    type: "Compute"
```
For more information about how to work with cloud templates, see **Part 3: Design and deploy the example Cloud Assembly template**.

**How hard and soft constraints work in projects and cloud templates**

Constraints in both projects and cloud templates can be hard or soft. The preceding code snippet shows examples of hard and soft constraints. By default all constraints are hard. Hard constraints allow you to rigidly enforce deployment restrictions. If one or more hard constraints are not met, the deployment will fail. Soft constraints express preferences that apply if available, but they won't cause a deployment to fail if not met.

If you have a series of hard and soft constraints on a specific resource type, the soft constraints can also serve as tie breakers. That is, if multiple resources meet a hard constraint, the soft constraints are used to select the actual resource used in the deployment.

For example you can specify up to three constraints on a project in any combination of network, storage and extensibility items. Also, you can select whether each constraint is hard or soft. Let's say that you create a hard storage constraint with a tag of `location:boston`. If no storage in the project matches this constraint, any related deployment will fail.

**Standard tags**

Cloud Assembly applies standard tags to some deployments to support analysis, monitoring, and grouping of deployed resources.

Standard tags are unique within Cloud Assembly. Unlike other tags, users do not work with them during deployment configuration, and no constraints are applied. These tags are applied automatically during provisioning on AWS, Azure, and vSphere deployments. These tags are stored as system custom properties, and they are added to deployments after provisioning.

The list of standard tags appears below.

**Table 4-4. Standard tags**

<table>
<thead>
<tr>
<th>Description</th>
<th>Tag</th>
</tr>
</thead>
<tbody>
<tr>
<td>Organization</td>
<td>org:orgID</td>
</tr>
<tr>
<td>Project</td>
<td>project:projectID</td>
</tr>
<tr>
<td>Requester</td>
<td>requester:username</td>
</tr>
<tr>
<td>Deployment</td>
<td>deployment:deploymentID</td>
</tr>
</tbody>
</table>
Table 4-4. Standard tags (continued)

<table>
<thead>
<tr>
<th>Description</th>
<th>Tag</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cloud template reference (if applicable)</td>
<td>blueprint:blueprintID</td>
</tr>
<tr>
<td>Component name in blueprint</td>
<td>blueprintResourceName:CloudMachine_1</td>
</tr>
<tr>
<td>Placement Constraints: applied in blueprint, request parameters, or via IT policy</td>
<td>constraints$key:value:soft</td>
</tr>
<tr>
<td>Cloud Account</td>
<td>cloudAccount:accountID</td>
</tr>
<tr>
<td>Zone or profile, if applicable</td>
<td>zone:zoneID, networkProfile:profileID,</td>
</tr>
<tr>
<td></td>
<td>storageProfile:profileID</td>
</tr>
</tbody>
</table>

How Cloud Assembly processes tags

In Cloud Assembly, tags express capabilities and constraints that determine how and where resources are allocated to provisioned deployments during the provisioning process.

Cloud Assembly uses a specific order and hierarchy of operations in resolving tags to create provisioned deployments. Understanding the basics of this process will help you to implement tags efficiently to create predictable deployments.

The following list summarizes the high level operations and sequence that Cloud Assembly uses to resolve tags and define a deployment:

- Cloud zones are filtered by several criteria, including availability and profiles; tags in profiles for the region the zone belongs to are matched at this point.

- Zone and compute capability tags are used to filter the remaining cloud zones by hard constraints.

- Out of the filtered zones, priority is used to select a cloud zone. If there are several cloud zones with the same priority, they are sorted by matching soft constraints, using a combination of the cloud zone and compute capabilities.

- After a cloud zone is selected, a host is selected by matching a series of filters, including hard & soft constraints as expressed in cloud templates.

How do I set up a simple tagging structure

This topic describes a basic approach and options for a logical Cloud Assembly tagging strategy. You can use these examples as a starting point for an actual deployment, or you can devise a different strategy that better suits your needs.

Typically, the cloud administrator is the primary individual responsible for creating and maintaining tags.

This topic refers to the WordPress use case described elsewhere in the Cloud Assembly documentation to illustrate how tags can be added to some key items. It also describes possible alternatives and extensions to the tagging examples that appear in the WordPress use case.
See Tutorial: Setting up and testing multi-cloud infrastructure and deployments in Cloud Assembly for more information about the WordPress use case.

The WordPress use case describes how to place tags on cloud zones and storage and network profiles. These profiles are like organized packages of resources. Tags placed on profiles apply to all items within the profile. You can also create and place tags on storage resources and individual network items as well as on compute resources, but these tags apply only to the specific resources on which they are placed. When setting up tags, it is usually best to begin by tagging compute resources, and then you can add tags to profiles and cloud zones later. Also, you use these tags to filter the list of compute resources for a cloud zone.

For example, while you can place tags on storage profiles as shown in this use case, you can also place tags on individual storage policies, data stores, and storage accounts. Tags on these resources enable you to exercise finer control over how storage resources are deployed. During processing in preparation for deployment, these tags are resolved as a next level of processing after the profile tags.

As an example of how you might configure a typical customer scenario, you could place a tag of region: eastern on a network profile. This tag would apply to all resources within that profile. Then you could place a tag of networktype:pci on a pci network resource within the profile. A cloud template with constraints of eastern and pci would create deployments that use this pci network for the eastern region.

Procedure

1  Tag your compute infrastructure resources in a logical and appropriate manner.

   It is particularly important that you tag compute resources in a logical manner so that you can find them using the search function on the Compute tab of the Create Cloud Zone page. Using this search function, you can quickly filter the compute resources associated with a cloud zone. If you tag Storage and Networks at the profile level, you may not need to tag individual storage and network resources.

   a  Select Resources > Compute to view the compute resources that have been imported for your Cloud Assembly instance.

   b  Select each compute resource as appropriate and click Tags to add a tag to the resource. You can add more than one tag to each resource if appropriate.

   c  Repeat the previous step for storage and network resources as appropriate.

2  Create cloud zone and network profile capability tags.

   You can use the same tags for both cloud zones and network profiles, or you can create unique tags for each item if that makes more sense for your implementation.

   In network profiles, you can place tags on the entire profile as well as on subnets within the profile. Tags applied at the profile level apply to all components, such as subnets, within that profile. Tags on subnets apply only to the specific subnet on which they are placed. During tag processing, the profile level tags take precedence over the subnet level tags.
See 2. Add cloud zones for information about adding tags to cloud zones or network profiles.

In this example we create three simple tags that appear throughout the use case documentation for Cloud Assembly cloud zone and network profile tags. These tags identify the environment for the profile components.

- zone: test
- zone: dev
- zone: prod

3 Create storage profile tags for your storage components.

Typically, storage tags identify the performance level of storage items, such as tier1 or tier2, or they identify the nature of storage items, such as pci.

See 6. Add storage profiles for information about adding tags to storage profiles.

- usage: general
- usage: fast

Results

After you create a basic tagging structure, you can begin working with it and add or edit tags as appropriate to refine and extend your tagging capabilities.

**How to work with resources in vRealize Automation Cloud**

A cloud administrator can review vRealize Automation Cloud resources that are exposed through data collection.

The cloud administrator can label resources with capability tags to affect where vRealize Automation Cloud cloud templates are deployed.

**Compute resources in vRealize Automation Cloud**

A cloud administrator can review compute resources that are exposed through data collection.

The cloud administrator might choose to apply tags directly to the resources to label capabilities for matching purposes in vRealize Automation Cloud provisioning.

**Network resources in vRealize Automation Cloud**

In vRealize Automation Cloud, cloud administrators can view and edit the network resources that have been data-collected from the cloud accounts and integrations that are mapped to your project.
After you add a cloud account to your Cloud Assembly infrastructure, for example by using the **Infrastructure > Connections > Cloud Accounts** menu sequence, data collection discovers the cloud account’s network and security information. That information is then available for to use in networks, network profiles, and other definitions.

Networks are the IP-specific components of an available network domain or transport zone. If you're an Amazon Web Services or Microsoft Azure user, think of networks as subnets.

You can display information about the networks in your project by using the **Infrastructure > Resources > Networks** page.

The Cloud Assembly **Networks** page contains information such as:

- Networks and load balancers that are defined externally in the network domain of your cloud account, for example in vCenter, NSX-T, or Amazon Web Services.
- Networks and load balancers that have been deployed by the cloud administrator.
- IP ranges and other network characteristics that have been defined or modified by your cloud administrator.
- External IPAM provider IP ranges for a particular address space in an provider-specific external IPAM integration.

For more information about networks, see the following information, signpost help for various settings on the **Networks** page, and Learn more about network profiles in vRealize Automation Cloud.

**Networks**

You can view and edit networks and their characteristics, for example to add tags or remove support for public IP access. You can also manage network settings such as DNS, CIDR, gateway, and tag values. You can also define new, and manage existing, IP ranges within a network.

For existing networks you can change the IP range and tag settings by selecting the network’s checkbox and selecting either **Manage IP Ranges** or **Tags**. Otherwise you can select the network itself to edit its information.

Tags provide a means for matching appropriate networks, and optionally network profiles, to network components in cloud templates. Network tags are applied to every instance of that network, regardless of any network profiles in which the network may reside. Networks can be instanced into any number of network profiles. Regardless of network profile residency, a network tag is associated with that network wherever the network is used. Network tag matching occurs with other components in the cloud template after the cloud template has been matched with one or more network profiles.

For global networks, existing and public networks are supported for NSX-T global manager and local manager cloud accounts and the vCenter cloud accounts that are associated to the local managers. Local manager representation of stretched networks is defined within a transport zone. The transport zone is an NSX-T local manager construct that defines the span of NSX-T networks for vCenter Server hosts and clusters.
Cloud Assembly enumerates, or data collects, existing and public networks. You can create a global network by adding an existing or public network on an NSX-T global manager. The global network can then be consumed by all the associated local managers. Global networks can span one, all, or a subset of the associated local managers.

You can provision a machine on a global network by using a static IP assignment. DHCP is not supported.

You can create the following types of global networks on a global manager:

1. **Overlay** - an overlay network is associated with a Tier-0/Tier-1 local manager and automatically stretches to all the sites connected to the Tier-0/Tier-1 local manager. For each local manager, the default overlay transport zone is used.

2. **VLAN** - a VLAN network applies to a single local manager and the transport zone can be manually selected.

Global networks are listed on the **Infrastructure > Resources** page with all the cloud accounts that they apply to.

The following Day 2 operations are supported for global networks:

- Reconfigure a network in a cloud template definition from a global network to a local network and vice versa.
- Scale-out/scale-in of machines on global networks.

For more information about using networks in cloud templates, see More about network resources in vRealize Automation Cloud cloud templates.

**IP Ranges**

Use an IP range to define or make changes to the start and end IP address for a particular network in your organization. You can display and manage IP ranges for listed networks. If the network is managed by an external IPAM provider, you can manage IP ranges in connection with the associated IPAM integration point.

Click **New IP Range** to add an additional IP range to the network. You can specify an internal IP range, or if there is a valid IPAM integration available you can specify an external IP range.

You cannot include the default gateway in an IP range. The subnet IP range cannot include the subnet gateway value.

If you are using an external IPAM integration for a particular IPAM provider, you can use the external IP range to select an IP range from an available external IPAM integration point. This process is described within the context of an overall external IPAM integration workflow at Configure a network and network profile to use external IPAM for an existing network in vRealize Automation Cloud.
vRealize Automation Cloud allows you to apply and manage an IP address range across multiple vSphere and NSX networks. Shared IP range support is provided for both internal and external IPAM. You can set a single IP range on an NSX stretch network such that machines on that network can use IP addresses that are assigned from the single IP address even if they are deployed to different vCenters.

**IP Addresses**

You can see the IP addresses that are currently used by your organization and display their status, for example available or allocated. The IP addresses that are displayed are either IP addresses that are managed internally by vRealize Automation Cloud or IP addresses that are designated for deployments that contain an external IPAM provider integration. External IPAM providers manage their own IP address allocation.

If the network is managed internally by vRealize Automation Cloud, and not by an external IPAM provider, you can also release IP addresses.

When using internal IPAM and releasing IP addresses, for example after deleting a machine that had been using the IP addresses or clicking **Release IP address** for a selected network, there is a wait period between when the unused addresses are released and when they become available for reuse. The wait period, or release timeout period, allows the DNS cache to clear. The IP addresses can then be allocated to a new machine. By default, the IP address release wait period is 30 minutes. You can change the wait period by clicking the **Settings** option in the upper right corner of the **Networks** page and changing the **Release timeout** value.

- During the release timeout period, relevant IP addresses are listed as released. When the release timeout period has expired, they are listed as available.
- The system checks every 5 minutes for newly released IP addresses, so even if the release timeout value is 1 minute it can take between 1 and 6 minutes for released IP addresses to become available, depending on when the last check was run. The 5 minute checking interval applies to all values other than 0.
- If you set the release timeout value to 0, IP addresses are released immediately and become available immediately.
- The release timeout value applies to all cloud accounts in the organization.

**Load Balancers**

You can manage information about available load balancers for the account/region cloud accounts in your organization. You can open and display the configured settings for each available load balancer. You can also add and remove tags for a load balancer.

For more information about using load balancers in cloud templates, see **More about load balancer resources in vRealize Automation Cloud cloud templates**.

**Network Domains**

The network domains list contains related and non-overlapping networks.
Security resources in vRealize Automation Cloud

After you add a cloud account in Cloud Assembly, data collection discovers the cloud account's network and security information and makes that information available for use in network profiles and other options.

Security groups and firewall rules support network isolation. Security groups are data-collected. Firewall rules are not data-collected.

Using the Infrastructure > Resources > Security menu sequence, you can view on-demand security groups that have been created in Cloud Assembly cloud template designs and existing security groups that were created in source applications, such as NSX-T and Amazon Web Services. Available security groups are exposed by the data collection process.

You can use a tag to match the machine interface (NIC) with a security group in a cloud template definition or in a network profile. You can view the available security groups and add or remove tags for selected security groups. A cloud template author can assign one or more security groups to a machine NIC to control security for the deployment.

In the cloud template design the securityGroupType parameter in the security group resource is specified as existing for an existing security group or new for an on-demand security group.

Existing security groups

Existing security groups are displayed and classified in the Origin column as Discovered.

Existing security groups from the underlying cloud account endpoint, such as NSX-V, NSX-T, or Amazon Web Services applications, are available for use.

A cloud administrator can assign one or more tags to an existing security group to allow it to be used in a cloud template. A cloud template author can use a Cloud.SecurityGroup resource in a cloud template design to allocate an existing security group by using tag constraints. An existing security group requires at least one constraint tag be specified in the security resource in the cloud template design.

If you edit an existing security group directly in the source application, such as in the source NSX application rather than in Cloud Assembly, the updates are not visible in Cloud Assembly until you data collection runs and data collects the associated cloud account or integration point from within Cloud Assembly. Data collection runs automatically ever 10 minutes.

Existing security groups are supported for NSX-T global manager and local manager cloud accounts and the vCenter cloud accounts that are associated to the local managers. Cloud Assembly enumerates, or data collects, existing security groups and attaches them to the machine's network interfaces (NICs). You can create a global security group by adding an existing security group on an NSX-T global manager. The global security group can then be consumed by the associated local managers. Global security groups can span one, all, or a subset of the associated local managers.

- Global existing security groups are supported and enumerated for all defined regions.
Global security groups are listed on the **Infrastructure > Resources** page with all the cloud accounts that they apply to.

You can associate a machine interface (NIC) with an existing global security group directly in a cloud template or in the selected network profile.

The following Day 2 operations are supported for global security groups:

- Security group reconfiguration in a cloud template from a global to a local security group and vice versa.
- Scale-out/scale-in of machines that are associated with global security groups.

**On-demand security groups**

On-demand security groups that you create in Cloud Assembly, either in a cloud template or in a network profile, are displayed and classified in the **Origin** column as **Managed by Cloud Assembly**. On-demand security groups that you create as part of a network profile are internally classified as an isolation security group with pre-configured firewall rules and are not added to a cloud template design as a security group resource. On-demand security groups that you create in a cloud template design, and that can contain express firewall rules, are added as part of a security group resource that is classified as **new**.

**Note** You can create firewall rules for on-demand security groups for NSX-V and NSX-T directly in a security group resource in cloud template design code. The **Applied To** column does not contain security groups that are classified or managed by an NSX Distributed Firewall (DFW). Firewall rules that apply to applications are for east/west DFW traffic. Some firewall rules can only be managed in the source application and cannot be edited in Cloud Assembly. For example, ethernet, emergency, infrastructure, and environment rules are managed in NSX-T.

On-demand security groups are not currently supported for NSX-T global manager cloud accounts.

**Learn more**

For more information about using security groups in network profiles, see Learn more about network profiles in vRealize Automation Cloud.

For information about defining firewall rules, see Using security group settings in network profiles and cloud template designs in vRealize Automation Cloud.

For more information about using security groups in a cloud template, see More about security group and tag resources in vRealize Automation Cloud cloud templates.

For cloud template design code samples that contain security groups, see Networks, security resources, and load balancers in vRealize Automation Cloud.

**Storage resources in vRealize Automation Cloud**

A cloud administrator can work with storage resources and their capabilities, which are discovered through vRealize Automation Cloud data collection from associated cloud accounts.
Storage resource capabilities are exposed through tags that typically originate at the source cloud account. A cloud administrator can choose to apply additional tags directly to storage resources though, using Cloud Assembly. The additional tags might label a specific capability for matching purposes at provisioning time.

vRealize Automation Cloud supports standard disk and first class disk capabilities. First class disk is available for vSphere only.

- **What can I do with standard disk storage in vRealize Automation Cloud**
- **What can I do with first class disk storage in vRealize Automation Cloud**

Capabilities on storage resources become visible as part of the definition of a Cloud Assembly storage profile. See Learn more about storage profiles in vRealize Automation Cloud.

First class disks that have been data-collected appear on the Volumes resource page. See Volume resources in vRealize Automation Cloud.

**Machine resources in vRealize Automation Cloud**

In vRealize Automation Cloud, all users can review machine resources that are exposed through data collection.

Information about all the machines in your projects is available. You can list only your machines or specify filters to control the display of listed machines.

Unmanaged machines that are associated to cloud accounts in your projects appear in this list, as do managed machines. The Origin column indicates the machine status.

- **Discovered** - machines that have not yet been onboarded.
- **Deployed** - machines that have been onboarded or provisioned from vRealize Automation Cloud and are considered to be managed machines.

Other machine information, such as custom properties and memory, is also collected and displayed.

You can use a workload onboarding plan to bring unmanaged machines into vRealize Automation Cloud management.

Disconnected machine NICs are not listed because vRealize Automation Cloud requires the presence of the network switch or subnet information to enumerate the ethernet card. For example, if you have removed a machine NIC from a deployment, the NIC is not listed.

For information about using onboarding plans to bring unmanaged machines into vRealize Automation Cloud management, see What are onboarding plans in Cloud Assembly.

**Volume resources in vRealize Automation Cloud**

In vRealize Automation Cloud, all users can review volume resources.

Cloud Assembly displays volumes or logical drives that originate from two sources:

- **Volumes discovered through data collection of source cloud accounts**
Volumes associated with workloads provisioned by Cloud Assembly

You can review capacity and capabilities according to volume or logical drive. The list also exposes capability tags that originated at the source cloud account or were added in Cloud Assembly itself. The volume’s status as a first class disk is also noted. For information about first class disk storage volumes, see What can I do with first class disk storage in vRealize Automation Cloud.

Learn more about resources in Cloud Assembly

Cloud Assembly can expose additional information around data-collected resources, such as pricing cards.

How does data collection work in vRealize Automation Cloud

After initial data collection, resource data collection occurs automatically every 10 minutes. The data collection interval is not configurable and you cannot manually initiate data collection.

You can discover information about resource data collection and image synchronization for an existing cloud account in the Status section of its page. Do so by selecting Infrastructure > Connections > Cloud Accounts and then clicking Open on the existing cloud account of your choice.

You can open an existing cloud account and see its associated endpoint version in the Status section of its page. If the associated endpoint has been upgraded, the new endpoint version is discovered during data collection and reflected in the Status section on the cloud account’s page.

Resource data collection

Data collection occurs every 10 minutes. Each cloud account displays when its data collection last completed.
Image data collection

Image synchronization occurs every 24 hours. You can initiate image synchronization for some cloud account types. To initiate image synchronization, open the cloud account (Infrastructure > Cloud Accounts then select and open the existing cloud account) and click the Sync Images button. There is no image synchronization option for NSX cloud accounts.

Note Images are internally classified as either public or private. Public images are shared and are not specific to a particular cloud subscription or organization. Private images are not shared and are specific to a specific subscription. Public and private images are automatically synchronized every 24 hours. An option on the cloud account page allows you to trigger synchronization for private images.

The cloud account page displays when image synchronization was last completed.

![Image status](image)

To facilitate fault tolerance and high availability in deployments, each NSX-T data center endpoint represents a cluster of three NSX managers. For related information, see Create an NSX-T cloud account in vRealize Automation Cloud.

Cloud accounts and onboarding plans

When you create a cloud account, all machines that are associated to it are data-collected and then displayed on the Infrastructure > Resources > Machines page. If the cloud account has machines that were deployed outside of Cloud Assembly, you can use an onboarding plan to allow Cloud Assembly to manage the machine deployments.

For information about adding cloud accounts, see Adding cloud accounts to Cloud Assembly.

For information about onboarding unmanaged machines, see What are onboarding plans in Cloud Assembly.

How to use the Insights dashboard to monitor resource capacity and notify project owners in vRealize Automation Cloud

A cloud administrator can monitor and manage infrastructure resources and deployment optimizations within each cloud zone. By visualizing real-time insights, and reviewing suggested actions for the resources you support, you can proactively help project owners manage their resource capacity and optimize their deployments.
You can use the **Insights** dashboard to explore metric data for the resources and deployments in cloud zones within the projects that you manage. Use that information, provided from a combination of vRealize Automation Cloud and your integrated vRealize Operations Cloud application, to make any needed adjustments to memory, CPUs, and so on, or share that information with your team so that they can be better informed and make any needed adjustments.

The Insights dashboard enables you to contact some or all of the project owners who have deployments in the cloud zone that contain reclaimable resource capacity. The cloud zone insights display reclaimable capacity for projects and deployments.

Contacted project owners see notification on their deployment's **Alerts** page. The notification contains their name and the name of (and link to) each deployment that can be optimized.

The **Insights** dashboard is available for vSphere and VMware Cloud on AWS cloud zones, provided that the cloud accounts are configured in both vRealize Automation Cloud and vRealize Operations Cloud and are being monitored in vRealize Operations Cloud.

**Prerequisites**

- Verify that you have vRealize Automation Cloud cloud administrator credentials and have enabled HTTPS access on port 443. See Credentials required for working with cloud accounts in vRealize Automation Cloud.
- Verify that you have the vRealize Automation Cloud cloud administrator user role. See What are the vRealize Automation Cloud user roles.

**About vRealize Operations Cloud and the collected resource capacity metrics**

vRealize Operations Cloud collects capacity metrics for the same infrastructure resources that you and the teams that you support use in vRealize Automation Cloud. By integrating vRealize Automation Cloud with vRealize Operations Cloud, the vRealize Operations Cloud metric data is made available and displayed for each managed project in an **Insights** dashboard within each cloud zone.

Project data is parsed to the vRealize Automation Cloud dashboard from the integrated vRealize Operations Cloud application. The Insights dashboard displays the following information:

- CPU utilization percentage relative to capacity
- Memory utilization percentage relative to capacity
- Storage utilization percentage relative to capacity
- Calculated CPU and memory demand history and projected demand
- Option to contact owners of some or all of the deployments in a cloud zone that can be optimized by reclaiming resources, for example by resizing or deleting machines. Optimization data is calculated in the order of days.

The Insights dashboard is available for vSphere resources.

A trend widget displays the compute components of a cloud zone (such as clusters and hosts), their CPU GHz usage relative to CPU capacity, and their memory GB usage relative to memory capacity.

Information about the roles that are required to use alerts is available at Custom user roles in vRealize Automation Cloud.

For related information, see Resource management and deployment optimization using vRealize Operations Cloud metrics in vRealize Automation Cloud.

**Procedure**

Open a cloud zone to discover its capacity metrics and optionally fetch information about project deployments that can be optimized. Data is collected and supplied by the associated vRealize Operations Cloud application.

1. From Cloud Assembly, click **Infrastructure > Configure > Cloud Zones** and select a cloud zone.

2. Click the **Insights** tab and examine the insights dashboard.

   The following example displays CPU, memory, and storage capacity information for the resources that are used by projects in the cloud zone.
To notify the project owner of any deployments that can be optimized, click **Contact Owner** in the **Projects** section. Notifications appear on the **Alerts** tab page.
4. To fetch optimization information about each deployment for the project, click **Proceed**.

If the project contains deployments that can be optimized, that information is conveyed to the project owner on the Cloud Assembly **Alerts** tab.

5. A message appears indicating the number of deployments that can be optimized.
Notification information about these resources and deployments is available to the project owner on the Cloud Assembly Alerts tab. For this example, that notification information includes the name of, and a link to, each deployment that can be optimized, as shown in the following example:
Next steps

Use the information that you have obtained from the Insights dashboard to make any needed adjustments to the resources that you manage. Open the Alerts page to obtain additional information, suggested actions, and links to deployments that can be optimized. See How to use Alerts to manage resource capacity, performance, and availability in vRealize Automation Cloud.

How to use Alerts to manage resource capacity, performance, and availability in vRealize Automation Cloud

As a cloud administrator, you need to know when vRealize Automation Cloud capacity, performance, and availability are becoming problematic so that you can proactively react before users begin to run out of resources.
You can display a range of alerts provided by the associated vRealize Operations Cloud application. Alerts are available for vSphere and VMware Cloud on AWS resource objects. Use information in alerts to modify the resources and deployments that you manage, or share that information with your team so they can modify objects that they manage.

**Note** To examine and act on project deployments that you should consider optimizing, see *How to use Alerts to optimize deployments in vRealize Automation Cloud*.

Alerts are currently available for vSphere and VMware Cloud on AWS resource objects only. The **Alerts** tab is only available if access to vRealize Operations Cloud is configured.

The vRealize Automation Cloud alerts threshold values are set in vRealize Operations Cloud. Some vRealize Automation Cloud alerts are currently predefined. Alerts notifications are also set in vRealize Operations Cloud. For information about setting alert definitions and configuring notifications, see *Configuring Alerts and Actions* in vRealize Operations Cloud **product documentation**.

**Prerequisites**

- Verify that you have vRealize Automation Cloud cloud administrator credentials and have enabled HTTPS access on port 443. See Credentials required for working with cloud accounts in vRealize Automation Cloud.
- Verify that you have the vRealize Automation Cloud cloud administrator user role. See What are the vRealize Automation Cloud user roles.
- Configure the vRealize Automation Cloud adapter in vRealize Operations Cloud. See Integrate vRealize Automation Cloud Service with vRealize Operations Cloud Service and Application Integration in vRealize Operations Cloud **product documentation**.
- Configure the roles that are need to manage alerts. See Custom user roles in vRealize Automation Cloud.

Role capabilities include:

- Cloud administrators can manage cloud zone alerts.
- Project administrators can manage project alerts.
- Service broker administrators can manage deployment alerts.
About vRealize Operations Cloud and resource alerts

vRealize Operations Cloud collects health, usage, and other metrics for the same infrastructure resources and deployments that you manage in vRealize Automation Cloud. By integrating vRealize Automation Cloud with vRealize Operations Cloud, that monitored data is made available to you in vRealize Automation Cloud by using the Alerts tab in the Cloud Assembly main menu.

The alerts data provided by vRealize Operations Cloud includes health and risk threshold concerns for cloud templates, deployments, organizations, and projects. It also contains information about deployments that can be optimized, based on the owner being contacted by an action taken on the cloud zone Insights tab. See How to use the Insights dashboard to monitor resource capacity and notify project owners in vRealize Automation Cloud.

Alert details for each deployment include:

- Project name
- Deployment name (and link to the deployment) that contain resources that can be optimized
- Suggested actions
- Potential cost savings from reclamation and optimization
- Total number of virtual CPUs used by the deployment
- Total amount of RAM memory used by the deployment
- Total amount of storage used by the deployment
- Virtual machines in the deployment that are recommended for reclamation and optimization, including resource name, idle machines, powered off machines, oversized and undersized machines, underutilized machines, and machine snapshots

By using the Contact project owners option on the cloud zone Insights dashboard, you can see a summary of all projects that have reclaimable capacity (CPU, memory, and storage) in the cloud zone and provide an alert to some or all of the project owners.

Procedure

You can display alerts threshold information about the resources that you manage by using filtering options on the Alerts page. Alerts data is supplied by your associated vRealize Operations Cloud application. Suggested actions are provided for each alert.
You can also select a deployment from the **Deployments to review** section to open and optimize that deployment. See *How to use Alerts to optimize deployments in vRealize Automation Cloud*.

1. From within the Cloud Assembly service, click the **Alerts** tab in the main menu.

2. To control how alerts are displayed, experiment with the available filters. For example, select the **Resources** option from the filters drop-down menu.

3. To display alerts and suggested actions for those alerts, use quick filter options in the selector panel.
   - Display alerts about cloud zone resources.
vRealize Operations Cloud can monitor time remaining, capacity remaining, reclaimable capacity, and so on.

- Display alerts about virtual machine resources.
The majority of virtual machine alerts pertain to on/off status, latency, and so on.

- Display alerts about deployment resources.
The deployment alerts pertain to reclaimable resources and right-sizing.

- Display alerts about project resources.
The project alerts pertain to reclaimable resources and allocation limits.

4 Explore other filter types and their quick filtering options to further control the list of alerts.
   - Use the **Impacts** quick filters of health, risk, and efficiency.
   - Use the **Severity** quick filters of critical, immediate, warning, and information.
   - Use the **Status** quick filters of active, cancelled, and dismissed.
   - Use the **Subtype** filters of availability, performance, and capacity.
   - Use the **Type** quick filters of application, hardware, infrastructure, storage, and network.

5 Take any needed actions based on alerts data and suggestions.
Next steps
To learn about other actions that are available, see How to use Alerts to optimize deployments in vRealize Automation Cloud.

You can also display capacity Insights for cloud zone-based resources in projects that you manage. For information about using vRealize Operations Cloud- supplied Insights data in vRealize Automation Cloud, see How to use the Insights dashboard to monitor resource capacity and notify project owners in vRealize Automation Cloud.

How to use Alerts to optimize deployments in vRealize Automation Cloud
As a cloud administrator or project owner, you can monitor and manage machine resources for best possible optimization by using data that is obtained from vRealize Operations Cloud and displayed in vRealize Automation Cloud.

When you connect vRealize Automation Cloud with vRealize Operations Cloud, you can access data-collected information about resources in the projects that you manage. Alerts and insights data is provided to inform you of various concerns about the projects that you manage, and provide an easy means to communicate optimization suggestions and supporting data collected from vRealize Operations Cloud to project owners easily and efficiently without ever leaving the vRealize Automation Cloud application. For example, you can see reclaimable resource capacity, with specific cost savings, for each deployment in a cloud zone. Where a cloud zone contains multiple deployments that can be optimized, you can notify some or all of the project and deployment owners.

Deployment optimization alerts can be generated from the Insights dashboard. See How to use the Insights dashboard to monitor resource capacity and notify project owners in vRealize Automation Cloud. You can contact project owners so that they can open a named deployment to be optimized from a link provided on the Alerts page. As well, project owners can open their deployments directly and use the Optimize tab to perform available optimization tasks. Actions that a project owner can take include reclaiming resources by deleting non-critical deployments, and stopping further provisioning within a cloud zone.

Note To learn about other resource remediation actions that you can take, see How to use Alerts to manage resource capacity, performance, and availability in vRealize Automation Cloud.

Prerequisites
See How to use Alerts to manage resource capacity, performance, and availability in vRealize Automation Cloud for needed credentials and configuration information for accessing vRealize Operations Cloud data in vRealize Automation Cloud.

To request that project owners be alerted of deployments that are optimizeable, see How to use the Insights dashboard to monitor resource capacity and notify project owners in vRealize Automation Cloud.
About

Each deployment contains an **Optimize** tab. The following optimization parameters are available:

- Machines that can be rightsized - Displays information and actions for oversized and undersized machines in the deployment, along with optimization cost savings.
- Machines that are under-utilized - Displays information and actions for idle or powered off machines in the deployment, along with optimization cost savings.
- Machine snapshots - Displays information and actions for machine snapshots if machines in the deployment contain snapshots, along with optimization cost savings.

As an administrator, you can notify project owners that they have deployments to optimize. Notifications appear on the **Alerts** tab in Cloud Assembly.

The **Alerts** tab is only available if access to vRealize Operations Cloud is configured. Project owners can open and optimize their deployments to respond to alerts.

Procedure

You can display alerts threshold information about the resources that you manage by using filtering options on the **Alerts** page. Alerts data is supplied by your associated vRealize Operations Cloud application. Suggested actions are provided for each alert. In this example the project owner opens their deployment from a link supplied on an alert notification. The deployment's **Optimize** tab displays available machine parameters to optimize.

1. As a project owner or administrator, click the **Alerts** tab in the main menu.

2. Find an alert that contains information about a deployment that can be optimized and click the deployment name from **Deployments to review** to open that deployment and display its **Optimize** tab.
3 When the deployment opens, click the **Optimize** tab.

4 If there are underutilized machines, examine and act on idle and powered off machines. You can power off or delete an undersized deployment.
If there are machines that can be rightsized, examine and act on any oversized and undersized machines in the deployment.

If one or more of the machines in the deployment contains a snapshot, you can delete or export each snapshot.

When you are finished, confirm that the deployment has been optimized to your satisfaction and close the deployment.

Next steps
To learn about other actions that are available, see How to use Alerts to manage resource capacity, performance, and availability in vRealize Automation Cloud.

You can also display capacity insights for cloud zone-based resources in projects that you manage. For information about using vRealize Operations Cloud-supplied insights data in vRealize Automation Cloud, see How to use the Insights dashboard to monitor resource capacity and notify project owners in vRealize Automation Cloud.

What can I do with standard disk storage in vRealize Automation Cloud

Standard disks can be persistent or non-persistent.

vRealize Automation Cloud supports two categories of storage – standard disk and first class disk. First class disk is only available for vSphere.

- vSphere
  
vSphere supports dependent (default), independent persistent, and independent non-persistent standard disks. For related information, see What can I do with persistent disk storage in vRealize Automation Cloud.

  When you delete a virtual machine, its dependent and independent non-persistent disks are also deleted.

  When you delete a virtual machine, its independent persistent disks are not deleted.

  You can create a snapshot of dependent and independent non-persistent disks. You cannot create a snapshot of an independent persistent disk.

- Amazon Web Services (AWS) EBS
  
  You can attach an EBS volume to an AWS compute instance or detach an EBS volume from an AWS compute instance.

  When you delete a virtual machine, its attached EBS volume is detached but not deleted.

- Microsoft Azure VHD
  
  Attached disks are always persistent.

  When you delete a virtual machine, you specify whether to remove its attached storage disks.

- Google Cloud Platform (GCP)
Attached disks are always persistent.

Persistent disks are located independently from your virtual machine instances, so you can detach or move persistent disks to keep your data even after you delete your instances.

When you delete a virtual machine, its attached disk is detached but not deleted.

For related information, see Learn more about storage profiles in vRealize Automation Cloud.

**What can I do with persistent disk storage in vRealize Automation Cloud**

Persistent disks preserve valuable data from accidental deletion.

In a cloud template, under a volume, you can add the `persistent: true` property to have the disk survive Cloud Assembly or Service Broker deletions. Persistent disks aren't removed during deployment deletion nor Day 2 delete or remove disk operations.

Because of that, persistent disks can remain in your infrastructure even after a deployment deletion or disk deletion. To remove them, you can use the following techniques:

- Explicitly pass the purge flag as a query parameter using the DELETE API.
- Delete them directly from your cloud endpoint.

Note that there is no Cloud Assembly or Service Broker user interface for removing them.

**What can I do with first class disk storage in vRealize Automation Cloud**

A first class disk (FCD) provides storage life-cycle management on virtual disks as a disk-as-a service or as EBS-like disk storage that allows you to create and manage disks independently of vSphere virtual machines.

vRealize Automation Cloud supports two categories of storage disks – standard disk and first class disk. First class disk functionality is supported for vSphere only. vRealize Automation Cloud currently provides first class disk functionality as an API-only capability.

A first class disk has its own life-cycle management capabilities that operate independently from a VM. One way that a first class disk differs from an independent persistent disk, is that you can use a first class disk to create and manage snapshots independent of a VM.

You can create a new vRealize Automation Cloud storage profile to support either first class disk capabilities or standard disk capabilities. See Learn more about storage profiles in vRealize Automation Cloud and Storage resources in vRealize Automation Cloud.

You can also add a `Cloud.vsphere.Disk` first class disk element in your vRealize Automation Cloud cloud templates and deployments to support vSphere first class disks. First class disks that have been data-collected appear on the Volumes resource page. See Volume resources in vRealize Automation Cloud.

In vCenter, first class disks are also referred to as Improved Virtual Disks (IVD) or managed virtual disks.
Capabilities
Using vRealize Automation Cloud API capabilities, you can:
- Create, list, and delete a first class disk.
- Resize a first class disk.
- Attach and detach a first class disk.
- Create and manage first class disk snapshots.
- Convert an existing standard disk to first class disk

API use case documentation for creating and managing first class disk (FCD) storage is available at code.vmware.com in the Creating and Using a First Class Disk section of the vRealize Automation Cloud Programming Guide. For related API documentation for FCD, see the First Class Disk (FCD) section of the Virtual Disk Development Kit Programming Guide.

Considerations and limitations
First class disk considerations and limitations currently include:
- First class disk is available for vSphere VMs only.
- vSphere 6.7 Update 2 or later is required to use first class disks.
- Provisioning first class disks on datastore clusters is not supported.
- Volume multi-attach is not supported for first class disks.
- First class disks with snapshots cannot be resized.
- First class disks with snapshots cannot be deleted.
- First class disk snapshot hierarchy can only be constructed by using the createdAt API option.
- The minimum VM hardware version required to attach a first class disk is vmx-13 (ESX 6.5 compatible).

Configuring Multi-provider tenant resources with vRealize Automation Cloud
In multi-tenancy environments, customers can manage allocation of resources on a per-tenant basis using Virtual Private Zones (VPZs).

vRealize Automation Cloud supports multi-organization configurations for customers that require such a configuration and that purchase support for this feature through Cloud Provider Navigator. Cloud Provider Navigator serves as the central management platform for managing VMware Cloud services. To use Cloud Provider Navigator, you must set up a commit contract for vRealize Automation Cloud. After this contract is signed, you will receive an email with the
appropriate onboarding link. This link enables you to set up your master organization and start creating tenants for vRealize Automation Cloud. Then you can create Virtual Private Zones within vRealize Automation Cloud and allocate infrastructure to these zones that can be assigned to tenants.


**How do I create a Virtual Private Zone for vRealize Automation Cloud**

Provider administrators can create a Virtual Private Zone (VPZ) to allocate infrastructure resources to tenants in a multi-organization vRealize Automation Cloud environment. Administrators can also use VPZ’s to control resource allocation in single tenant deployments.

You can use Virtual Private Zones to allocate resources such as images, networks, and storage resources. VPZs function much as cloud zone on a per tenant basis but they are designed specifically for use with multi tenant deployments. For any given project, you can use either cloud zones or VPZ's but not both. Also, there is a one to one relationship between VPZ's and tenants. That is, a VPZ can be assigned to only one tenant at a time.

**Note** You configure image and flavor mappings for a VPZ on the Tenant Management page.

You can create a VPZ with or without NSX. If you create a zone without NSX, there are limits regarding NSX-related functionality on vSphere endpoints.

- Security (groups, firewall)
- Network components (NAT)

**Prerequisites**

- Create tenant administrators as appropriate for your tenant configuration.
- If you want to use NSX, you must create an appropriate NSX cloud account in your provider organization.

**Procedure**

1. Select **Infrastructure > Configure > Virtual Private Zones**

   The VPZ page shows all existing zones and enables you to create zones.
2 Click **New Virtual Private Zone**.

There are four selections on the left side of the page that you can use to configure summary information and infrastructure components for the zone.

3 Enter Summary information for the new zone.

   a Add a Name and Description.
   
   b Select an Account to which the zone applies.
   
   c Select the Placement Policy.

Placement policy drives host selection for deployments within the specified cloud zone.

- Default - Distributes compute resources across clusters and hosts randomly. This selection works at an individual machine level. For example, all machines in a particular deployment are distributed randomly across the available clusters and hosts that satisfy the requirements.
- binpack - Places compute resources on the most loaded host that has enough available resources to run the given compute.
- spread - Provisions deployment compute resources to the cluster or host with the least number of virtual machines. For vSphere, Distributed Resource Scheduler (DRS) distributes the virtual machines across the hosts. For example, all requested machines in a deployment are placed on the same cluster, but the next deployment might select another vSphere cluster depending on the current load.
Select the Compute resource for the zone.

Add compute resources as appropriate for the cloud zone. Initially, the filter selection is Include all Compute and the following list shows all available compute resources, and they are allocated to the applicable zone. You have two additional options for adding compute resources to a cloud zone.

- Manually select compute - Select this menu item if you want to select compute resources manually from the list below. After you select them, click Add Compute to add the resources to the zone.
- Dynamically include compute by tags - Select this menu item if you want to select compute resource to be added to the zone based on tags. All compute resources are shown until you add appropriate tags. You can select or enter one or more tags in the Include compute with these tags option.

For either compute selection, you can remove one or more of the compute resources shown on the page by selecting the box to the right and clicking Remove.

Enter or select tags as appropriate.

Select Storage on the left menu and select the Storage policy and other storage configurations for the zone.

On the left menu, select Network and define the networks and, optionally, a network policy to use with this zone. You can also configure load balancers and security groups for selected network policies.

<table>
<thead>
<tr>
<th>Network policies</th>
<th>If configured, select a network policy to use with this zone to enforce an isolation policy for outbound and private networks.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Select an isolation policy if desired.</td>
</tr>
<tr>
<td></td>
<td>Select a Tier-0 logical router and an Edge cluster if desired.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Network</th>
<th>All existing networks associated with this VPZ appear in the table on the Networks tab.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Click Add Network to see all networks associated with the selected region. add a network for use with this zone.</td>
</tr>
<tr>
<td></td>
<td>Select a network and click Tags to add one or more tags to the specified network.</td>
</tr>
<tr>
<td></td>
<td>Select Manage IP Ranges to specify the IP Range through which users can access this network.</td>
</tr>
<tr>
<td></td>
<td>If applicable, click the Network Policies tab and select an isolation policy.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Load Balancers</th>
<th>Click Add Load Balancer to configure load balancers for the account/region cloud accounts.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Security Groups</td>
<td>Click Add Security Group to use security groups to apply firewall rules to provisioned machines.</td>
</tr>
</tbody>
</table>
Results

The Virtual Private Zone is created with the specified resource allocations.

What to do next

Cloud administrators can associate the VPZ with a project.

1. In Cloud Assembly, select Administration > Projects
2. Select the Provisioning tab.
3. Click Add Zone and choose Add Virtual Private Zone.
4. Select the desired VPZ from the list.
5. You can set the provision priority and limits on the number of instances, the amount of memory available and the number of CPUs available.
6. Click Add.

Manage Virtual Private Zone configuration for vRealize Automation Cloud tenants

Provider administrators can manage Virtual Private Zones (VPZs) within Cloud Assembly to control infrastructure resource allocation on a per tenant basis. Using the Tenant Management page, administrators can view tenants and VPZ zones and enable or disable VPZs for tenants.

By default, Virtual Private Zones are not allocated to any tenants. You must allocate VPZ’s on this page in order to use them with your tenants.

When initially created, VPZ’s are enabled by default. An enabled VPZ is ready to be allocated and used with the specified tenant. When VPZ’s are disabled, they cannot be used for provisioning or allocated to a tenant. A VPZ can be disabled but still allocated for a tenant.

When a provider administrator navigates to the Tenant Management page, the page shows all available tenants and the administrator can select one. After a tenant is selected, the page shows VPZs currently allocated for that tenant, if any. The administrator can use this page to allocate VPZs to the selected tenant.

When a VPZ is allocated, tenant administrators can add it to their projects, and it becomes available for provisioning by tenant users. After a VPZ is allocated to one tenant, it can be allocated to another tenant.

After a VPZ is enabled, it is ready for use within the specified tenant. Provider administrators can disable VPZ’s to facilitate maintenance or tenant re-configuration, and they can provide notification to users of the disablement. If you want to make a VPZ unavailable to a tenant on a more permanent basis, you can de-allocate it. If an existing VPZ is de-allocated from a tenant for some reason, it cannot be used to create deployments from that tenant.

Prerequisites

- Set up multi-tenancy and create Virtual Private Zones as appropriate for your deployment.
Configure global image and flavor mappings for the VPZ and tenant configuration using the image mapping and flavor mapping menu selections on the left side of the Tenant Management page in Cloud Assembly. See Create global image and flavor mapping for vRealize Automation Cloud tenants.

You can override these global assignments now or later using the tenant specific image and flavor mapping selections at the top of the Tenant Management page. See Configure tenant specific image and flavor mappings for vRealize Automation Cloud.

Procedure

1. In Cloud Assembly select Manage Tenants.
   The Tenant Management page shows all tenants configured for the administrator's organization in a card view.

2. Click on a tenant to select it.

3. Click the infrastructure management tab to see all allocated VPZ's for the tenant.

4. Select Allocate Virtual Private Zone to open a dialog that shows all zones not currently allocated to tenants. allocate the zone to a tenant.

5. Select one or more zones on the dialog and click Allocate To Tenant.

What to do next

After VPZs are allocated, tenant administrators can assign them to projects.

Provider administrators can use the card view of tenants to monitor and manager status of VPZs.

- If you want to disable a tenant, click Disable on the card for the tenant.
- To enable a tenant, click Enable on the card for the tenant.
- If you want to de-allocate a tenant, click Deallocate on the card for that tenant.

Create global image and flavor mapping for vRealize Automation Cloud tenants

Provider administrators can select or create global image and flavor mappings that can be assigned to vRealize Automation Cloud tenants.

Global image and flavor mapping enables you to quickly set up mappings that apply to multiple tenants. You can also quickly update these mappings. The tenant management page also enables you to create tenant specific image and flavor mappings that can override the default configurations.

Note Image and flavor mappings configured on the Tenant Management page apply only to tenants as configured and are not applicable to the broader provider organization.

Prerequisites
Procedure

1. In Cloud Assembly select Manage Tenants.
   The Tenant Management page shows all tenants configured for the administrator's organization in a card view.

2. Select Image Mapping on the Tenant Management page left menu.
   The Image Mapping page displays all image currently configured for tenants in Cloud Assembly and indicates whether the mappings are global or associated with a specific tenant.
Select **Add Image Mapping** to add an image mapping for use with tenants.

- Select the Account/Region to which the image mapping will apply.
- Enter a name for the image mapping and select the specific image instance or version to which it relates.
- Enter any desired constraint tags.
- Select the scope for the image mapping. The scope can be either All tenants, or global, or you can select a specific tenant to which the image mapping will apply.

If desired, you can use a cloud configuration script to define custom OS characteristics for deployments.

For example, based on whether you are deploying a cloud template to a public or private cloud, you can apply specific user permissions, OS permissions, or other conditions to the image. A cloud configuration script adheres to a **cloud-init** format for Linux-based images or a **cloudbase-init** format for Windows-based images. See Learn more about image mappings in vRealize Automation Cloud for more information.

Click **Create** to create the image mapping.

Select **Add Flavor Mapping** to add a flavor mapping for use with tenants.

- Select the Account/Region to which the flavor mapping will apply.
- Enter a name for the flavor mapping you are creating.
c Select the Size parameters for the flavor mapping you are creating.
   You can specify the number of processors and the amount of memory for this flavor.

d Select the scope for the flavor mapping. The scope can be either All tenants, or global, or you can select a specific tenant to which the flavor mapping will apply. All tenants applies to all tenants in the provider administrator's organization.

7 Click Create to create the flavor mapping.

Results
After you create global mappings, these mapping will show up on the Flavor Mapping or Tenant Mapping tabs on the Tenant Management page for applicable tenants.

What to do next
You can edit or delete global image and flavor mappings on this page. To edit a mapping select it and make the desired changes.

Configure tenant specific image and flavor mappings for vRealize Automation Cloud
Cloud Assembly enables you to configure global image and flavor mappings that are available to all Virtual Private Zones (VPZs) within your organization. Alternatively, you can override the global settings and configure tenant specific image and flavor mappings as appropriate for your deployments.

Typically, a cloud administrator configures global image and flavor mappings using the left navigation links on the Tenant Management page, and these mappings apply across the board for all of your tenants. In some cases, you may want to create custom, tenant-specific, image and flavor mappings for specific tenants and the Tenant Management page supports this option.

Image and flavor mapping are shown on their respective tabs on the Tenant Management page. Click on any of the existing image and flavor mappings to edit them. To delete an image or flavor mapping, select the mapping and then click Delete.

Prerequisites
- Enable multi-tenancy and configure tenants for your deployment.
- Create appropriate VPZs.

Procedure
1 Select Tenant Management on the Cloud Assembly main menu.
2 Select the tenant for which you wish to configure custom image or flavor mapping.
3 Select the Image Mapping link on the top of the page, then click Add Image Mapping.
   The Create Image Mapping dialog appears.
4. Ensure that the Account/Region specified is correct and add a name for the mapping in the **Image Name** text box.

5. Select the underlying machine image to use in the **Image** drop-down.

6. Add constraint tags if applicable for your image usage.

7. Select the appropriate **Scope** for the image.
   - Click the Available for this tenant only radio button if you want this image mapping to be available only for use by the selected tenant.
   - Click the Shared Across tenants radio button if you want this image mapping to be available for use by other tenants.

8. Click **Create** to save the image mapping as configured.

9. Select the Flavor Mapping link at the top of the page and then click **Add Flavor Mapping** to create a flavor mapping.

   The Create Flavor Mapping dialog appears.

10. Ensure that the Account/Region specified is correct and add a name for the mapping in the **Name** text box.

11. Specify the flavor CPU and memory settings in the **Value** field.

12. Select the appropriate **Scope** for the image.
   - Click the Available for this tenant only radio button if you want this image mapping to be available only for use by the selected tenant.
   - Click the Shared Across tenants radio button if you want this image mapping to be available for use by other tenants.

13. Click **Create** to save the flavor mapping as configured.

**Results**

Tenant-specific image and flavor mappings are configured as specified.

**Create extensibility subscriptions for providers or tenants**

Provider and tenant administrators can create extensibility subscriptions to access vRealize Orchestrator workflows. vRealize Orchestrator workflows are triggered based on events if there is a subscription for some event topics which corresponds to a particular lifecycle phase of the application.

The characteristics of an extensibility subscription differ depending on whether the subscription was created by a provider administrator or a tenant administrator.

- The Tenant administrator can create a subscription but cannot specify organization scope. That subscription will be activated on events triggered by tenant only.
The Provider administrator can create a subscription and specify provider scope. The subscription will behave just like tenant subscription or non multi-tenant environment. It will be activated based on events coming from the Provider.

The Provider can create a subscription and specify the tenant scope. The subscription is activated based on events coming from any Tenant. It is not activated by events coming from Provider.

Subscriptions trigger vRealize Orchestrator workflows based on specific events. They do not invoke extensibility actions. Currently only a single vRealize Orchestrator instance is supported for any particular provider organization. For more information about events, event topics, and subscriptions see Extensibility terminology.

Prerequisites

Configure tenants and virtual private zones as appropriate for your deployment.

Procedure

1. In vRealize Automation Cloud, navigate to the Subscriptions page and click New Subscription.
2. Enter a Name and Description for the subscription.
3. Make sure the Enable Subscription radio button is On.
   You can leave this button in the Off position if you don't want the subscription to be immediately active.
4. If you are a provider administrator, select the appropriate Organization Scope.
   The organization scope options are either provider or tenant. If you select tenant, then the project scope is any project and cannot be changed. If you select provider, you can specify the project scope using the selection at the bottom of the Subscriptions page.
5. Select the Event Topic to which you wish to subscribe.
6. Select one or more workflows.

Results

Providers and tenants can view the returned events for a specific deployment on the Events page in Cloud Assembly. The displayed results depend on your role and the organization scope.

- If organization scope is Provider, then providers will see events based on their actions in same provider organization.
- If organization scope is Tenant, tenants will see the events, but the provider cannot see them. Events always live in the organization of the publisher.

1. Select Extensibility > Events in Cloud Assembly.
2. In the Events page Search box, enter the deployment ID for which you wish to view events.

The page displays events that match the search criteria.
Working with legacy Virtual Private Zones in newer versions of vRealize Automation Cloud

The configuration options for VPZs have changed in Cloud Assembly. You can update or work with legacy Virtual Private Zones in current versions of vRealize Automation Cloud.

In the initial release of Virtual Private Zones (VPZs), users configured image and flavor mappings within VPZs using the VPZ page in Cloud Assembly. In newer versions of vRealize Automation Cloud, users create image and flavor mappings on a per-tenant basis, which increases efficiency and configuration flexibility especially in deployments with large numbers of tenants. While there is no way to migrate these older legacy VPZs, there are several options for using them with newer versions of vRealize Automation Cloud.

The first, and most flexible, option is to delete the legacy image and flavor mappings from the older VPZs and re-configure them with new mappings created on the Tenant Management page.

1. Select **Infrastructure > Configure > Virtual Private Zones** to open the VPZ page.
2. Select Image Mapping to view the existing mapping.
3. Select mappings and click to delete them.
4. Select Image Mapping to view the existing mapping.
5. Select mappings and click to delete them.
6. Close the VPZ page.
7. Select Tenant Mapping and create select a global mapping for the applicable tenants or create a tenant specific mapping.

Alternatively, you can use legacy vPZs with newer versions of vRA in their existing configuration. The legacy image and flavor mappings still function as configured, but their configuration options are read only on the VPZ page. This options offers less flexibility than the first option.
Adding and managing Cloud Assembly projects

Projects control who has access to Cloud Assembly cloud templates and where the templates are deployed. You use projects to organize and govern what your users can do and to what cloud zones they can deploy cloud templates in your cloud infrastructure.

Cloud administrators set up the projects, to which they can add users and cloud zones. Anyone who creates and deploys cloud templates must be a member of at least one project.

This chapter includes the following topics:

- How do I add a project for my Cloud Assembly development team
- Learn more about Cloud Assembly projects

How do I add a project for my Cloud Assembly development team

You create a project to which you add members and cloud zones so that the project members can deploy their cloud templates to the associated zones. As the Cloud Assembly administrator, you create a project for a development team. You can then assign a project administrator or you can operate as the project administrator.

When you create a cloud template, you first select the project to associate it with. The project must exist before you can create the cloud template.
Ensure that your projects support the business needs of the development team.

- Does the project provide the resources that support the team's goals. For an example of how the infrastructure resources and a project support a cloud template, see Tutorial: Setting up and testing multi-cloud infrastructure and deployments in Cloud Assembly.

- Do your project members require or expect their deployments to be shared or private. Shared deployments are available to all the project members on the Deployments tab, not only the deploying member. You can change the deployment sharing state at anytime.

When you share the deployment with project members, the members can run the same day 2 action. To manage the ability of members to run day 2 actions, you can create day 2 policies in Service Broker. The policies apply to Cloud Assembly and Service Broker deployments.

To learn more about the day 2 policies, see How do I entitle deployment users to day 2 actions using policies

This procedure is based on creating an initial project that includes only the basic configurations. As your development team creates and deploys their cloud templates, you might modify to the project. You can add constraints, custom properties, and other options to improve deployment efficiencies. See the articles available in Learn more about Cloud Assembly projects.

Prerequisites

- Verify that you configured the cloud zones. See Chapter 4 Building your Cloud Assembly resource infrastructure.

- Verify that you configured the mappings and profiles for the regions that include as cloud zones for this project. See Chapter 4 Building your Cloud Assembly resource infrastructure.

- Verify that you have the necessary permissions to perform this task. See What are the vRealize Automation Cloud user roles.

- Determine who you are designating as the project administrator. To understand what the project administrator can do in Cloud Assembly, see What are the vRealize Automation Cloud user roles.

- If you are adding Active Directory groups to projects, verify that you configured Active Directory groups for your organization. See How do I enable Active Directory groups in vRealize Automation Cloud for projects. If the groups are not synchronized, they are not available when you try to add them to a project.

Procedure

1. Select Infrastructure > Administration > Projects, and click New Project.

2. Enter the project name.
Click the Users tab.

a. To make deployments by project members accessible only to the requesting user, turn off Deployment sharing. To ensure that you can assign the ownership of a deployment to another member of the project, verify that the Deployment sharing is turned on.

b. Add users with assigned roles.

Click the Provisioning tab and add one or more cloud zones.

Add any cloud zones and virtual private zones that contain the resources that support the cloud templates deployed by the project users.

For each zone, you can set a zone priority and you can limit the amount of resources that the project can use. The possible limits include the number of instances, memory, and CPUs. For vSphere cloud zones only, you can configure storage limits for deployed resources that are based on vSphere VM templates. The storage limits are evaluated with you request deployment and when you make changes using the resize disk, resize boot disk, remove disk, and the update count actions. These storage limits do not apply to other resource types such as AWS, Microsoft Azure, or Google Cloud Platform.

As you add each zone and apply limits, don't limit the project resources so narrowly that the members cannot deploy their cloud templates.

When your users submit a deployment request, the zones are evaluated to determine which zones have the resources to support the deployment. If more than one zone supports the deployment, then the priority is evaluated and the workload is placed on the one with the higher priority, which is the lowest integer.

If the workloads requested for this project take more than two hours to deploy, enter a longer value for the Timeout.

The default value is two hours.

Click Create.

To test your project with the project cloud zones, click Test Configuration on the Projects page.

The simulation runs a standardized hypothetical deployment test against the project cloud zone resources. If it fails, you can review the details and correct your resource configuration.

What to do next

Get started with cloud templates. See Chapter 6 Designing your Cloud Assembly deployments.

Learn more about Cloud Assembly projects

Projects are the connector between cloud templates and resources. The more you understand about how they work and how you can make them work for you, the more effective your Cloud Assembly development and deployment process will be.
Using Cloud Assembly project tags and custom properties

As an administrator, you can add project-level governance constraints or custom properties when the requirements of the project are different from the Cloud Assembly cloud templates. In addition to constraint tags, you can add resource tags that are added to deployed resources during the provisioning process so that you can manage the resources.

What are project resource tags

A project resource tag operates as an standardized identifying tag that you can use to manage the deployed resources and ensure compliance.

The resource tags defined in a project are added to all component resources deployed as part of that project. You can then use the standard tagging to manage the resources using other applications, for example, monitor spending cost using CloudHealth, and, importantly, to ensure compliance.

For example, as a cloud administrator, you want to use an application like CloudHealth to manage costs. You add the costCenter:eu-cc-1234 tag to a project dedicated to developing a European Union human resources tool. When the project team deploys from this project, the tag is added to the deployed resources. You then configure the costing tool to identify and manage the resources that include this tag. Other projects with other cost centers would have alternative values to go with the key.

What are project constraint tags

A project constraint operates as a governance definition. It is a key:value tag that defines what resources the deployment request consumes or avoids in the project cloud zones.

The deployment process looks for tags for the networks and storage that match the project constraints, and deploys based on matching tags.

The extensibility constraint is used to specify which vRealize Orchestrator integrated instance to use for extensibility workflows.

Consider the following formats when you configure project constraints.

- **key:value** and **key:value:hard**. Use this tag, in either format, when the cloud template must be provisioned on resources with the matching capability tag. The deployment process fails when no matching tag is found. For example, a cloud template deployed by the members of a project must be provisioned on a PCI-compliant network. You use security:pci. If no networks are found in the project cloud zones, the deployment fails, ensuring no insecure deployments.

- **key:value:soft**. Use this tag when you prefer a matching resource, but you want the deployment process to proceed without failing and can accept resources where the tag does not match. For example, you prefer that the project members deploy their cloud templates to a less expensive storage, but you do not want storage availability to interfere with their ability to deploy. You use tier:silver:soft. If there is no storage tagged tier:silver in the project cloud zones, the cloud template still deploys on other storage resources.
- **!key:value**. Use this tag, with hard or soft, when you want to avoid deploying to resources with a matching tag.

Importantly, the project constraint tags have a higher priority than the cloud template constraint tags and override them at deployment time. If you have a cloud template where this must never happen, you can use the `failOnConstraintMergeConflict:true` in the template. For example, if your project has a network `loc:london` constraint, but the cloud template is `loc:mumbai`, but rather than the project location taking precedence, you want the deployment to fail with a constraint conflict message, you add a property similar to the following sample.

```yaml
constraints:
  - tag: 'loc:mumbai'
failOnConstraintMergeConflict:true
```

### How might I use project custom properties

You can use a project custom property for reporting, to trigger and populate extensibility actions and workflow, and to override cloud template level properties.

Adding a custom property to a deployment allows you to use the value in the user interface or to retrieve it using the API so that you can generate reports.

Extensibility can also use a custom property for an extensibility subscription. For more information about extensibility, see [Extending and automating application life cycles with extensibility](Extending and automating application life cycles with extensibility).

A cloud template might have a particular property value that you want to change for a project. You can provide an alternative name and value as a custom property.

You can also encrypt the property value so that neither you nor your users can see the value that is included in the deployment. For example, you can encrypt a password that all users in the project use, but that you do not want visible. After you encrypt the value and save the project, you cannot unmask or replace the value. If you clear the Encrypted check box, the value is removed. You must re-enter a value.

### How do project-level placement policies affect resource allocation in vRealize Automation Cloud

As an administrator, you can define the placement policy for projects where more than one cloud zone is eligible as the deployment target zone. For example, you might have a project where you want to deploy cloud templates based on the set priority. Or you might want to balance the deployed resources across multiple zones based on which one has the best VM to host ratio.
Allocation considerations
For a default or spread placement policy.
- If the deploying user has permission to manage cloud accounts that are in maintenance mode, the allocation process can select a cloud account that is in maintenance mode because the user might need to run a test deployment before closing the maintenance window.
- If the user does not have permission to manage cloud accounts, then the cloud accounts that are in maintenance mode are filtered out of the allocation process.
- Hosts that are in maintenance mode are counted as part of the spread ratio. To exclude a host in maintenance from the ratio calculation, you must set the power state to off.

For a spread policy.
- Ratios are calculated based on hosts. The hosts can be standalone or part of a cluster.
- If a standalone host is powered off, it is not counted as part of the ratio.
- If a host that is part of a cluster is powered off, the powered off state is not reflected in the cluster and the host is still considered when calculating the ratio.

How to set the placement policy
If you have multiple cloud zones in a project that are equally eligible as the target for a deployment, the deployment request evaluates where to place them based on how you have the Placement policy configured.

1. Select Infrastructure > Projects and create or select a project.
2. In the project, click the Provisioning tab.
3. Select a policy.

<table>
<thead>
<tr>
<th>Placement policy</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default</td>
<td>Deploys the requested resources to the first cloud zone that matches the requirements. Select Default when you want the workloads deployed in the priority order and don't mind utilizing all the resources on a host. If this option is selected, the VM and Hosts values are not retrieved.</td>
</tr>
<tr>
<td>Spread</td>
<td>Deploys the requested resources to the cloud zone with the smallest number of virtual machines per hosts. Select Spread when you want to distribute the workloads across hosts, utilizing resources broadly across hosts. If this option is selected, the number of VMs and hosts are retrieved from the cloud zone resources and evaluated.</td>
</tr>
</tbody>
</table>

4. Click Save.
Review how the policy is applied

After you configure the project-level placement policy, you can view where the system plans to deploy the cloud template in a provisioning diagram.

1. Select **Design > Cloud Templates** and select or configure a template that uses the project to which you selected a policy.
2. Click **Test**.
3. When the test completes successfully, click **Provisioning Diagram** in the test results.
The diagram will resemble one of the two examples.
<table>
<thead>
<tr>
<th>Policy Type</th>
<th>Provisioning diagram</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default</td>
<td>![Default Diagram]</td>
</tr>
<tr>
<td>Spread</td>
<td>![Spread Diagram]</td>
</tr>
</tbody>
</table>
5 If you are ready to deploy, return to the cloud template and click **Deploy**.

**Placement policy evaluation during the allocation process**

The following diagram helps you understand when the policy is evaluated during the allocation process and when the target zone and host are identified.

![Diagram showing the evaluation process of placement policies during allocation.](image-url)
What are the project prices in Cloud Assembly

The costs available in your Cloud Assembly projects help you manage the resource expenses associated with entire projects. The project also includes the individual deployment costs.

The cost information that you see for a project and for the individual deployments appears after at least one deployment associated with the project is provisioned. The costs are calculated and updated daily so that you can track the cost of a deployment over time. The initial values are based on industry benchmarks.

Cloud administrators can adjust the values to reflect your actual costs.

For more information, see How to use Pricing Cards.

How do Cloud Assembly projects work at deployment time

Projects control user access to the cloud zones and user ownership of the provisioned resources. Whether your are a cloud administrator or a cloud template developer, you must understand how the projects work at deployment time so that you can manage your deployments and troubleshoot any problems.

As a cloud administrator who is setting up projects for various teams, you must understand how projects determine where cloud template components are deployed. This understanding helps you create projects that support cloud template developers and to troubleshoot failed deployments.

When you create a cloud template, you first associate it with a project. At deployment time, the cloud template requirements are evaluated against the project cloud zones to find the best deployment location.

The following workflow illustrates the process.

1. You submit a cloud template deployment request.
2 The project evaluates the template and project requirements, for example, flavor, image, and constraint tags. The requirements are compared to the project cloud zones to locate a zone that supports the requirements.

3 These zones did not have the resources to support the request.

4 This cloud zone supports the request requirements and the template is deployed to this cloud zone account region.
How do I enable Active Directory groups in vRealize Automation Cloud for projects

If a group is not available on the Add Groups page when you are adding users to projects, check the Identity & Access Management page and add the group if it is available. If the group is not listed on the Identity & Access Management page in vRealize Automation Cloud, the group may not be synchronized in your Workspace One Access instance. You can verify that it has been synchronized and then use this procedure to add the group as shown herein.

To add members of an Active Directory group to a project, you must ensure that the group is synchronized with your Workspace One Access instance and that the group is added to the organization.

Prerequisites

If the groups are not synchronized, they are not available when you try to add them to a project. Submit a request to VMware Cloud Services to synchronize your Active Directory groups with your VMware Cloud Services Workspace One Access instance.

Procedure

1. Log in to vRealize Automation Cloud as a user from the same Active Directory domain that you are adding. For example, @mycompany.com
2. In Cloud Assembly, click Identity & Access Management in the header right navigation.
3. Click Enterprise Groups, and then click Assign Roles.
4. Use the search function to find the group that you are adding and select it.
5. Assign an organization role.
6. Click Add Service Access, add one or more services, and select a role for each.
7. Click Assign.

Results

You can now add the Active Directory group to a project.
Designing your Cloud Assembly deployments

Deployments begin with cloud templates, formerly called blueprints, which are encoded specifications that define machines, applications, and services to create on cloud resources by way of Cloud Assembly.

How cloud templates work

Templates can target specific cloud vendors or be cloud agnostic. The cloud zones assigned to your project determine which approach you might take. Check with your cloud administrator so that you know what kind of resources make up your cloud zones.

Cloud Assembly template creation is an infrastructure-as-code process. You start by adding resources in the design canvas. Then, you complete the details using the code editor. The code editor allows you to type code directly or enter values in a form.

Before you create a cloud template

You can create a Cloud Assembly template at any time. To deploy it, however, you first need to Chapter 4 Building your Cloud Assembly resource infrastructure and Chapter 5 Adding and managing Cloud Assembly projects that includes that infrastructure.

Ready to design?

Explore the navigation on the left, or go directly to topics in the following table.

<table>
<thead>
<tr>
<th>Get started</th>
<th>Learn more about cloud template designs and features</th>
<th>More examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Getting started with Cloud Assembly designs</td>
<td>User input in vRealize Automation Cloud requests</td>
<td>Documented Cloud Assembly template example</td>
</tr>
<tr>
<td>Creating bindings and dependencies between resources in Cloud Assembly</td>
<td>Custom naming for deployed resources in Cloud Assembly</td>
<td>Cloud Assembly expressions</td>
</tr>
<tr>
<td>Versioning your Cloud Assembly templates</td>
<td>Reusing a group of properties in Cloud Assembly</td>
<td>Secret Cloud Assembly properties</td>
</tr>
</tbody>
</table>
This chapter includes the following topics:

- Getting started with Cloud Assembly designs
- Getting code completion help in Cloud Assembly
- Creating bindings and dependencies between resources in Cloud Assembly
- Versioning your Cloud Assembly templates
- User input in vRealize Automation Cloud requests
- Reusing a group of properties in Cloud Assembly
- Cloud Assembly resource flags for requests
- Cloud Assembly expressions
- Secret Cloud Assembly properties
- Remote access to a Cloud Assembly deployment
- SCSI disk placement with Cloud Assembly
- Machine initialization in Cloud Assembly
- Machine and disk clusters in Cloud Assembly
- Custom naming for deployed resources in Cloud Assembly
- How to add the SaltStack Config minion in Cloud Assembly designs
- Terraform configurations in Cloud Assembly
- Custom resource types for Cloud Assembly cloud templates
- Cloud Assembly designs that prepare for day 2 changes
Other Cloud Assembly code examples
vRealize Automation Cloud resource property schema
Other ways to create Cloud Assembly templates
Extending and automating application life cycles with extensibility

Getting started with Cloud Assembly designs

You use the design page to create Cloud Assembly template specifications for the machines and applications that you want to provision.

How to use the design page

To create a cloud template from scratch, go to Design > Cloud Templates. Then, click New from > Blank canvas.

1. Locate resources.
2. Drag resources to the canvas.
3. Connect resources.
4. Configure resources by editing the cloud template code.

Selecting and adding resources to the canvas

Resources appear at the left of the design page for selecting and dragging.
Cloud agnostic resources

You can deploy cloud agnostic resources to any cloud vendor. At provisioning time, the deployment uses cloud specific resources that match. For example, if you expect a cloud template to deploy to both AWS and vSphere cloud zones, use cloud agnostic resources.

Cloud vendor resources

Vendor resources, such as those specific to Amazon Web Services, Microsoft Azure, Google Cloud Platform, or VMware vSphere, can only be deployed to matching AWS, Azure, GCP, or vSphere cloud zones. You can add cloud agnostic resources to a cloud template that contains cloud specific resources for a particular vendor. Just be aware of what the project cloud zones support in terms of vendor.

Configuration management resources

Configuration management resources depend on your integrated applications. For example, a Puppet resource can monitor and enforce the configuration of the other resources.

**Connecting resources**

Use the Cloud Assembly design canvas graphical controls to connect resources.

Resources must be compatible for a connection. For example:

- Connecting a load balancer to a cluster of machines.
- Connecting a machine to a network.
- Connecting external storage to a machine.

**Important**  A solid line connector requires that the two resources be deployed in the same cloud zone. If you add conflicting constraints to the resources, deployment might fail.

For example, you can't deploy connected resources where constraint tags force the placement of one to a zone in us-west-1, and the other to a zone in us-east-1.

Solid or dashed arrows only indicate a dependency, not a connection. For more about dependencies, see Creating bindings and dependencies between resources in Cloud Assembly.

To connect, hover over the edge of a resource to reveal the connection bubble. Then, click and drag the bubble to the target resource and release.

In the code editor, additional code for the source resource appears in the target resource code.
In the figure, the SQL machine and private network are connected, so they must be deployed in the same cloud zone.

**Editing cloud template code**

The code editor allows you to type, cut, copy, and paste code directly. If you're uncomfortable editing code, you can click a resource that's already in the design canvas, click the code editor Properties tab, and enter values there. Property values that you enter appear in the code as if you had typed them directly.
Note that you can copy and paste code from one cloud template to another.

**Getting code completion help in Cloud Assembly**

Adding Cloud Assembly resources and connecting them in the canvas only creates starter code. To fully configure them, edit the code.

The code editor allows you to type code directly or enter property values into a form. To help with direct code creation, the Cloud Assembly editor includes syntax completion and error checking features.
### Editor

<table>
<thead>
<tr>
<th>Hints</th>
<th>Example</th>
</tr>
</thead>
</table>
| **Available values** | Cloud Machine:
  type: Cloud Machine
  properties:
    - name: string
    - image: string
    - flavor: string
  constraints: array
  networkType: existing |

| **Allowed properties** | Cloud Machine:
  type: Cloud Machine
  properties:
    - name: string
    - image: string
    - flavor: string
    - constraints: array
    - networkType: string |

| **Child properties** | Cloud Machine:
  type: Cloud Machine
  properties:
    - name: string
    - image: string
    - flavor: string
    - constraints: array
    - networkType: string |

### Syntax errors

Please correct errors in YAML editor before editing in canvas: row 14, column 17.

### Ctrl+F to search

<table>
<thead>
<tr>
<th>Inputs: ()</th>
<th>resources:</th>
</tr>
</thead>
</table>
| Cloud Machine:
  type: Cloud Machine
  properties:
    - name: string
    - image: string
    - flavor: string
    - networks:
      - name: string |
| Cloud Machine:
  type: Cloud Machine
  properties:
    - name: string
    - image: string
    - flavor: string
    - networks:
      - name: string |
Creating bindings and dependencies between resources in Cloud Assembly

When you deploy a Cloud Assembly template, one resource might need another resource to be available first.

**Important** Arrows only indicate a dependency, not a connection. To connect resources so that they communicate, see Getting started with Cloud Assembly designs.

**Explicit dependencies**

Sometimes, a resource needs another to be deployed first. For example, a database server might need to exist first, before an application server can be created and configured to access it.
An explicit dependency sets the build order at deployment time, or for scale in or scale out actions. You can add an explicit dependency using the graphical design canvas or the code editor.

- Design canvas option—draw a connection starting at the dependent resource and ending at the resource to be deployed first.

- Code editor option—add a `dependsOn` property to the dependent resource, and identify the resource to be deployed first.

An explicit dependency creates a solid arrow in the canvas.

```
resources:
  - Application:
    type: Cloud.Machine
    dependsOn:
    - Database
  - Database:
    type: Cloud.Machine
    properties:
      image: ""  
      flavor: ""
```

**Property bindings**

Sometimes, a resource property needs a value found in a property of another resource. For example, a backup server might need the operating system image of the database server that is being backed up, so the database server must exist first.

Also called an implicit dependency, a property binding controls build order by waiting until the needed property is available before deploying the dependent resource. You add a property binding using the code editor.

- Edit the dependent resource, adding a property that identifies the resource and property that must exist first.

A property binding creates a dashed arrow in the canvas.

```
resources:
  - Backup:
    type: Cloud.Machine
    properties:
      image: "$[Database.image]"
      flavor: ""
  - Database:
    type: Cloud.Machine
    properties:
      image: "cocos"
      flavor: ""
```
Versioning your Cloud Assembly templates

As a cloud template developer, you can safely capture a snapshot of a working design before risking further changes.

At deployment time, you can select any of your versions to deploy.

Capturing a cloud template version

From the design page, click Version, and provide a name.

The name must be alphanumeric, with no spaces, and only periods, hyphens, and underscores allowed as special characters.

Restoring an older version

From the design page, click Version History.

On the left, select an older version to inspect it in the canvas and code editor. When you find the version that you want, click Restore. Restoring overwrites the current draft without removing any named versions.

Releasing a version to Service Broker

From the design page, click Version History.

On the left, select a version and release it.

You can't release a Current Draft until you version it.

Reimporting the version in Service Broker

To enable the new version for catalog users, reimport it.

In Service Broker, go to Content & Policies > Content Sources.

In the list of sources, click the source for the project that contains the cloud template with the newly released version.
Click **Save & Import**.

**Comparing cloud template versions**

When changes and versions accumulate, you might want to identify differences among them.

In Cloud Assembly, from the Version History view, select a version, and click **Diff**. Then, from the **Diff against** drop-down, select another version to compare to.

Note that you can toggle between reviewing code differences or visual topology differences.

---

**Figure 6-1. Code Differences**

![Figure 6-1. Code Differences](image1)

**Figure 6-2. Visual Topology Differences**

![Figure 6-2. Visual Topology Differences](image2)
Cloning a cloud template

Although it’s not the same as saving a version, from the design page, **Actions > Clone** makes a copy of the current template for alternative development.

User input in vRealize Automation Cloud requests

As a cloud template designer, you use input parameters so that users can make custom selections at request time.

How inputs work

When users supply inputs, you no longer need to save multiple copies of templates that are only slightly different. In addition, inputs can prepare a template for day 2 operations. See [How to use cloud template inputs for vRealize Automation Cloud day 2 updates](#).

The following inputs show how you might create one cloud template for a MySQL database server, where users can deploy that one template to different cloud resource environments and apply different capacity and credentials each time.

```plaintext
inputs:
wp-size:
  type: string
  enum:
    - small
    - medium
  description: Size of Nodes
  title: Node Size
```

Adding input parameters

Add an inputs section to your template code, where you set the selectable values.

In the following example, machine size, operating system, and number of clustered servers are selectable.
If you're uncomfortable editing code, you can click the code editor **inputs** tab, and enter settings there. The following example shows some inputs for the MySQL database mentioned earlier.

**Referencing input parameters**

Next, in the **resources** section, you reference an input parameter using `${input.property-name}` syntax.
If a property name includes a space, delimit with square brackets and double quotes instead of using dot notation: `${input["property name"]}`

**Important** In cloud template code, you cannot use the word `input` except to indicate an input parameter.

```yaml
resources:
  WebTier:
    type: Cloud.Machine
    properties:
      name: wordpress
      flavor: '${input.wp-size}'
      image: '${input.wp-image}'
      count: '${input.wp-count}'
```

### Optional Inputs

Inputs are usually required and marked with an asterisk. To make an input optional, set an empty default value as shown.

```yaml
owner:
  type: string
  minLength: 0
  maxLength: 30
  title: Owner Name
  description: Account Owner
  default: ''
```

![Testing Basic Form](image-url)
# List of input properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>const</code></td>
<td>Used with oneOf. The real value associated with the friendly title.</td>
</tr>
<tr>
<td><code>default</code></td>
<td>Prepopulated value for the input. The default must be of the correct type. Do not enter a word as the default for an integer.</td>
</tr>
<tr>
<td><code>description</code></td>
<td>User help text for the input.</td>
</tr>
<tr>
<td><code>encrypted</code></td>
<td>Whether to encrypt the input that the user enters, true or false. Passwords are usually encrypted. You can also create encrypted properties that are reusable across multiple cloud templates. See Secret Cloud Assembly properties.</td>
</tr>
<tr>
<td><code>enum</code></td>
<td>A drop-down menu of allowed values. Use the following example as a format guide.</td>
</tr>
</tbody>
</table>
|          | ```
|          | enum:
|          |   - value 1
|          |   - value 2
<p>|          | ``` |
| <code>format</code> | Sets the expected format for the input. For example, (25/04/19) supports date-time. Allows the use of the date picker in Service Broker custom forms. |
| <code>items</code> | Declares items within an array. Supports number, integer, string, Boolean, or object. |
| <code>maxItems</code> | Maximum number of selectable items within an array. |
| <code>maxLength</code> | Maximum number of characters allowed for a string. For example, to limit a field to 25 characters, enter <code>maxLength: 25</code>. |
| <code>maximum</code> | Largest allowed value for a number or integer. |
| <code>minItems</code> | Minimum number of selectable items within an array. |
| <code>minLength</code> | Minimum number of characters allowed for a string. |
| <code>minimum</code> | Smallest allowed value for a number or integer. |
| <code>oneOf</code> | Allows the user input form to display a friendly name (title) for a less friendly value (const). If setting a default value, set the const, not the title. Valid for use with types string, integer, and number. |
| <code>pattern</code> | Allowable characters for string inputs, in regular expression syntax. For example, '[a-z]+' or '[a-z0-9A-2@#$]+' |</p>
<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>properties</td>
<td>Declares the key:value properties block for objects.</td>
</tr>
<tr>
<td>readOnly</td>
<td>Used to provide a form label only.</td>
</tr>
<tr>
<td>title</td>
<td>Used with oneOf. The friendly name for a const value. The title appears on the user input form at deployment time.</td>
</tr>
<tr>
<td>type</td>
<td>Data type of number, integer, string, Boolean, or object.</td>
</tr>
<tr>
<td>writeOnly</td>
<td>Hides keystrokes behind asterisks in the form. Cannot be used with enum. Appears as a password field in Service Broker custom forms.</td>
</tr>
</tbody>
</table>

**Additional examples**

**String with enumeration**

```json
image:
  type: string
  title: Operating System
  description: The operating system version to use.
  enum:
    - ubuntu 16.04
    - ubuntu 18.04
  default: ubuntu 16.04

shell:
  type: string
  title: Default shell
  Description: The default shell that will be configured for the created user.
  enum:
    - /bin/bash
    - /bin/sh
```

**Integer with minimum and maximum**

```json
count:
  type: integer
  title: Machine Count
  description: The number of machines that you want to deploy.
  maximum: 5
  minimum: 1
  default: 1
```

**Array of objects**

```json
tags:
  type: array
  title: Tags
  description: Tags that you want applied to the machines.
  items:
    type: object
```
properties:
  key:
    type: string
    title: Key
  value:
    type: string
    title: Value

String with friendly names

platform:
  type: string
oneOf:
  - title: AWS
    const: platform:aws
  - title: Azure
    const: platform:azure
  - title: vSphere
    const: platform:vsphere
default: platform:aws

String with pattern validation

username:
  type: string
  title: Username
  description: The name for the user that will be created when the machine is provisioned.
  pattern: ^[a-zA-Z]+$

String as password

password:
  type: string
  title: Password
  description: The initial password that will be required to logon to the machine.
  Configured to reset on first login.
  encrypted: true
  writeOnly: true

String as text area

ssh_public_key:
  type: string
  title: SSH public key
  maxLength: 256

Boolean

public_ip:
  type: boolean
  title: Assign public IP address
  description: Choose whether your machine should be internet facing.
  default: false
Date and time calendar selector

leaseDate:
  type: string
  title: Lease Date
  format: date-time

vRealize Orchestrator actions as inputs

In a Cloud Assembly template, vRealize Orchestrator actions can be included as cloud template inputs.

Adding a vRealize Orchestrator action to cloud template inputs

To use vRealize Orchestrator actions as cloud template inputs, follow these guidelines.

1. In the instance of vRealize Orchestrator that is embedded with vRealize Automation Cloud, create an action that does what you want.

   The vRealize Orchestrator action must only include primitive string, integer, number, and boolean types. vRealize Orchestrator types are not supported.

   In this simple example, the vRealize Orchestrator action collects three inputs and returns a hard-coded string.

2. In Cloud Assembly, create or edit a cloud template.

3. In the code editor, click the Inputs tab, and New Cloud Template Input.

4. To add the vRealize Orchestrator action inputs, click the type, and click Constant.

   Separately add each vRealize Orchestrator action input as a new cloud template input.
5. After adding the action inputs, create another new cloud template input, click the type, click **External source**, and click **Select**.

6. In **Action**, search for and select the vRealize Orchestrator action that you created, and click **Save**.
When deploying the cloud template, the vRealize Orchestrator action settings appear in the input form for the requesting user.

**Configurable defaults**

To populate the input form with default values, do one of the following when adding the vRealize Orchestrator action as the external source.

- Manually supply the default property value.
  
  Clear the **Bind** option, and enter the value.

- Use another property value from the inputs already in the cloud template.
Select the **Bind** option, and select a property from the drop-down list.

![Add an existing action](image)

**Adding vRealize Orchestrator enumerated input selections**

To create a vRealize Orchestrator based selection list in an input form, do the following when adding to the cloud template inputs.

1. In vRealize Orchestrator, create an action that maps the values that you want for the list.
2. In Cloud Assembly, when adding the cloud template input, expand **More Options**.
3. For **Pairs**, click **External source**, click **Select**, and add the vRealize Orchestrator action that you created.

**Note** If you also create a default value when adding the property, that default must exactly match one of the enumerated values from the vRealize Orchestrator action.
Reusing a group of properties in Cloud Assembly

When you have Cloud Assembly properties that always appear together, you can assemble them into a property group. You can quickly add a property group to different Cloud Assembly designs, which saves the time of adding the same multiple properties one by one. In addition, you have a single place to maintain or modify the set of properties, which ensures their consistent application.

Only users with the Cloud Assembly Administrator role may create, update, or delete a property group. The administrator can share a property group with an entire organization or limit its use to only within a project.

Caution  A property group might be included in many cloud templates, including ones that are already released to the catalog. Changes to a property group can affect other users.

There are two types of property groups.

- Input property groups in Cloud Assembly
Input property groups gather and apply a consistent set of properties at user request time. Input property groups can include entries for the user to add or select, or they might include read-only values that are needed by the design.

Properties for the user to edit or select can be readable or encrypted. Read-only properties appear on the request form but can't be edited. If you want read-only values to remain totally hidden, use a constant property group instead.

- **Constant property groups in Cloud Assembly**

Constant property groups silently apply known properties. In effect, constant property groups are invisible metadata. They provide values to your Cloud Assembly designs in a way that prevents a requesting user from reading those values or even knowing that they're present. Examples might include license keys or domain account credentials.

The two property group types are handled very differently by Cloud Assembly. When you create a property group, you must first select whether to create inputs or constants. You can't create a blended property group nor convert an existing set of properties and their property group from one type to the other.

**Input property groups in Cloud Assembly**

Cloud Assembly input property groups usually include related settings for the user to enter or select. They might also include read-only values needed by the cloud template design.

**Creating the input property group**

1. Go to **Design > Property Groups**, and click **New Property Group**.
2. Select **Input Values**.
3. Name and describe the new property group.

<table>
<thead>
<tr>
<th>Name</th>
<th>Property group names must be unique within a given organization. Only letters, numbers, and underscores are permitted.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Display Name</td>
<td>Add a heading for the entire group of properties, which appears on the request form.</td>
</tr>
<tr>
<td>Description</td>
<td>Explain what this set of properties is for.</td>
</tr>
<tr>
<td>Scope</td>
<td>Decide whether an administrator may share the property group with the whole organization. Otherwise, only one project can access the property group. Although you can always add or modify properties in the group, the scope is permanent and can't be changed later.</td>
</tr>
<tr>
<td>Project</td>
<td>When the scope is project-only, this project can access the property group.</td>
</tr>
</tbody>
</table>

4. To add a property to the group, click **New Property**.
The panel for adding a new property is very similar to the Inputs tab of the Cloud Assembly design page code editor.

<table>
<thead>
<tr>
<th>Name</th>
<th>Free-form name for the individual property. Only letters, numbers, and underscores are permitted.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Display Name</td>
<td>Add an individual property name to appear on the request form.</td>
</tr>
<tr>
<td>Type</td>
<td>String, Integer, Number, Boolean (T/F), Object, or Array.</td>
</tr>
<tr>
<td>Default Value</td>
<td>Preset value entry that appears in the request form.</td>
</tr>
<tr>
<td>Encrypted</td>
<td>When selected, obscures the value when entering it into the request form and in the subsequent deployment. Encrypted properties can't have a default value.</td>
</tr>
<tr>
<td>Read-only</td>
<td>An uneditable but visible value in the request form. Requires a default.</td>
</tr>
<tr>
<td>More Options</td>
<td>Options that vary according to property type. Expand the drop-down, add any additional settings, and click Create.</td>
</tr>
</tbody>
</table>

In the following example, the property being added represents the operating system image, and the requesting user can select from two.

**Note**  The operating systems shown in the example figure must already be part of the configured Cloud Assembly infrastructure.
5 Add more properties to the group, and click **Save** when finished.
Adding the property group to cloud template inputs

Even for a long list of property inputs, you only need to add the property group to make them all part of the request form.

1. In the cloud template design page, above the editing area on the right, click the Inputs tab.

2. Click New Cloud Template Input.

3. Name and describe the property group.

   | Name | Enter something similar to the property group name that you created earlier. |
   | Display Name | Enter the same heading that you created earlier for the entire group of properties, which appears on the request form. |
   | Type | Select Object. |
   | Object Type | Select Property Group. |

   Property groups list
   Select the property group that you want. Only property groups that are created and available for your project appear. Note that constant property groups don't appear.

4. Click Create.
The process creates cloud template inputs code similar to the following example.

```json
inputs:
  pgmachine:
    type: object
    title: Machine Properties
    $ref: /ref/property-groups/machine
  pgrequester:
    type: object
    title: Requester Details
    $ref: /ref/property-groups/requesterDetails
```

You may also enter code directly into the Cloud Assembly design page, and take advantage of the automatic prompting as you type $ref: /ref/p... in the code editor.

**Binding cloud template resources to the property group**

To make use of property group input values, add bindings under the resource.

Depending on what kind of values are in a property group, you might want to reference them individually. You can enter them separately, by property group name and property name.

```json
resources:
  Cloud_Machine_1:
    type: Cloud.Machine
    properties:
      image: '${input.pgmachine.image}'
      flavor: '${input.pgmachine.flavor}'
```

You can also quickly add an entire set of values to a resource by referencing an entire property group.

```json
resources:
  Cloud_Machine_1:
    type: Cloud.Machine
    properties:
      requester: '${input.pgrequester}'
```

**Completed code**

When you're finished with the inputs and resources, the finished code looks similar to the following example.
Upon deployment request, your property groups appear for the requesting user to complete.

### Property groups in the Service Broker custom form editor

Input property groups appear within the Service Broker custom form interface and are available for customization there. There are no special considerations unique to property groups when customizing them. Service Broker users don't even need to know that the source of the entries is a property group instead of separately created properties.
In a Cloud Assembly input property group, you can add dynamic interaction with vRealize Orchestrator.

**Adding a vRealize Orchestrator action to an input property group**

To add dynamic interaction with vRealize Orchestrator to an input property group, follow these guidelines.

1. In the instance of vRealize Orchestrator that is embedded with vRealize Automation Cloud, create an action that does what you want.

   The vRealize Orchestrator action must only include primitive string, integer, number, and boolean types. vRealize Orchestrator types are not supported.

   In this simple example, the vRealize Orchestrator action collects three inputs and returns a hard-coded string.
2 In Cloud Assembly, start the process of creating or editing an input property group. See Input property groups in Cloud Assembly if necessary.

3 To add the vRealize Orchestrator action inputs to a property group, add new properties, click the type, and click **Constant**.

   Separately add each vRealize Orchestrator action input.

4 After adding the inputs, add a new property, click the type, click **External source**, and click **Select**.
5 In **Action**, search for and select the vRealize Orchestrator action that you created, and click **Save**.

6 Save the property group, and add it to your cloud template. See **Input property groups in Cloud Assembly** if necessary.

   When deploying the cloud template, the vRealize Orchestrator action property group appears in the input form for the requesting user.
Configurable defaults

To populate the input form with default values, do one of the following when adding the vRealize Orchestrator action as the external source.

- Manually supply the default property value.
  
  Clear the **Bind** option, and enter the value.

- Use another property value from the same property group.
  
  Select the **Bind** option, and select a property from the drop-down list.
Adding vRealize Orchestrator enumerated input selections

To create a vRealize Orchestrator based selection list in an input form, do the following when adding to a property group.

1. In vRealize Orchestrator, create an action that maps the values that you want for the list.
2. In Cloud Assembly, when adding a property to the group, expand More Options.
3. For Pairs, click External source, click Select, and add the vRealize Orchestrator action that you created.

Note: If you also create a default value when adding the property, that default must exactly match one of the enumerated values from the vRealize Orchestrator action.

Constant property groups in Cloud Assembly

Cloud Assembly constants allow you to silently apply known key-value pairs to your designs.
How constants work

The key appears in the cloud template code, and the value becomes part of deployments that are based on that cloud template. Constants require the propgroup binding under the resource. The propgroup binding is only used with constant property groups, not input property groups.

Secret properties

If you expect to add an secret property to a property group, create the secret property before proceeding. See Secret Cloud Assembly properties.

Creating the constant property group

1. Go to Design > Property Groups, and click New Property Group.
2. Select Constant Values.
3. Name and describe the new property group.

<table>
<thead>
<tr>
<th>Name</th>
<th>Property group names must be unique within a given organization. Only letters, numbers, and underscores are permitted.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Display Name</td>
<td>Leave blank. No heading appears on the request form.</td>
</tr>
<tr>
<td>Description</td>
<td>Explain what this set of constants is for.</td>
</tr>
<tr>
<td>Scope</td>
<td>Decide whether an administrator may share the property group with the whole organization. Otherwise, only one project can access the property group. Although you can always add or modify properties in the group, the scope is permanent and can't be changed later. Secrets—If you expect to add a secret property to the property group, you must use single project scope. Secret properties are saved only at the project level.</td>
</tr>
<tr>
<td>Project</td>
<td>When the scope is project-only, this project can access the property group.</td>
</tr>
</tbody>
</table>

4. To add a constant property to the group, click New Property.
5. Enter a name that acts as the key, and a description.
6. Select a property type.
7. Enter the constant value that you want, and click Create.
   - String, integer, and number types use direct entry.
   - For a secret string value, select from the list of secret properties for the project.
   - The boolean type uses a selection box to indicate true.
   - For the object or array type, replace null with the value that you want.
8 Add more constants to the group, and click **Save** when finished.

**Binding cloud template resources to the property group**

To silently use constant values within a resource, add `propgroup` bindings under the resource.

You can quickly add an entire set of constants to a resource by referencing the property group itself.

```
resources:
  Cloud_Machine_1:
    type: Cloud.Machine
    properties:
      payerInfo: '${propgroup.payerDetails}'
```
Alternatively, you can add individual constants from the property group to selected parts of your design.

```
resources:
  Cloud_Machine_1:
    type: Cloud.Machine
    properties:
      payerAccount: '${propgroup.payerDetails.payerAccountNumber}'
      payerCost: '${propgroup.payerDetails.payerCostCenter}'
      payerFed: '${propgroup.payerDetails.payerFederal}'
```

Learn more about Cloud Assembly property groups

One Cloud Assembly property group might be included in many cloud templates, which affects how you need to manage property groups.

**Modifying a property group**

Changes to a Cloud Assembly property group affect every cloud template that uses it, including ones that might have already been released to the Service Broker catalog.

The property group list and property group editing pages show the number of cloud templates that include the property group. To see which cloud template would be affected by a change, click the number.

Before modifying a property group, make sure that the change is acceptable to everyone who is creating or updating deployments based on the cloud templates listed.
Deleting a property group

Deleting a property group would cause errors in every cloud template that uses it.

You cannot delete a property group until you manually remove it from all of the cloud templates in which it is included. To remove a property group from a cloud template, open the cloud template in the design canvas.

- Input property groups
  
  Under the inputs tab, select and remove the property group. Alternatively, use the code editor to delete the associated property group in the inputs section of the code.

- Constant property groups
  
  Use the code editor to delete the associated propgroup entry or entries in the resources section of the code.

Cloud Assembly resource flags for requests

Cloud Assembly includes several cloud template settings that adjust how a resource is handled at request time.

Resource flag settings aren't part of the resource object properties schema. For a given resource, you add the flag settings outside of the properties section as shown.

```json
resources:
  Cloud_Machine_1:
    type: Cloud.Machine
    preventDelete: true
    properties:
      image: coreos
      flavor: small
      attachedDisks:
        - source: '${resource.Cloud_Volume_1.id}'
  Cloud_Volume_1:
    type: Cloud.Volume
    properties:
      capacityGb: 1
```
<table>
<thead>
<tr>
<th>Resource Flag</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>allocatePerInstance</td>
<td>When set to true, resource allocation can be customized for each machine in a cluster. The default is false, which allocates resources equally across the cluster, resulting in the same configuration for each machine. In addition, day 2 actions might not be separately possible for individual resources. Per instance allocation allows count.index to correctly apply the configuration for individual machines. For code examples, see Machine and disk clusters in Cloud Assembly.</td>
</tr>
<tr>
<td>createBeforeDelete</td>
<td>Some update actions require that the existing resource be removed and a new one be created. By default, removal is first, which can lead to conditions where the old resource is gone but the new one wasn’t created successfully for some reason. Set this flag to true if you need to make sure that the new resource is successfully created before deleting the previous one.</td>
</tr>
<tr>
<td>createTimeout</td>
<td>The Cloud Assembly default timeout for resource allocate, create, and plan requests is 2 hours (2h). In addition, a project administrator can set a custom default timeout for these requests, applicable throughout the project. This flag lets you override any defaults and set the individual timeout for a specific resource operation. See also updateTimeout and deleteTimeout.</td>
</tr>
<tr>
<td>deleteTimeout</td>
<td>The Cloud Assembly default timeout for delete requests is 2 hours (2h). In addition, a project administrator can set a different default timeout for delete requests, applicable throughout the project. This flag lets you override any defaults and set the individual timeout for a specific resource delete operation. See also updateTimeout and createTimeout.</td>
</tr>
<tr>
<td>dependsOn</td>
<td>This flag identifies an explicit dependency between resources, where one resource must exist before creating the next one. For more information, see Creating bindings and dependencies between resources in Cloud Assembly.</td>
</tr>
<tr>
<td>dependsOnPreviousInstances</td>
<td>When set to true, create cluster resources sequentially. The default is false, which simultaneously creates all resources in a cluster. For example, sequential creation is useful for database clusters where primary and secondary nodes must be created, but secondary node creation needs configuration settings that connect the node to an existing, primary node.</td>
</tr>
<tr>
<td>forceRecreate</td>
<td>Not all update actions require that the existing resource be removed and a new one be created. If you want an update to remove the old resource and create a new one, independent of whether the update would have done so by default, set this flag to true.</td>
</tr>
</tbody>
</table>

Using and Managing VMware Cloud Assembly

VMware, Inc. 438
<table>
<thead>
<tr>
<th>Resource Flag</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ignoreChanges</td>
<td>Users of a resource might reconfigure it, changing the resource from its deployed state. If you want to perform a deployment update but not overwrite the changed resource with the configuration from the cloud template, set this flag to true.</td>
</tr>
<tr>
<td>ignorePropertiesOnUpdate</td>
<td>Users of a resource might customize certain properties, and those properties might be reset to their original cloud template state during an update action. To prevent any properties from being reset by an update action, set this flag to true.</td>
</tr>
<tr>
<td>preventDelete</td>
<td>If you need to protect a resource from subsequent delete requests, set this flag to true.</td>
</tr>
<tr>
<td>recreatePropertiesOnUpdate</td>
<td>Users of a resource might reconfigure properties, changing the resource from its deployed state. During an update, a resource might or might not be recreated. Resources that aren't recreated might remain with properties in changed states. If you want a resource and its properties to be recreated, independent of whether the update would have done so by default, set this flag to true.</td>
</tr>
<tr>
<td>updateTimeout</td>
<td>The Cloud Assembly default timeout for update requests is 2 hours (2h). In addition, a project administrator can set a different default timeout for update requests, applicable throughout the project. This flag lets you override any defaults and set the individual timeout for a specific resource update operation. See also deleteTimeout and createTimeout.</td>
</tr>
</tbody>
</table>

**Cloud Assembly expressions**

For increased flexibility, you can add expressions to cloud template code in Cloud Assembly.

**How expressions work**

Cloud Assembly expressions use the `${expression}` construct, as shown in the following examples.

**Note** Cloud Assembly expressions aren’t the same as regular expressions. See the Cloud Assembly expression syntax for Cloud Assembly.

The following code samples are pruned to show only the important lines. The entire, unedited cloud template appears at the end.
Examples

At deployment time, allow the user to paste in the encrypted key needed for remote access:

```json
inputs:
  sshKey:
    type: string
    maxLength: 500
resources:
  frontend:
    type: Cloud.Machine
    properties:
      remoteAccess:
        authentication: publicPrivateKey
        sshKey: '${input.sshKey}'
```

For deploying to VMware Cloud on AWS, set the folder name to the required name of *Workload*:

```json
inputs:
  environment:
    type: string
    enum:
      - AWS
      - vSphere
      - Azure
      - VMC
      - GCP
    default: vSphere
resources:
  frontend:
    type: Cloud.Machine
    properties:
      folderName: '${input.environment == "VMC" ? "Workload" : ""}'}
```

At deployment time, tag the machine with an all-lowercase *env* tag that matches the selected environment:

```json
inputs:
  environment:
    type: string
    enum:
      - AWS
      - vSphere
      - Azure
      - VMC
      - GCP
    default: vSphere
resources:
  frontend:
    type: Cloud.Machine
    properties:
      constraints:
        - tag: '${"env": to_lower(input.environment)}'
```
Set the number of machines in the front-end cluster to one (small) or two (large). Note that the large cluster is set by process of elimination:

```yaml
inputs:
  envsize:
    type: string
    enum:
      - Small
      - Large
resources:
  frontend:
    type: Cloud.Machine
    properties:
      count: '${input.envsize == "Small" ? 1 : 2}'
```

Attach machines to the same Default network by binding to the property found in the network resource:

```yaml
resources:
  frontend:
    type: Cloud.Machine
    properties:
      networks:
        - network: '${resource.Cloud_Network_1.name}'
  apitier:
    type: Cloud.Machine
    properties:
      networks:
        - network: '${resource.Cloud_Network_1.name}'
  Cloud_Network_1:
    type: Cloud.Network
    properties:
      name: Default
      networkType: existing
```

Encrypt access credentials submitted to the API:

```yaml
resources:
  apitier:
    type: Cloud.Machine
    properties:
      cloudConfig: |
        #cloud-config
        runcmd:
          - export apikey=${base64_encode(input.username:input.password)}
```

Discover the address of the API machine:

```yaml
resources:
  frontend:
    type: Cloud.Machine
    properties:
```
Complete cloud template

```yaml
cloudConfig: |
  runcmd:
    - echo ${resource.apitier.networks[0].address}

apitier:
  type: Cloud.Machine
  properties:
    networks:
      - network: '${resource.Cloud_Network_1.name}'
```

```yaml
inputs:
  environment:
    type: string
    enum:
      - AWS
      - vSphere
      - Azure
      - VMC
      - GCP
    default: vSphere
  sshKey:
    type: string
    maxLength: 500
  envsize:
    type: string
    enum:
      - Small
      - Large
  resources:
    frontend:
      type: Cloud.Machine
      properties:
        folderName: '${input.environment == "VMC" ? "Workload" : ""}''
        image: ubuntu
        flavor: medium
        count: '${input.envsize == "Small" ? 1 : 2}''
        remoteAccess:
          authentication: publicPrivateKey
          sshKey: '${input.sshKey}'
  cloudConfig: |
    packages:
      - nginx
    runcmd:
      - echo ${resource.apitier.networks[0].address}
    constraints:
      - tag: '${"env:" + to_lower(input.environment)}''
    networks:
      - network: '${resource.Cloud_Network_1.name}'
  apitier:
    type: Cloud.Machine
    properties:
      folderName: '${input.environment == "VMC" ? "Workload" : ""}''
      image: ubuntu
```
flavor: small
cloudConfig: |
  #cloud-config
  runcmd:
    - export apikey=${base64_encode(input.username:input.password)}
remoteAccess:
  authentication: publicPrivateKey
  sshKey: '${input.sshKey}'
constraints:
  - tag: '${"env:" + to_lower(input.environment)}'
networks:
  - network: '${resource.Cloud_Network_1.name}'
Cloud_Network_1:
  type: Cloud.Network
  properties:
    name: Default
    networkType: existing
  constraints:
    - tag: '${"env:" + to_lower(input.environment)}'

Cloud Assembly expression syntax

The expression syntax exposes all of the available capabilities of expressions in Cloud Assembly templates.

**Note**  Cloud Assembly expressions aren't the same as regular expressions.

The following syntax is only partly represented in the examples shown in Cloud Assembly expressions.

**Literals**
The following literals are supported:

- Boolean (true or false)
- Integer
- Floating point
- String

Backslash escapes double quote, single quote, and backslash itself:

" is escaped as \\n'
' is escaped as \\n\ is escaped as \\

Quotes only need to be escaped inside a string enclosed with the same type of quote, as shown in the following example.
"I am a "double quoted" string inside "double quotes"."

- Null

**Environment variables**

Environment names:

- orgId
- projectId
- projectName
- deploymentId
- deploymentName
- blueprintId
- blueprintVersion
- blueprintName
- requestedBy (user)
- requestedAt (time)

Syntax:

```
env.ENV_NAME
```

Example:

```
${env.blueprintId}
```

**Resource variables**

Resource variables let you bind to resource properties from other resources.

Syntax:

```
resource.RESOURCE_NAME.PROPERTY_NAME
```

Examples:

- `${resource.db.id}`
- `${resource.db.networks[0].address}`
- `${resource.app.id}` (Return the string for non-clustered resources, where count isn't specified. Return the array for clustered resources.)
- `${resource.app[0].id}` (Return the first entry for clustered resources.)

**Resource self variables**

Resource self variables are allowed only for resources supporting the allocation phase. Resource self variables are only available (or only have a value set) after the allocation phase is complete.
Syntax:

self.property_name

Example:

${self.address} (Return the address assigned during the allocation phase.)

Note that for a resource named resource_x, self.property_name and resource.resource_x.property_name are the same and are both considered self-references.

Conditions

Syntax:

- Equality operators are == and !=.
- Relational operators are < > <= and >=.
- Logical operators are && || and !.
- Conditionals use the pattern:

  condition-expression ? true-expression : false-expression

Examples:

${input.count < 5 && input.size == 'small'}

${input.count < 2 ? "small" : "large"}

Cluster count index

Syntax:

count.index

Examples:

- Return the node type for clustered resources:

  ${count.index == 0 ? "primary" : "secondary"}

- Set the size of each disk during allocation:

```json
inputs:
  disks:
    type: array
    minItems: 0
    maxItems: 12
    items:
      type: object
      properties:
        size:
          type: integer
          title: Size (GB)
          minSize: 1
          maxSize: 2048
```
resources:

Cloud_vSphere_Disk_1:
  type: Cloud.vSphere.Disk
  allocatePerInstance: true
  properties:
    capacityGb: '${input.disks[count.index].size}'
    count: '${length(input.disks)}'

For more examples, see Machine and disk clusters in Cloud Assembly.

**Arithmetic operators**

Syntax:

Operators are + – / * and %.

Example:

${(input.count + 5) * 2}

**String concatenation**

Syntax:

${'ABC' + 'DEF'} evaluates to ABCDEF.

**Operators [ ] and .**

The expression follows ECMAScript in unifying the treatment of the [ ] and . operators.

So, expr.identifier is equivalent to expr["identifier"]. The identifier is used to construct a literal whose value is the identifier, and then the [ ] operator is used with that value.

Example:

${resource.app.networks[0].address}

In addition, when a property includes a space, delimit with square brackets and double quotes instead of using dot notation.

Incorrect:

input.operating system

Correct:

input["operating system"]

**Construction of map**

Syntax:

${{"key1":"value1", "key2":input.key2}}

**Construction of array**

Syntax:
```$[[
    'key1',
    'key2'
  ]]$```

Example:
```$[[1,2,3]]$```

## Functions

**Syntax:**

```$\{function(\text{arguments...})\}$```

Example:
```$\{$to\_lower(\text{resource.app.name})\}$```

### Table 6-1. Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>abs(number)</td>
<td>Absolute number value</td>
</tr>
<tr>
<td>avg(array)</td>
<td>Return average of all values from array of numbers</td>
</tr>
<tr>
<td>base64_decode(string)</td>
<td>Return decoded base64 value</td>
</tr>
<tr>
<td>base64_encode(string)</td>
<td>Return base64 encoded value</td>
</tr>
<tr>
<td>ceil(number)</td>
<td>Returns the smallest (closest to negative infinity) value that is greater than or equal to the argument and is equal to a mathematical integer</td>
</tr>
<tr>
<td>contains(array, value)</td>
<td>Check if array contains a value</td>
</tr>
<tr>
<td>contains(string, value)</td>
<td>Check if string contains a value</td>
</tr>
<tr>
<td>digest(value, type)</td>
<td>Return digest of value using supported type (md5, sha1, sha256, sha384, sha512)</td>
</tr>
<tr>
<td>ends_with(subject, suffix)</td>
<td>Check if subject string ends with suffix string</td>
</tr>
<tr>
<td>filter_by(array, filter)</td>
<td>Return only the array entries that pass the filter operation</td>
</tr>
<tr>
<td></td>
<td>filter_by([1,2,3,4], x =&gt; x &gt;= 2 &amp;&amp; x &lt;= 3)</td>
</tr>
<tr>
<td></td>
<td>returns [2, 3]</td>
</tr>
<tr>
<td></td>
<td>filter_by({'key1':1, 'key2':2}, (k,v) =&gt; v != 1)</td>
</tr>
<tr>
<td></td>
<td>returns [{'key2': 2}]</td>
</tr>
<tr>
<td>floor(number)</td>
<td>Returns the largest (closest to positive infinity) value that is less than or equal to the argument and is equal to a mathematical integer</td>
</tr>
<tr>
<td>format(format, values...)</td>
<td>Return a formatted string using Java Class Formatter format and values.</td>
</tr>
<tr>
<td>from_json(string)</td>
<td>Parse json string</td>
</tr>
<tr>
<td>join(array, delim)</td>
<td>Join array of strings with a delimiter and return a string</td>
</tr>
<tr>
<td>json_path(value, path)</td>
<td>Evaluate path against value using XPath for JSON.</td>
</tr>
<tr>
<td>keys(map)</td>
<td>Return keys of map</td>
</tr>
<tr>
<td>length(array)</td>
<td>Return array length</td>
</tr>
</tbody>
</table>
### Table 6-1. Functions (continued)

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>length(string)</td>
<td>Return string length</td>
</tr>
<tr>
<td>map_by(array, operation)</td>
<td>Return each array entry with an operation applied to it</td>
</tr>
<tr>
<td></td>
<td>map_by([1,2], x =&gt; x * 10)</td>
</tr>
<tr>
<td></td>
<td>returns [10, 20]</td>
</tr>
<tr>
<td></td>
<td>map_by([1,2], x =&gt; to_string(x))</td>
</tr>
<tr>
<td></td>
<td>returns ['1', '2']</td>
</tr>
<tr>
<td></td>
<td>map_by({'key1':1, 'key2':2}, (k,v) =&gt; {k:v*10})</td>
</tr>
<tr>
<td></td>
<td>returns [{'key1':10},{'key2':20}]</td>
</tr>
<tr>
<td>map_to_object(array, keyname)</td>
<td>Return an array of key:value pairs of the specified key name paired with values from another array</td>
</tr>
<tr>
<td></td>
<td>map_to_object(resource.Disk[*].id, &quot;source&quot;)</td>
</tr>
<tr>
<td></td>
<td>returns an array of key:value pairs that has a key field called source paired with disk ID strings</td>
</tr>
<tr>
<td></td>
<td>Note that</td>
</tr>
<tr>
<td></td>
<td>map_by(resource.Disk[*].id, id =&gt; {'source':id})</td>
</tr>
<tr>
<td></td>
<td>returns the same result</td>
</tr>
<tr>
<td>matches(string, regex)</td>
<td>Check if string matches a regex expression</td>
</tr>
<tr>
<td>max(array)</td>
<td>Return maximum value from array of numbers</td>
</tr>
<tr>
<td>merge(map, map)</td>
<td>Return a merged map</td>
</tr>
<tr>
<td>min(array)</td>
<td>Return minimum value from array of numbers</td>
</tr>
<tr>
<td>not_null(array)</td>
<td>Return the first entry which is not null</td>
</tr>
<tr>
<td>now()</td>
<td>Return current time in ISO-8601 format</td>
</tr>
<tr>
<td>range(start, stop)</td>
<td>Return a series of numbers in increments of 1 that begins with the start number and ends just before the stop number</td>
</tr>
<tr>
<td>replace(string, target, replacement)</td>
<td>Replace string containing target string with target string</td>
</tr>
<tr>
<td>reverse(array)</td>
<td>Reverse entries of array</td>
</tr>
<tr>
<td>slice(array, begin, end)</td>
<td>Return slice of array from begin index to end index</td>
</tr>
<tr>
<td>split(string, delim)</td>
<td>Split string with a delimiter and return array of strings</td>
</tr>
<tr>
<td>starts_with(subject, prefix)</td>
<td>Check if subject string starts with prefix string</td>
</tr>
<tr>
<td>substring(string, begin, end)</td>
<td>Return substring of string from begin index until end index</td>
</tr>
<tr>
<td>sum(array)</td>
<td>Return sum of all values from array of numbers</td>
</tr>
<tr>
<td>to_json(value)</td>
<td>Serialize value as json string</td>
</tr>
<tr>
<td>to_lower(str)</td>
<td>Convert string to lower case</td>
</tr>
</tbody>
</table>
Table 6-1. Functions (continued)

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>to_number(string)</td>
<td>Parse string as number</td>
</tr>
<tr>
<td>to_string(value)</td>
<td>Return string representation of the value</td>
</tr>
<tr>
<td>to_upper(str)</td>
<td>Convert string to upper case</td>
</tr>
<tr>
<td>trim(string)</td>
<td>Remove leading and trailing spaces</td>
</tr>
<tr>
<td>url_encode(string)</td>
<td>Encode string using url encoding specification</td>
</tr>
<tr>
<td>uuid()</td>
<td>Return randomly generated UUID</td>
</tr>
<tr>
<td>values(map)</td>
<td>Return values of map</td>
</tr>
</tbody>
</table>

Secret Cloud Assembly properties

A secret Cloud Assembly property is a reusable, encrypted value that project users may add to their cloud template designs.

Secure access keys and credentials are typical examples of secret properties. Once created and saved, a secret property value can never be unencrypted or read.

Creating a secret property

1. Log in to Cloud Assembly with project administrator role privileges.
2. Go to **Infrastructure > Administration > Secrets**, and click **New Secret**.
3. Select the project.
4. Enter a unique property name for the secret, without spaces or special characters.
   The name is the visible identifier for the secret.
5. Enter the secret value.
   When typing, the value is obscured by default, which protects it if the screen is shared.
   If needed, you can click the eye symbol to reveal and verify a value. After it is saved though, a secret value becomes encrypted in the database and can never be re-exposed.
6. Optionally, enter a longer description of the secret property.
7. Click **Create**.
Adding a secret property to a cloud template

Project users may add a secret property as a binding in cloud template code.

Note that starting to type the `${secret.` characters reveals a selection list of secrets that have been created for the project.

```
type: Cloud.Machine
properties:
  name: ourvm
  image: mint20
  flavor: small
  remoteAccess:
    authentication: publicPrivateKey
    sshKey: '${secret.ourPublicKey}'
  username: root
```

To add a secret property to a Terraform configuration, see Using a secret Cloud Assembly property in a Terraform configuration.

Remote access to a Cloud Assembly deployment

To remotely access a machine that Cloud Assembly has deployed, you add properties, before deployment, to the cloud template for that machine.

For remote access, you can configure one of the following authentication options.

**Note** In cases where keys need to be copied, you might also create a cloudConfig section in the cloud template, to automatically copy the keys upon provisioning. The specifics aren't documented here, but Machine initialization in Cloud Assembly provides general information about cloudConfig.

**Generate a key pair at Cloud Assembly provisioning time**

If you don't have your own public-private key pair for remote access authentication, you can have Cloud Assembly generate a key pair.
Use the following code as a guideline.

1. In Cloud Assembly, before provisioning, add `remoteAccess` properties to the cloud template as shown in the example.

   The username is optional. If you omit it, the system generates a random ID as the username.

   Example:

   ```json
   type: Cloud.Machine
   properties:
     name: our-vm2
     image: Linux18
     flavor: small
     remoteAccess:
       authentication: generatedPublicPrivatekey
       username: testuser
   ```

2. In Cloud Assembly, provision the machine from its cloud template, and bring it to a started-up state.

   The provisioning process generates the keys.

3. Locate the key name in the **Deployments > Deployments > Topology** properties.

4. Use the cloud provider interface, such as the vSphere client, to access the provisioned machine command line.

5. Grant read permission to the private key.

   ```
   chmod 600 key-name
   ```

6. Go to the Cloud Assembly deployment, select the machine, and click **Actions > Get Private Key**.

7. Copy the private key file to your local machine.

   A typical local file path is `/home/username/.ssh/key-name`.

8. Open a remote SSH session, and connect to the provisioned machine.

   ```
   ssh -i key-name user-name@machine-ip
   ```

**Supply your own public-private key pair to Cloud Assembly**

Many enterprises create and distribute their own public-private key pairs for authentication.

Use the following code as a guideline.

1. In your local environment, obtain or generate your public-private key pair.

   If you need it, here's some background on generating key pairs in **Linux** and **Windows**.

   For now, just generate and save the keys locally.

2. In Cloud Assembly, before provisioning, add `remoteAccess` properties to the cloud template as shown in the example.
The sshKey includes the long alphanumeric found within the public key file key-name.pub.

The username is optional and gets created for you to log in with. If you omit it, the system generates a random ID as the username.

Example:

```yaml
type: Cloud.Machine
properties:
  name: our-vm1
  image: Linux18
  flavor: small
  remoteAccess:
    authentication: publicPrivateKey
    sshKey: ssh-rsa
    Iq+5aGgBF3ZNT4o1baP5Ii+dstIcowRRkyo9bfpA1mj9t1s2f
    qGzvU66PX9iEaax5h2vNWFgw6ag+Zlzd6LhVdVo649f274/mIRi1d7UGW...
    username: testuser
```

3 In Cloud Assembly, provision the machine from its cloud template, and bring it to a started-up state.

4 Using the cloud vendor client, access the provisioned machine.

5 Add the public key file to the home folder on the machine. Use the key that you specified in remoteAccess.sshKey.

6 Verify that the private key file counterpart is present on your local machine.
   The key is typically /home/username/.ssh/key-name with no .pub extension.

7 Open a remote SSH session, and connect to the provisioned machine.
   ```
   ssh -i key-name user-name@machine-ip
   ```

**Supply an AWS key pair to Cloud Assembly**

By adding an AWS key pair name to the cloud template, you can remotely access a machine that Cloud Assembly deploys to AWS.

Be aware that AWS key pairs are region specific. If you provision workloads into us-east-1, the key pair must exist in us-east-1.

Use the following code as a guideline. This option works for AWS cloud zones only.

```yaml
type: Cloud.Machine
properties:
  image: Ubuntu
  flavor: small
  remoteAccess:
    authentication: keyPairName
    keyPair: cas-test
  constraints:
  - tag: 'cloud:aws'
```
Supply a username and password to Cloud Assembly

By adding a username and password to the cloud template, you can have simple remote access to a machine that Cloud Assembly deploys.

Although it is less secure, logging in remotely with a username and password might be all that your situation requires. Be aware that some cloud vendors or configurations might not support this less secure option.

1 In Cloud Assembly, before provisioning, add remoteAccess properties to the cloud template as shown in the example.

   Set the username and password to the account that you expect to log in with.

   Example:

   ```json
   type: Cloud.Machine
   properties:
     name: our-vm3
     image: Linux18
     flavor: small
     remoteAccess:
       authentication: usernamePassword
       username: testuser
       password: admin123
   ```

2 In Cloud Assembly, provision the machine from its cloud template, and bring it to a started-up state.

3 Go to your cloud vendor's interface, and access the provisioned machine.

4 On the provisioned machine, create or enable the account.

5 From your local machine, open a remote session to the provisioned machine IP address or FQDN, and log in with the username and password as usual.

SCSI disk placement with Cloud Assembly

To manage a SCSI disk, you must specify and know its SCSI controller and logical unit number (LUN). For a vSphere disk object, you can use Cloud Assembly to assign both values in the cloud template.

The ability to use different SCSI controllers is important for performance and is required for some deployment types, such as Oracle Real Application Clusters (RAC).

SCSI controller and LUN disk properties

To assign a SCSI controller and LUN, add the following cloud template properties:

   SCSIController

   unitNumber
You also have the option to omit the properties, in which case assignment follows a predictable default. Cloud Assembly no longer deploys SCSI disks in random order, which made them difficult to manage.

SCSI controllers and disks are numbered in order, with zero being first. Each SCSI controller can support SCSI disks of unit numbers 0–15.

**Option 1: Set both SCSI controller and unit number**

You may fully specify both properties as shown in the following example. If so, assignment of the SCSI controller and unit number match the values that you enter.

```yaml
resources:
  Cloud_vSphere_Machine_1:
    type: Cloud.vSphere.Machine
    properties:
      image: centos
      cpuCount: 1
      totalMemoryMB: 1024
      attachedDisks:
        - source: '${resource.Cloud_vSphere_Disk_1.id}'
        - source: '${resource.Cloud_vSphere_Disk_2.id}'
        - source: '${resource.Cloud_vSphere_Disk_3.id}'

  Cloud_vSphere_Disk_1:
    type: Cloud.vSphere.Disk
    properties:
      capacityGb: 1
      SCSIController: SCSI_Controller_2
      unitNumber: 0
  Cloud_vSphere_Disk_2:
    type: Cloud.vSphere.Disk
    properties:
      capacityGb: 1
      SCSIController: SCSI_Controller_2
      unitNumber: 1
  Cloud_vSphere_Disk_3:
    type: Cloud.vSphere.Disk
    properties:
      capacityGb: 1
      SCSIController: SCSI_Controller_3
      unitNumber: 4
```

**Option 2: Set only the SCSI controller**

You may specify the SCSI controller and omit the unit number. In this case, assignment of the SCSI controller matches the value you enter. The unit number is set to the first available unit number under that controller.

```yaml
resources:
  Cloud_vSphere_Machine_1:
    type: Cloud.vSphere.Machine
    properties:
      image: centos
```

Using and Managing VMware Cloud Assembly

VMware, Inc. 454
cpuCount: 1
totalMemoryMB: 1024
attachedDisks:
- source: '#{resource.Cloud_vSphere_Disk_1.id}'
- source: '#{resource.Cloud_vSphere_Disk_2.id}'
- source: '#{resource.Cloud_vSphere_Disk_3.id}'

Cloud_vSphere_Disk_1:
  type: Cloud.vSphere.Disk
  properties:
    capacityGb: 1
    SCSIController: SCSI_Controller_0

Cloud_vSphere_Disk_2:
  type: Cloud.vSphere.Disk
  properties:
    capacityGb: 1
    SCSIController: SCSI_Controller_0

Cloud_vSphere_Disk_3:
  type: Cloud.vSphere.Disk
  properties:
    capacityGb: 1
    SCSIController: SCSI_Controller_1

Option 3: Omit both properties

You may omit the SCSI controller and unit number. In this case, assignment is set to the first available SCSI controller, and the first available unit number under that controller.

resources:
  Cloud_vSphere_Machine_1:
    type: Cloud.vSphere.Machine
    properties:
      image: centos
      cpuCount: 1
      totalMemoryMB: 1024
      attachedDisks:
        - source: '#{resource.Cloud_vSphere_Disk_1.id}'
        - source: '#{resource.Cloud_vSphere_Disk_2.id}'
        - source: '#{resource.Cloud_vSphere_Disk_3.id}'
  Cloud_vSphere_Disk_1:
    type: Cloud.vSphere.Disk
    properties:
      capacityGb: 1
  Cloud_vSphere_Disk_2:
    type: Cloud.vSphere.Disk
    properties:
      capacityGb: 1
  Cloud_vSphere_Disk_3:
    type: Cloud.vSphere.Disk
    properties:
      capacityGb: 1
Not an option: LUN only

You cannot omit the SCSI controller and specify only a unit number. Doing so might result in a deployment where multiple SCSI controllers have a disk of that number but, for management purposes, you won’t know which disk is which.

Using inputs to set the SCSI controller and LUN

To make the design more dynamic, use inputs so that the user may specify which SCSI controller and unit number at request or update time.

```json
inputs:
  diskProperties:
    type: array
    minItems: 1
    maxItems: 10
    items:
      type: object
      properties:
        size:
          type: integer
        SCSIController:
          type: string
          title: SCSI Controller
          enum:
            - SCSI_Controller_0
            - SCSI_Controller_1
            - SCSI_Controller_2
            - SCSI_Controller_3
        unitNumber:
          type: integer
          title: Unit Number

resources:
  app:
    type: Cloud.vSphere.Machine
    allocatePerInstance: true
    properties:
      flavor: small
      image: centos
      attachedDisks: '${map_to_object(slice(resource.disk[*].id, 0, 4), ''source'')}'
  disk:
    type: Cloud.vSphere.Disk
    allocatePerInstance: true
    properties:
      capacityGb: '${input.diskProperties[count.index].size}''
      SCSIController: '${input.diskProperties[count.index].SCSIController}''
      unitNumber: '${input.diskProperties[count.index].unitNumber}''
    count: 4
```
Machine initialization in Cloud Assembly

You can apply machine initialization in Cloud Assembly by running commands directly or, if deploying to vSphere-based cloud zones, through customization specifications.

How commands and customization specifications work

- **Commands**
  A cloudConfig section in your cloud template code holds the commands that you want to run.

- **Customization specifications**
  A property in your cloud template code references a vSphere customization specification by name.

Commands and customization specifications might not mix

When deploying to vSphere, proceed carefully if you attempt to combine cloudConfig and customization specification initialization. They aren't formally compatible and might produce inconsistent or unwanted results when used together.

For an example of how commands and customization specifications interact, see vSphere static IP addresses in Cloud Assembly.

vSphere customization specifications in Cloud Assembly templates

When deploying to vSphere based cloud zones in Cloud Assembly, customization specifications can apply guest operating system settings at deployment time.

Enabling the customization specification

The customization specification must exist in vSphere, at the target that you deploy to.
Edit the cloud template code directly. The following example points to a `cloud-assembly-linux` customization specification for a WordPress host on vSphere.

```yaml
resources:
  WebTier:
    type: Cloud.vSphere.Machine
    properties:
      name: wordpress
      cpuCount: 2
      totalMemoryMB: 1024
      imageRef: 'Template: ubuntu-18.04'
      customizationSpec: 'cloud-assembly-linux'
      resourceGroupName: '/Datacenters/Datacenter/vm/deployments'
```

Whether to use customization specifications or cloudConfig commands

If you want the provisioning experience to match what you are currently doing in vSphere, continuing to use customization specifications might be the best approach. However, to expand to hybrid or multiple cloud provisioning, a more neutral approach is cloudConfig initialization commands.

For more about cloudConfig sections in cloud templates, see [Configuration commands in Cloud Assembly templates](#).

Commands and customization specifications might not mix

When deploying to vSphere, proceed carefully if you attempt to combine embedded cloudConfig command and customization specification initialization. They aren't formally compatible and might produce inconsistent or unwanted results when used together.

For an example of how commands and customization specifications interact, see [vSphere static IP addresses in Cloud Assembly](#).

Configuration commands in Cloud Assembly templates

You can add a cloudConfig section to Cloud Assembly template code, in which you add machine initialization commands that run at deployment time.

cloudConfig command formats

- **Linux**—initialization commands follow the open `cloud-init` standard.
- **Windows**—initialization commands use `Cloudbase-init`.

Linux `cloud-init` and Windows `Cloudbase-init` don't share the same syntax. A cloudConfig section for one operating system won't work in a machine image of the other operating system.
What cloudConfig commands can do

You use initialization commands to automate the application of data or settings at instance creation time, which can customize users, permissions, installations, or any other command-based operations. Examples include:

- Setting a hostname
- Generating and setting up SSH private keys
- Installing packages

Where cloudConfig commands can be added

You can add a cloudConfig section to cloud template code, but you can also add one to a machine image in advance, when configuring infrastructure. Then, all cloud templates that reference the source image get the same initialization.

You might have an image map and a cloud template where both contain initialization commands. At deployment time, the commands merge, and Cloud Assembly runs the consolidated commands.

When the same command appears in both places but includes different parameters, only the image map command is run.

See Learn more about image mappings in vRealize Automation Cloud for additional details.

Example cloudConfig commands

The following example cloudConfig section is taken from Create a basic cloud template cloud template code for the Linux-based MySQL server.

Note To ensure correct interpretation of commands, always include the pipe character `cloudConfig: |` as shown.

```
cloudConfig: |
    #cloud-config
    repo_update: true
    repo_upgrade: all
    packages:
        - apache2
        - php
        - php-mysql
        - libapache2-mod-php
        - php-mcrypt
        - mysql-client
    runcmd:
        - i=0; while [ $i -le 5 ]; do mysql --connect-timeout=3 -h ${DBTier.networks[0].address} -u root -pmysqlpassword -e "SHOW STATUS;" && break || sleep 15; i=$((i+1)); done
```
- mysql -u root -pmysqlpassword -h ${DBTier.networks[0].address} -e "create database wordpress_blog;"
- service apache2 reload

If a cloud-init script behaves unexpectedly, check the captured console output in /var/log/cloud-init-output.log when troubleshooting. For more about cloud-init, see the cloud-init documentation.

Commands and customization specifications might not mix

When deploying to vSphere, proceed carefully if you attempt to combine embedded cloudConfig command and customization specification initialization. They aren't formally compatible and might produce inconsistent or unwanted results when used together.

For an example of how commands and customization specifications interact, see vSphere static IP addresses in Cloud Assembly.

vSphere static IP addresses in Cloud Assembly

When deploying to vSphere in Cloud Assembly, you can assign a static IP address but must take care not to introduce conflicts between cloudConfig initialization commands and customization specifications.

Sample designs

The following designs safely apply a static IP address without any conflict between cloud template initialization commands and customization specifications. All contain the assignment: static network setting.
<table>
<thead>
<tr>
<th>Design</th>
<th>Sample Cloud Template Code</th>
</tr>
</thead>
</table>
| Assign a static IP address to a Linux machine that has no cloud-init code | resources:  
wpnet:  
  type: Cloud.Network  
  properties:  
    name: wpnet  
    networkType: public  
    constraints:  
      - tag: sqa  
DBTier:  
  type: Cloud.vSphere.Machine  
  properties:  
    flavor: small  
    image: linux-template  
    networks:  
      - name: '${wpnet.name}'  
        assignment: static  
        network: '${resource.wpnet.id}'  |
| Assign a static IP address to a Linux machine with cloud-init code that doesn't contain network assignment commands. NOTE: The vSphere customization spec is applied whether you set the customizeGuestOs property to true or omit the customizeGuestOs property. | Ubuntu sample | resources:  
wpnet:  
  type: Cloud.Network  
  properties:  
    name: wpnet  
    networkType: public  
    constraints:  
      - tag: sqa  
DBTier:  
  type: Cloud.vSphere.Machine  
  properties:  
    flavor: small  
    image: ubuntu-template  
    customizeGuestOs: true  
    cloudConfig: |  
      #cloud-config  
      ssh_pwauth: yes  
      chpasswd:  
        list: |  
        root:Pa$$w0rd  
        expire: false  
      write_files:  
        - path: /tmpFile.txt  
          content: |  
          ${resource.wpnet.dns}  
      runcmd:  
        - hostnamectl set-hostname --pretty $ {self.resourceName}  
        - touch /etc/cloud/cloud-init.disabled  
      networks:  
        - name: '${wpnet.name}'  
          assignment: static  
          network: '${resource.wpnet.id}'  |
| CentOS sample | resources:  
wpnet:  
  type: Cloud.Network  
  properties: |
### Sample Cloud Template Code

```plaintext
name: wpnet
networkType: public
constraints:
  - tag: sqa

DBTier:
  type: Cloud.vSphere.Machine
  properties:
    flavor: small
    image: centos-template
    customizeGuestOs: true
    cloudConfig: |
      #cloud-config
      write_files:
        - path: /test.txt
          content: |
            deploying in power off.
            then rebooting.

networks:
  - name: '${wpnet.name}'
    assignment: static
    network: '${resource.wpnet.id}''
```
### Design

Assign a static IP address to a Linux machine with cloud-init code that contains network assignment commands.

The `customizeGuestOs` property must be false.

### Sample Cloud Template Code

**Ubuntu sample**

```yaml
resources:
  wpnet:
    type: Cloud.Network
    properties:
      name: wpnet
      networkType: public
      constraints:
        - tag: sqa
  DBTier:
    type: Cloud.vSphere.Machine
    properties:
      flavor: small
      image: ubuntu-template
      customizeGuestOs: false
      cloudConfig: |
        #cloud-config
        write_files:
        - path: /etc/netplan/99-installer-config.yaml
          content: |
            network:
              version: 2
              renderer: networkd
              ethernets:
                ens160:
                  addresses:
                    - ${resource.DBTier.networks[0].address}/$
                      {resource.wpnet.prefixLength}
                  gateway4: $
                      {resource.wpnet.gateway}
                  nameservers:
                    search: $
                      {resource.wpnet.dnsSearchDomains}
                    addresses: ${resource.wpnet.dns}
          runcmd:
            - netplan apply
            - hostnamectl set-hostname --pretty $
              {self.resourceName}
            - touch /etc/cloud/cloud-init.disabled
              networks:
                - name: '${wpnet.name}'
                  assignment: static
                  network: '${{resource.wpnet.id}}'
```

**CentOS sample**

```yaml
resources:
  wpnet:
    type: Cloud.Network
    properties:
      name: wpnet
      networkType: public
      constraints:
        - tag: sqa
  DBTier:
    type: Cloud.vSphere.Machine
    properties:
      flavor: small
      image: centos-template
```
When basing the deployment on a referenced image, assign a static IP address to a Linux machine with cloud-init code that contains network assignment commands. The customizeGuestOs property must be false. In addition, the cloud template must not include the ovfProperties property, which blocks customization.

### Resources

```python
customizeGuestOs: false
cloudConfig: |
  #cloud-config
  ssh_pwauth: yes
  chpasswd:
    list: |
      root:VMware1!
      expire: false
  runcmd:
    - nmcli con add type ethernet con-name 'custom ens192'
      ifname ens192 ip4 ${self.networks[0].address}/
      ${resource.wpnet.prefixLength} gw4 $
      (resource.wpnet.gateway)
    - nmcli con mod 'custom ens192' ipv4.dns "$
      (join(resource.wpnet.dns,' '))"
    - nmcli con mod 'custom ens192' ipv4.dns-search "$
      (join(resource.wpnet.dnsSearchDomains,' '))"
    - nmcli con down 'System ens192' ; nmcli
      con up 'custom ens192'
    - nmcli con del 'System ens192'
    - hostnamectl set-hostname --static `dig -x
      ${self.networks[0].address} +short | cut -d "." -f 1`
      ; hostnamectl set-hostname --pretty $
      {self.resourceName}
    - touch /etc/cloud/cloud-init.disabled
  networks:
    - name: '${wpnet.name}'
      assignment: static
      network: '${resource.wpnet.id}'
```

### When basing the deployment on a referenced image

When basing the deployment on a referenced image, assign a static IP address to a Linux machine with cloud-init code that contains network assignment commands. The customizeGuestOs property must be false. In addition, the cloud template must not include the ovfProperties property, which blocks customization.

### Resources

```python
resources:
  wpnet:
    type: Cloud.Network
    properties:
      name: wpnet
      networkType: public
      constraints:
        - tag: sqa
  DBTier:
    type: Cloud.vSphere.Machine
    properties:
      flavor: small
  imageRef: 'https://cloud-images.ubuntu.com/releases/focal/release/ubuntu-20.04-server-cloudimg-amd64.ova'
  customizeGuestOs: false
  cloudConfig: |
    #cloud-config
    ssh_pwauth: yes
    chpasswd:
      list: |
        root:Pa$$w0rd
        ubuntu:Pa$$w0rd
        expire: false
    write_files:
      - path: /etc/netplan/99-netcfg-vrac.yaml
        content: |
          network:
            version: 2
            renderer: networkd
```
Design

### Sample Cloud Template Code

```yaml
ethernets:
  ens192:
    dhcp4: no
dhcp6: no
addresses:
- ${resource.DBTier.networks[0].address}://${resource.wpnet.prefixLength}
gateway4: ${resource.wpnet.gateway}
nameservers:
  search: ${resource.wpnet.dnsSearchDomains}
  addresses: ${resource.wpnet.dns}
runcmd:
  - netplan apply
  - hostnamectl set-hostname --pretty ${self.resourceName}
  - touch /etc/cloud/cloud-init.disabled
networks:
  - name: '${wpnet.name}'
    assignment: static
    network: '${resource.wpnet.id}'
```

#### Designs that won't work or might produce unwanted results

- The cloud-init code doesn't contain network assignment commands, and the customizeGuestOs property is false.
  
  Neither initialization commands nor customization spec are present to configure network settings.

- The cloud-init code doesn't contain network assignment commands, and the ovfProperties property is set.
  
  Initialization commands aren't present, but ovfProperties blocked the customization spec.

- The cloud-init code contains network assignment commands, and the customizeGuestOs property is missing or set to true.
  
  Application of the customization spec conflicts with initialization commands.

#### Other workarounds for cloud-init and customization specs

When deploying to vSphere, you can also customize an image to work around cloud-init and customization spec conflicts. See the following external repository for more information.

- [vSphere Image Preparation Scripts](#)

#### Delayed deployment in Cloud Assembly

A virtual machine might need to be fully initialized before proceeding with Cloud Assembly deployment.
For example, deploying a machine that is still installing packages and starting a web server might lead to conditions where a fast user tries to reach the application before it's available.

Be aware of the following considerations when using this feature.

- The feature makes use of the `cloud-init` phone_home module and is available when deploying Linux machines.
- Phone home isn't available for Windows because of Cloudbase-init limitations.
- Phone home can affect deployment order like an explicit dependency, but has more flexibility around timing and processing options.
  
  See [Creating bindings and dependencies between resources in Cloud Assembly](#).
- Phone home requires a cloudConfig section in the cloud template.
- Your creativity is a factor. Initialization commands might include embedded wait time between operations, which can be used in concert with phone home.
- Cloud template-based phone home won't work if the machine template already contains phone_home module settings.
- The machine must have outbound communication access back to Cloud Assembly.

To introduce a deployment delay in Cloud Assembly, add a `cloudConfigSettings` section to the cloud template:

```
cloudConfigSettings:
  phoneHomeShouldWait: true
  phoneHomeTimeoutSeconds: 600
  phoneHomeFailOnTimeout: true
```

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>phoneHomeShouldWait</td>
<td>Whether to wait for initialization, true or false.</td>
</tr>
<tr>
<td>phoneHomeTimeoutSeconds</td>
<td>When to decide whether to proceed with deployment even though initialization is still running. Default is 10 minutes.</td>
</tr>
<tr>
<td>phoneHomeFailOnTimeout</td>
<td>Whether to proceed with deployment after timing out, true or false. Note that even when proceeding, deployment might still fail for separate reasons.</td>
</tr>
</tbody>
</table>

**Windows guest customization in Cloud Assembly**

To have Cloud Assembly automatically initialize a Windows machine at deployment, prepare an image that supports Cloudbase-Init, then a cloud template that contains the appropriate commands.

The image creation process varies depending on cloud vendor. The example shown here is for vSphere.
Windows Cloud Assembly image for vSphere

For Cloud Assembly to initialize a Windows machine deployed to vSphere, the image needs to be based on a vSphere template with Cloudbase-Init installed and configured.

Creating the image

1. Use vSphere to make and power on a Windows virtual machine.
2. On the virtual machine, log in to Windows.
3. Download Cloudbase-Init.
   
   https://cloudbase.it/cloudbase-init/#download

4. Start the Cloudbase-Init setup .msi file.
   
   During installation, enter Administrator as the username, and select the option to run as LocalSystem.

   ![Cloudbase-Init Setup](image)

   Other setup selections can remain as default values.

5. Allow the installation to run, but do not close the final Completed page of the setup wizard.

   **Important** Do not close the final page of the setup wizard.

6. With the Completed page of the setup wizard still open, use Windows to navigate to the Cloudbase-Init installation path, and open the following file in a text editor.

   conf\cloudbase-init-unattend.conf

7. Set `metadata_services` to `OvfService` as shown. Add the setting if it doesn't already exist.

   `metadata_services=cloudbaseinit.metadata.services.ovfservice.OvfService`

8. Save and close `cloudbase-init-unattend.conf`.

9. In the same folder, open the following file in a text editor.

   `conf\cloudbase-init.conf`
10 Set `first_logon_behaviour`, `metadata_services`, and `plugins` as shown. Add the settings if they don't already exist.

```bash
first_logon_behaviour=always
metadata_services=cloudbaseinit.metadata.services.ovfservice.OvfService
```

11 Save and close `cloudbase-init.conf`.

12 On the Completed page of the setup wizard, select the options to run Sysprep and to shut down after Sysprep, then click Finish.

**Note** VMware has seen cases where running Sysprep prevents deployments of the image from working.

When deploying, Cloud Assembly applies a dynamically generated customization specification, which disconnects the network interface. The pending Sysprep state in the image might cause the customization specification to fail and leave the deployment disconnected.

If you suspect that this is happening in your environment, try leaving the Sysprep options deactivated when creating the image.

13 After the virtual machine shuts down, use vSphere to turn it into a template.

**Additional details**

The following table expands upon the configuration entries made during setup.
## Configuration Setting

<table>
<thead>
<tr>
<th>Configuration Setting</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Username, CreateUserPlugin, and SetUserPasswordPlugin</td>
<td>After Sysprep, first boot uses CreateUserPlugin to create the username Administrator account with a blank password. SetUserPasswordPlugin allows Cloudbase-Init to change the blank password to the remote access password that will be included in the cloud template.</td>
</tr>
<tr>
<td>First Logon Behavior</td>
<td>This setting prompts the user to change the password upon first login.</td>
</tr>
<tr>
<td>Metadata services</td>
<td>By listing only OvfService, Cloudbase-Init won’t try to find other metadata services that aren’t supported in vCenter. This results in cleaner log files, because the logs would otherwise fill with entries about failing to find those other services.</td>
</tr>
<tr>
<td>Plugins</td>
<td>By listing only plugins with capabilities supported by OvfService, logs are again cleaner. Cloudbase-Init runs plugins in the order specified.</td>
</tr>
<tr>
<td>Run as LocalSystem</td>
<td>This setting supports any advanced initialization commands that might require Cloudbase-Init to run under a dedicated administrator account.</td>
</tr>
</tbody>
</table>

## Cloudbase-Init commands for Windows in Cloud Assembly

To run Windows machine initialization at deployment time, add Cloudbase-Init commands to the Cloud Assembly template code.

The example shown here is based on vSphere, but other cloud vendors should be similar.

### Prerequisites

- Create infrastructure. In Cloud Assembly, add your vSphere cloud account and an associated cloud zone.
- Add flavor and image mappings, and add network and storage profiles.
  
  In your infrastructure, an image mapping must point to a Windows template that you created to support Cloudbase-Init. See [Windows Cloud Assembly image for vSphere](#).
  
  If the template isn’t listed, go to Cloud Accounts, and synchronize images. Otherwise, automatic synchronization runs every 24 hours.
- Add a project, add users, and make sure the users can provision to your cloud zone.

For more about creating infrastructure and projects, see the examples in the [Tutorial: Setting up and testing multi-cloud infrastructure and deployments in Cloud Assembly](#).

### Procedure

1. In Cloud Assembly, go to the **Design** tab, and create a new cloud template.
2. Add a `cloudConfig` section with the Cloudbase-init commands that you want.
The following command examples create a new file at the Windows C: drive and set the host name.

```yaml
resources:
  Cloud_Machine_1:
    type: Cloud.Machine
    properties:
      image: cloudbase-init-win-2016
      flavor: small
      remoteAccess:
        authentication: usernamePassword
        username: Administrator
        password: Password1234@$
      cloudConfig: |
        #cloud-config
        write_files:
          content: Cloudbase-Init test
          path: C:\test.txt
      set_hostname: testname
```

For more information, see the Cloudbase-init documentation.

3 Add `remoteAccess` properties so that you configure the machine for initial login to Windows.

As mentioned when you created the template, the metadata service picks up the login credentials and exposes them to CreateUserServicePlugin and SetUserPasswordPlugin. Note that the password must meet Windows password requirements.

4 From Cloud Assembly, test and deploy the cloud template.

5 After deploying, use Windows RDP and the credentials in the template to log in to the new Windows machine and verify the customization.

In the example above, you would look for the `C:\test.txt` file, and check the system properties for the host name.

### Machine and disk clusters in Cloud Assembly

Cloud Assembly template designs can deploy a cluster of machines and attach a cluster of disks.

To deploy clusters of machines and disks, take advantage of the `allocatePerInstance` Cloud Assembly resource flags for requests, and `count.index` and `map_to_object` Cloud Assembly expression syntax in your cloud templates.

The following cloud template code examples can serve as guidelines for designs that deploy clusters.

#### Two machines that share a disk cluster

```yaml
resources:
  app0:
    type: Cloud.Machine
    allocatePerInstance: true
```
Variable number of machines with one disk each

inputs:
  count:
    type: integer
    default: 2
resources:
  Cloud_Machine_1:
    type: Cloud.Machine
    allocatePerInstance: true
    properties:
      image: ubuntu
      flavor: small
      count: '${input.count}'
      attachedDisks: '${map_to_object(slice(resource.disk[*].id, count.index, count.index + 1), "source")}'
    disk:
      type: Cloud.Volume
      allocatePerInstance: true
      properties:
        count: '${input.count}'
        capacityGb: 5

Variable number of machines with two disks each

inputs:
  count:
    type: integer
    default: 2
resources:
  Cloud_Machine_1:
    type: Cloud.Machine
    allocatePerInstance: true
    properties:
      image: ubuntu
flavor: small
count: ${input.count}
attachedDisks: '${map_to_object(slice(resource.disk[*].id, 2*count.index, 2*(count.index + 1)), "source")}'
disk:
  type: Cloud.Volume
  allocatePerInstance: true
  properties:
    count: ${2*input.count}
    capacityGb: 5

Set disk sizes at request time

inputs:
disksize:
  type: array
  minItems: 2
  maxItems: 2
  items:
    type: object
    properties:
      size:
        type: integer
resources:
  app:
    type: Cloud.Machine
    allocatePerInstance: true
    properties:
      flavor: small
      image: ubuntu
      attachedDisks: '${map_to_object(slice(resource.disk[*].id, 0, 2), "source")}'
disk:
  type: Cloud.Volume
  allocatePerInstance: true
  properties:
    count: 2
    capacityGb: ${input.disksize[count.index].size}

Custom naming for deployed resources in Cloud Assembly

As a cloud or project administrator, you have a prescribed naming convention for resources in your environment, and you want the deployed resource to follow those conventions without user interaction. You can create a naming template for all deployments from a Cloud Assembly project.
For example, your host naming convention is to prefix a resource as \textit{projectname-sitecode-costcenter-whereDeployed-identifier}. You configure the custom naming template for the machines for each project. Some of the template variables are pulled from the system as it is deployed, other are based on project custom properties. The custom naming template for the above prefix looks similar to the following example.

\begin{verbatim}
${project.name}-${resource.siteCode}-${resource.costCenter}-${endpoint.name}-${#####}
\end{verbatim}

The identifier, provided in the template as \texttt{${#####}} , shows a six digit identifier. The identifier is a counter that ensures uniqueness. The counter is global for the organization and increments across all projects, not only the current project. When you have multiple projects, do not expect a sequence from 000123 to 000124 for deployments in you current project. You can expect an increment from 000123 to 000127.

All resource names must be unique. To ensure uniqueness, use the incremental number property. The numbers increment for all deployments, including deployments that Cloud Assembly names. As your system becomes more robust, and because the system applies custom names to many resource types, the numbering can appear random, but the values still ensure uniqueness. The numbers also increment when you run a test deployment.

The following list is a sample of where the custom names are applied. The list is not meant to be definitive.

\textbf{Table 6-2. Sample list of resources to which custom names are applied}

<table>
<thead>
<tr>
<th>Resource Group</th>
<th>Resource Types</th>
</tr>
</thead>
</table>
| Virtual machines | Cloud.Machine  
Cloud.vSphere.Machine  
Cloud.AWS.EC2.Instance  
Cloud.GCP.Machine  
Cloud.Azure.Machine |
| Load balancers  | Cloud.LoadBalancer  
Cloud.NSX.LoadBalancer |
| Networks  | Cloud.Network  
Cloud.vSphere.Network  
Cloud.NSX.Network |
| Security groups  | Cloud.SecurityGroup |
| Disks  | Cloud.volume  
Cloud.vSphere.Disk  
Cloud.AWS.Volume  
Cloud.GCP.Disk  
Cloud.Azure.Disk |
| NSX  | Cloud.NSX.Gateway  
Cloud.NSX.NAT |
| Microsoft Azure  | Cloud.Azure.ResourceGroup |
In addition to the examples provided here, you can also add the user name, the image that is used, other built-in options, and simple strings. As you build the template, hints regarding possible options are provided.

Remember that some of the values you see are only use case examples. You won’t be able to use them letter-by-letter in your environment. Think about where you would make your own substitutions, or extrapolate from the example values, in order to fit your own cloud infrastructure and deployment management needs.

**Prerequisites**

- Verify that you know the naming convention that you want to use for deployments from a project.
- This procedure assumes you have or can create a simple cloud template that you use to test your custom host prefix naming.

**Procedure**

1. Select **Infrastructure > Projects**.
2. Select an existing project or create a new one.
3. On the **Provisioning** tab, locate the Custom Properties section and create the properties for the site code and cost center values.

   This is where you replace the values you see here with ones pertinent to your environment.

   a. Create a custom property with the name **siteCode** and the value **BGL**.
   b. Add another custom property with the name **costCenter** and the value **IT-research**.

4. Locate the Custom Naming section and add the following template.

   
   '${project.name}-${resource.siteCode}-${resource.costCenter}-${endpoint.name}-${#####}'

   You can copy in the string, but if this is your first naming template, consider using the hint text and quick select as you build the template.
Deploy a cloud template associated with the project to verify that the custom name is applied to the resource.

a. Click the Design tab, and then click a cloud template associated with the project.

b. Deploy the cloud template.

The Deployments tab opens, showing your deployment in process.

c. When deployment is completed, click the deployment name.

d. On the Topology tab, notice that your custom name is the resource name in the right pane.

If you deployed a test cloud template to verify the naming convention, you can delete the deployment.

What to do next

Create custom naming templates for your other projects.

How to add the SaltStack Config minion in Cloud Assembly designs

If you integrated SaltStack Config with vRealize Automation Cloud, you can install the minions on virtual machines in your deployments and use the SaltStack Config capabilities to perform additional management operations on the machines.

You can deploy minions as a part of your cloud templates or add one to machines as a day 2 action. You can also specify the state files as part of the template or as a day 2 action. The instructions for deploying in a cloud template are provided here. For more about the day 2 action, see Apply Salt Configuration in What actions can I run on Cloud Assembly deployments.

Minions are nodes that run the salt-minion service. The service listens to commands from a Salt master and performs the requested tasks.

At present, you can use the cloud template to install the minion on Linux machines.
What to do first

- Verify that you installed SaltStack Config and configured the integration. See Create a SaltStack Config integration in vRealize Automation Cloud.

- To familiarize yourself with how SaltStack Config works, including the key concepts of minions, see Minions.

- In SaltStack Config, verify that the FQDN name resolution from minion to master is working. To verify the FQDN on the Salt master in SaltStack Config, select Minions > All Minions, and filter the Minion ID column for the value saltmaster. Click saltmaster to see the details. Verify that the FQDN value is correct.

- Verify that the images in vSphere that you intend to deploy with a Salt minion have SSH capabilities enabled. SSH is used to remotely access the machine and deploy the minion.

- Verify that you can assign IP addresses to the machines you deploy. SaltStack Config requires the machines to have IP addresses. Use the IP addresses for the public IP CIDR range for the SDDC (software-defined datacenter) where your Salt master is located.

- Verify that the cloud template that you are adding the minion to is deployable before you add the saltConfiguration properties.

Add the minion to the cloud template

As a cloud template developer, you can add properties to the YAML that install the SaltStack Config minion when you deploy the template.

The core properties that you add to the template include remote access and some saltConfiguration properties. The procedure includes only selected properties. The YAML includes other saltConfiguration properties that are not used in this example. For more information, review the schema.

Although this example shows how to add the username and password for the remote access properties, you can configure a secret property and add the property to the template. For an example, see Secret Cloud Assembly properties.

Procedure

1. In Cloud Assembly, select Design > Cloud Templates.
2. Open an existing template.
3. In the code pane, add any saltConfiguration properties to the machine resources.

You are not required to include all of the possible properties. The values used in this example are explained in the table.

```yaml
saltConfiguration:
  masterId: saltstack_enterprise_installer
  minionId: '${input.minionID}'
  saltEnvironment: sse
```
Description of the saltConfiguration properties used in this example.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>masterID</td>
<td>The saltstack_enterprise_installer value is the default masterID. You might have master IDs defined in SaltStack Config in the Administration &gt; Master Keys.</td>
</tr>
<tr>
<td>minionID</td>
<td>The minion ID is assigned by SaltStack Config if you do not provide it. The default ID is the machine address. You cannot configure it to use the machine hostname. The current value in the sample YAML, '${input.minionID}', is a variable. This procedure includes the steps for adding an input value so that you can name it and test it. However, allowing users to define the minionID is not a common practice.</td>
</tr>
<tr>
<td>saltEnvironment</td>
<td>The sse value is the default file location for the state files. You might have your state files in other file server locations in SaltStack Config in Config &gt; File Server.</td>
</tr>
<tr>
<td>stateFiles</td>
<td>The doe.sls value is a default state file provided in the file server directory specified as the saltEnvironment.</td>
</tr>
<tr>
<td>variables</td>
<td>The variables are the values that the state file uses. In this example, the doe.sls accepts a user value.</td>
</tr>
</tbody>
</table>

4 Add the remoteAccess properties that SaltStack Config uses to communicate with the minion.

remoteAccess:
   authentication: usernamePassword
   username: adminUser
   password: adminPassword

5 Add Inputs so that the deploying user can specify the minionID.

inputs:
   minionID:
      type: string
      title: Minion ID

6 Verify that your YAML includes the properties similar to the following sample.

inputs:
   minionID:
      type: string
      title: Minion ID

resources:
Cloud_vSphere_Machine_1:
  type: Cloud.vSphere.Machine
  properties:
    image: centos
    flavor: small
  saltConfiguration:
    masterId: saltstack_enterprise_installer
    minionId: '${input.minionID}'
    saltEnvironment: sse
    stateFiles:
    - /doe.sls
    variables:
      user: sylvia
  remoteAccess:
    authentication: usernamePassword
    username: adminUser
    password: adminPassword

7 Test and deploy the cloud template.
8 Verify the Salt Configuration properties for the deployed machine.
   a Select Deployments > Deployments and open the deployment details.
   b On the Topology tab, click the machine and expand the properties in the right-hand pane.

Verify the minion in SaltStack Config

After you install the minion on the virtual machine, locate the minion and run any jobs or commands on the resource.

Procedure
1 To open SaltStack Config, click the applications menu in the upper right corner and click Cloud Services Console.
2 Click the SaltStack Config service tile.
3 In SaltStack Config, expand Minion Keys and click Accepted.
4 In the Minion ID column, click the filter icon and enter the name of the minion.

The minion name is the string that entered with when you deployed the cloud template in Cloud Assembly. In this example the minion ID is MinionID-0001.

5 To view the details, click the minion name.

You can run jobs or commands on the minion. For example, Sample Disk Usage. This job returns disk usage statistics for a minion.

What to do next

To use the SaltStack Config capabilities to manage your resources, see the SaltStack Config documentation.

Terraform configurations in Cloud Assembly

You can embed Terraform configurations as a resource in cloud templates in Cloud Assembly.
Preparing for Terraform configurations in Cloud Assembly

Before you add a Terraform configuration to a Cloud Assembly template, set up and integrate your version control repository.

1. About the Terraform runtime environment
2. Store Terraform configuration files in a version control repository
3. Enable cloud zone mapping
4. Integrate your repository with Cloud Assembly

About the Terraform runtime environment

Designs that include Terraform configurations require access to a Terraform runtime environment. The runtime environment consists of a Kubernetes cluster that runs Terraform CLI commands to perform requested operations. In addition, the runtime collects logs and returns the results from Terraform CLI commands.

The vRealize Automation Cloud hosted service uses its own preconfigured Kubernetes cluster that can't be modified by users.

Store Terraform configuration files in a version control repository

Cloud Assembly supports the following version control repositories for Terraform configurations.

- GitHub cloud
- GitLab cloud

In your version control repository, create a default directory with one layer of subdirectories, each with Terraform configuration files. Create one subdirectory per Terraform configuration.

1. Default directory
2. Single subdirectory layer
3. Deployment-ready Terraform configuration files

Don't include a Terraform state file with configuration files. If `terraform.tfstate` is present, errors occur during deployment.

Enable cloud zone mapping

If you expect to deploy to a cloud account, the Terraform runtime engine needs those cloud zone credentials.
In the project **Provisioning** tab, enable **Allow Terraform cloud zone mapping**.

Even though credentials are securely transmitted, for additional security, you should leave the option deactivated if project users don't need to deploy to a cloud account.

**Integrate your repository with Cloud Assembly**

In Cloud Assembly, go to **Infrastructure > Connections > Integrations**.

Add an integration to the repository offering type where you stored the Terraform configurations: GitHub or GitLab.

When you add your project to the integration, select the **Terraform Configurations** type, and identify the repository and branch. **Folder** is the default directory of your earlier structure.

![Add Repository: testProject](image)

**Designing for Terraform configurations in Cloud Assembly**

With your repository and Terraform configuration files in place, you can design a Cloud Assembly template for them.

1. **Prerequisites**
2. **Enable Terraform runtime versions**
3. **Add Terraform resources to the design**
4. **Deploy the cloud template**
Prerequisites
Set up and integrate your version control repository. See Preparing for Terraform configurations in Cloud Assembly.

Enable Terraform runtime versions
You can define the Terraform runtime versions available to users when deploying Terraform configurations. Note that Terraform configurations might also include internally coded version constraints.

To create the list of allowable versions, go to Infrastructure > Configure > Terraform Versions.

Add Terraform resources to the design
Create your cloud template that includes Terraform configurations.

1  In Cloud Assembly, go to Design > Cloud Templates and click New from > Terraform.

The Terraform configuration wizard appears.

2  Follow the prompts.

<table>
<thead>
<tr>
<th>Wizard Page</th>
<th>Setting</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>New Cloud Template</td>
<td>Name</td>
<td>Give the design an identifying name.</td>
</tr>
<tr>
<td></td>
<td>Description</td>
<td>Explain what the design is for.</td>
</tr>
<tr>
<td></td>
<td>Project</td>
<td>Select the project that includes the repository integration where the Terraform configuration is stored.</td>
</tr>
<tr>
<td>Configuration Source</td>
<td>Repository</td>
<td>Select the integrated repository where you stored the Terraform configuration.</td>
</tr>
<tr>
<td></td>
<td>Commit</td>
<td>Select a repository commit, or leave the entry blank to use the Terraform configuration from the repository head.</td>
</tr>
<tr>
<td></td>
<td>Source directory</td>
<td>Select a subdirectory from the repository structure that you created. The example subdirectories shown in the earlier setup were demo1, demo2, and demo3.</td>
</tr>
<tr>
<td>Finalize Configuration</td>
<td>Repository</td>
<td>Verify the correct repository selection.</td>
</tr>
<tr>
<td></td>
<td>Source directory</td>
<td>Verify the correct directory selection.</td>
</tr>
<tr>
<td></td>
<td>Terraform version</td>
<td>Select the Terraform runtime version to run when deploying the Terraform configuration.</td>
</tr>
</tbody>
</table>
If the Terraform configuration included a provider block, verify the provider and cloud zone that this cloud template will deploy to.

Having no provider isn’t a problem. After finishing the wizard, just edit the provider and cloud zone in the template properties to add or change the deployment target.

<table>
<thead>
<tr>
<th>Wizard Page</th>
<th>Setting</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Providers</td>
<td>If the Terraform configuration included a provider block, verify the provider and cloud zone that this cloud template will deploy to. Having no provider isn’t a problem. After finishing the wizard, just edit the provider and cloud zone in the template properties to add or change the deployment target.</td>
</tr>
<tr>
<td></td>
<td>Variables</td>
<td>Select sensitive values for encryption, such as passwords.</td>
</tr>
<tr>
<td></td>
<td>Outputs</td>
<td>Verify the outputs from the Terraform configuration, which convert to expressions that your design code can further reference.</td>
</tr>
</tbody>
</table>

3  **Click Create.**

The Terraform resource appears on the cloud template canvas, with Cloud Assembly code that reflects the Terraform configuration to deploy.

If desired, you can add other Cloud Assembly resources to the cloud template, to combine Terraform and non-Terraform code into a hybrid design.

**Note** Updating Terraform configurations in the repository doesn't synchronize the changes into your cloud template. Automatic synchronization can introduce security risks, such as newly added sensitive variables.

To capture Terraform configuration changes, rerun the wizard, choose the new commit, and identify any new sensitive variables.
Deploy the cloud template

When you deploy the cloud template, the deployment **History** tab lets you expand an event such as an allocate or create phase, to inspect a log of messages from the Terraform CLI.

Approvals—In addition to the expected Terraform phases such as PLAN, ALLOCATE, or CREATE, Cloud Assembly introduces governance by means of an approval phase. See [How do I configure Service Broker approval policies](#) for more information about request approvals.

<table>
<thead>
<tr>
<th>Timestamp</th>
<th>Status</th>
<th>Resource type</th>
<th>Resource name</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aug 3, 2022</td>
<td>PLAN_FINISHED</td>
<td>Cloud.Terraform.Configuration</td>
<td>terraform</td>
<td>Creating 2 Terraform resources, updating 0 Terraform resources, deleting 0 Terraform resources</td>
</tr>
<tr>
<td>Aug 3, 2022</td>
<td>PLAN_IN_PROGRESS</td>
<td>Cloud.Terraform.Configuration</td>
<td>terraform</td>
<td>Show Logs <img src="image" alt="" /></td>
</tr>
<tr>
<td>Aug 3, 2022</td>
<td>INITIALIZATION_IN_PROGRESS</td>
<td></td>
<td></td>
<td>2:24:23 PM: Terraform has been successfully initialized!</td>
</tr>
<tr>
<td>Aug 3, 2022</td>
<td>INITIALIZATION_IN_PROGRESS</td>
<td></td>
<td></td>
<td>2:24:23 PM: Refreshing Terraform state in-memory prior to plan...</td>
</tr>
<tr>
<td>Aug 3, 2022</td>
<td>INITIALIZATION_IN_PROGRESS</td>
<td></td>
<td></td>
<td>2:24:23 PM: The refreshed state will be used to calculate this plan, but will not be persisted to local or remote state storage.</td>
</tr>
</tbody>
</table>

After deploying, you see an outer resource that represents the overall Terraform component, with child resources inside for the separate components that Terraform created. The parent Terraform resource controls the lifecycle of the child resources.

Using a secret Cloud Assembly property in a Terraform configuration

You can apply secret, encrypted values to Terraform configurations that you add to Cloud Assembly cloud template designs.

1. In your git repository, add a Terraform configuration source file that references the secret properties as variables.
In this Terraform configuration source example, API and application keys are the secret variables.

```terraform
variable "datadog_api_key" {
  description = "Datadog API Key"
}

variable "datadog_app_key" {
  description = "Datadog App Key"
}

provider "datadog" {
  api_key = "${var.datadog_api_key}"
  app_key = "${var.datadog_app_key}"
}

# Create a new monitor
resource "datadog_monitor" "default" {
  # ...
}

# Create a new timeboard
resource "datadog_timeboard" "default" {
  # ...
}
```

2 In Cloud Assembly, go to Infrastructure > Administration > Secrets, and enter your secret property values.

Add secret names and corresponding values. For the names, it's easiest to simply enter the same name as the variable name from your Terraform source.

If needed, see Secret Cloud Assembly properties for more details.

3 In Cloud Assembly, import the Terraform configuration for use in a cloud template.

Go to Design > Cloud Templates and click New From > Terraform.

**Note** Even though the variables appear for selection on the last page of the wizard, you do not need to set the secret variables as sensitive. Secret Cloud Assembly variables will already be encrypted and do not need the encryption that the wizard applies.

If needed, see Designing for Terraform configurations in Cloud Assembly for more details.
The example cloud template should look similar to the following code:

```json
inputs:
  datadog_api_key:
    type: string
    description: Datadog API Key
  datadog_app_key:
    type: string
    description: Datadog App Key
resources:
  terraform:
    type: Cloud.Terraform.Configuration
    properties:
      variables:
        datadog_api_key: '${input.datadog_api_key}'
        datadog_app_key: '${input.datadog_app_key}'
      providers: []
      terraformVersion: 0.12.29
      configurationSource:
        repositoryId: 0fbf8f5e-54e1-4da3-9508-2b701gf25f51
        commitId: ed12424b249aa50439kr1c268942a4616bd751b6
        sourceDirectory: datadog
```

4 In the code editor, for the secret values, manually change `input` to `secret` as shown.

```json
terraform:
  type: Cloud.Terraform.Configuration
  properties:
    variables:
      datadog_api_key: '${secret.datadog_api_key}'
      datadog_app_key: '${secret.datadog_app_key}'
```

5 In the `inputs:` section of the code, remove the input entries that were replaced by the bindings to secret properties.

Learn more about Terraform configurations in vRealize Automation Cloud

Be aware of certain limitations and troubleshooting when you embed Terraform configurations as a resource in vRealize Automation Cloud.

Limitations for Terraform configurations

- When validating a design with Terraform configurations, the TEST button checks Cloud Assembly syntax but not the native Terraform code syntax.
  In addition, the TEST button doesn't validate commit IDs associated with Terraform configurations.
- The recently released Terraform version 0.13 isn't officially supported yet.
For a cloud template that includes Terraform configurations, cloning the template to a different project requires the following workaround.

a. In the new project, under the **Integrations** tab, copy the `repositoryId` for your integration.

b. Open the clone template. In the code editor, replace the `repositoryId` with the one you copied.

In the version control repository, don’t include a Terraform state file with configuration files. If `terraform.tfstate` is present, errors occur during deployment.

**Supported day 2 actions for the parent Terraform resource**

For the parent Terraform resource, you can view or refresh the Terraform state file. For more about the state file actions, see the comprehensive list of actions at [What actions can I run on Cloud Assembly deployments](#).

**Supported day 2 actions for child resources**

After deploying Terraform configurations, it might take up to 20 minutes for a day 2 action to become available on child resources.

For child resources in a Terraform configuration, only the following subset of day 2 actions are supported. For details about the actions, look them up in the comprehensive list of actions at [What actions can I run on Cloud Assembly deployments](#).

<table>
<thead>
<tr>
<th>Provider</th>
<th>Terraform Resource Type</th>
<th>Supported Day 2 Actions</th>
</tr>
</thead>
<tbody>
<tr>
<td>AWS</td>
<td>aws_instance</td>
<td>Power On</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Power Off</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Reboot</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Reset</td>
</tr>
<tr>
<td>Azure</td>
<td>azurerm_virtual_machine</td>
<td>Power On</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Power Off</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Restart</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Suspend</td>
</tr>
<tr>
<td>vSphere</td>
<td>vsphere_virtual_machine</td>
<td>Power On</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Power Off</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Reboot</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Reset</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Shutdown</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Suspend</td>
</tr>
<tr>
<td>Provider</td>
<td>Terraform Resource Type</td>
<td>Supported Day 2 Actions</td>
</tr>
<tr>
<td>----------</td>
<td>------------------------</td>
<td>-------------------------</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Create Snapshot</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Delete Snapshot</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Revert Snapshot</td>
</tr>
<tr>
<td>GCP</td>
<td>google_compute_instance</td>
<td>Power On</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Power Off</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Create Snapshot</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Delete Snapshot</td>
</tr>
</tbody>
</table>

**Troubleshooting day 2 action availability**

Out-of-the-box (OOTB) day 2 actions that are missing or deactivated might need troubleshooting.

<table>
<thead>
<tr>
<th>Problem</th>
<th>Cause</th>
<th>Resolution</th>
</tr>
</thead>
<tbody>
<tr>
<td>A Terraform resource does not have an expected OOTB day 2 action on the Actions menu.</td>
<td>The action might not be supported for the provider and resource type as mentioned in the previous list. Alternatively, the action might need up to 20 minutes to appear due to the timing of resource discovery and resource caching.</td>
<td>Check the provider and resource type in the design. Wait up to 20 minutes for data collection to complete.</td>
</tr>
<tr>
<td>A Terraform resource does not have an expected day 2 action even after the 20 minutes to account for data collection.</td>
<td>A resource discovery problem is preventing the action from appearing. One way that happens is when the resource is accidentally created on an out-of-project cloud zone. For example, your project only includes a cloud account and region us-east-1 cloud zone, but the Terraform configuration includes a provider block for us-west-1, and you didn't change it at design time. Another possibility is that data collection isn't working.</td>
<td>Check the project cloud zones against the cloud zones in the design. Go to Infrastructure &gt; Connections &gt; Cloud Accounts and check the data collection status and last successful collection time for the cloud account.</td>
</tr>
<tr>
<td>Even though there are no obvious problems with the resource state and data collection, a day 2 action is deactivated (gray).</td>
<td>Occasional, intermittent timing issues and data collection failures are known to occur.</td>
<td>The problem should resolve itself within 20 minutes.</td>
</tr>
<tr>
<td>The wrong day 2 action is deactivated, one that should be active based on the resource state. For example, Power Off is enabled, and Power On is deactivated, even though the resource was powered off using the provider interface.</td>
<td>Data collection timing can cause a temporary mismatch. If you change the power state from outside vRealize Automation Cloud, it takes time to correctly reflect the change.</td>
<td>Wait up to 20 minutes.</td>
</tr>
</tbody>
</table>
Using custom Terraform providers in vRealize Automation Cloud

If you have created and want to use a custom Terraform provider, take the following steps.

1. Under the default Terraform directory in your git version control repository, add the following subdirectory structure.
   
   ```
   terraform.d/plugins/linux_amd64
   ```

2. Add your custom Terraform provider Go binaries to the `linux_amd64` directory.

   By default, `terraform init` will search that directory for custom provider plug-ins.

   **Note** VMware has seen cases where a custom Terraform provider fails to run and posts a `no such file or directory` message.

   If that happens, try recompiling your custom provider Go binaries with CGO deactivated (set to zero). CGO is for Go packages that call C code.

Custom resource types for Cloud Assembly cloud templates

When you create a cloud template in Cloud Assembly, the resource type palette includes resource types for the supported cloud account and integration endpoints. You might have use cases where you want to create cloud templates based on an expanded list of resource types. You can create custom resource types, add them to the design canvas, and create cloud templates that support your design and deployment needs.

Custom resource name and resource type

The custom resource name identifies your custom resource inside the cloud template resource type palette.

The resource type of a custom resource must begin with `Custom`, and each resource type must be unique. For example, you might set `Custom.ADUser` as a resource type for a custom resource that adds Active Directory users. Although the inclusion of `Custom` is not validated in the text box, the string is automatically added if you remove it.

Extensibility action custom resources

With custom resource types, you can use extensibility actions in cloud templates to build complex applications. For example, you can use the native integration of extensibility actions with Amazon Web Services and Microsoft Azure to easily integrate with their respective services. You can create extensibility action custom resources by clicking on the `Based on` option in the custom resource editor and selecting `ABX user-defined schema`. 
Lifecycle actions for extensibility action custom resources

When using a extensibility action for your custom resource, you can define the following lifecycle actions:

- **Create**: this extensibility action is called when a deployment is started.
- **Read**: this extensibility action is used to retrieve the latest state of the deployed resource.
- **Update**: this extensibility action is called when a cloud template property is updated. This action is triggered only when a property is not marked with `recreateOnUpdate`.
- **Destroy**: this extensibility action is called when a deployment is deleted.

These lifecycle actions can either be selected manually from your existing extensibility actions or generated automatically by selecting Generate Actions. When you select Generate Actions you must specify the project in which the new extensibility action will be generated in.

**Note** You can edit the extensibility actions associated with your lifecycle actions by clicking on the Open option next to the specific action.

vRealize Orchestrator custom resources

Each vRealize Orchestrator custom resource is based on a SDK inventory type and is created by a vRealize Orchestrator workflow that has an output which is an instance of your desired SDK type. Primitive types, such as Properties, Date, string, and number are not supported for the creation of custom resource types.

**Note** SDK object types can be differentiated from other property types by the colon ("\:" ) used to separate the plug-in name and the type name. For example, AD:UserGroup is an SDK object type used to manage Active Directory user groups.

You can use the built-in workflows in vRealize Orchestrator, or you can create your own. Using vRealize Orchestrator to create anything-as-a-service/XaaS workflows means that you can create a cloud template that adds an Active Directory user to machines at deployment time, or add a custom F5 load balancer to a deployment. You can create vRealize Orchestrator custom resources by clicking on the Based on option in the custom resource editor and selecting vRO inventory.
**vRealize Orchestrator custom resource external type**

The external type property defines the type of your vRealize Orchestrator custom resource. When you select a Create workflow in your custom resource type in Cloud Assembly, the external type drop-down appears underneath it. The drop-down includes external type properties, that are selected from the output parameters of the vRealize Orchestrator workflow. The selected workflow output properties included in the drop-down must be non-array SDK object types such as VC:VirtualMachine or AD:UserGroup.

**Note**  When creating custom workflows that use the dynamic type plug-in, verify that their variables are created by using the DynamicTypesManager.getObject() method.

When you define your custom resource types, you also define the scope of the availability of the select external type. The selected external type can be:

- Shared across projects.
- Available only for the selected project.

You can only have one custom resource type with a specific external type value per defined scope. For example, if you create a custom resource in your project that uses VC:VirtualMachine as an external type, you cannot create another custom resource for the same project that uses the same external type. You also cannot create two shared custom resources that use the same external type.

**vRealize Orchestrator lifecycle action validation**

When you add Create, Delete, and Update workflows as lifecycle actions to your custom resource, Cloud Assembly validates that the selected workflows have correct input and output property definitions.

- The Create workflow must have an output parameter that is an SDK object type, such as SSH:Host or SQL:Database. If the selected workflow does not pass the validation, you cannot add Update or Delete workflows, or save your changes to the custom resource.
- The Delete workflow must have an input parameter that is an SDK object type that matches the external type of the custom resource.
- The Update workflow must have both an input and output parameter that is an SDK object type that matches the external type of the custom resource.

**Custom resource property schema**

You can view the custom resource properties schema by selecting the Properties tab. The schema includes the name, data type, property type, and, if it is available, the description of a given property. The schema also defines if a specific property is required or optional in the cloud template.

**Note** For the property schema of extensibility action custom resources, all properties are required in the cloud template.
When you add vRealize Orchestrator workflows to your custom resource, their input and output parameters are added as properties.

For extensibility action custom resources, you must create the property schema of extensibility action custom resources manually in the Properties tab. You can create the property schema by using either the Code or Form tab.

- **Code**: Create the property schema by using YAML content.
- **Form**: By clicking New Property, you create a new property by configuring its name, display name, description, property type, and default value.

### Day 2 Operation Custom Request Forms

You can streamline the request form of the day 2 operations included in your custom resource by adding and modifying different types of resource properties.

For example, you can bind the value of an input parameter in your request form to an external source, such as a vRealize Orchestrator action that retrieves a deployment name or project name. You can also bind the value of a specific input parameter to the computed value of two other text boxes included in the same request form.

**Note**  This functionality is available for both custom resources and resource actions. You can customize the value of the input properties of your request form from the Values tab of the Request Parameters page of the custom resource or resource action editor.

### How to create a Cloud Assembly template that adds users to Active Directory

In addition to the Cloud Assembly cloud template resources that you use when you create cloud templates, you can also create your own custom resources.

Custom resources are vRealize Orchestrator or extensibility action objects that you manage through vRealize Automation Cloud with the lifecycle actions defined in the custom resource. The cloud template service automatically calls up the appropriate vRealize Orchestrator workflows or extensibility actions when the operation associated with a specific lifecycle action is triggered. You can extend the functionality of the resource type by also selecting vRealize Orchestrator workflows or extensibility actions that can be used as day 2 operations.

This use case uses built-in workflows that are provided in the vRealize Orchestrator library. It includes prescriptive values or strings to demonstrate how to perform the process. You can modify them to suit your environment.

For reference purposes, this use case uses a project named **DevOpsTesting**. You can replace this sample project with any project in your environment.

**Prerequisites**

- Verify that you configured a vRealize Orchestrator integration. See Configure a vRealize Orchestrator integration in Cloud Assembly.
Verify that the workflows that you are using for the create, update, destroy, and day 2 actions exist in vRealize Orchestrator and run successfully from there.

In vRealize Orchestrator, locate the resource type used by the workflows. The workflows included in this custom resource must all use the same resource type. In this use case, the resource type is AD:User. For more information on resource type validation, see Custom resource types for Cloud Assembly cloud templates.

By using the built-in Active Directory workflows in your vRealize Orchestrator integration, configure an Active Directory server.

Verify that you know how to configure and deploy a machine cloud template.

Procedure

1. Create an Active Directory custom resource for adding a user in a group.

   This step adds the custom resource to the cloud template design canvas as a resources type.


   b. Provide the following values.

      Remember, except for the workflow names, these are sample values.

      | Setting          | Sample Value |
      |------------------|--------------|
      | Name             | AD user      |
      |                  | This is the name that appears in the cloud template resource type palette. |
      | Resource Type    | Custom.ADUser |
      |                  | The resource type must begin with Custom. and each resource type must be unique. |
      |                  | Although the inclusion of Custom. is not validated in the text box, the string is automatically added if you remove it. |
      |                  | This resource type is added to the resource type palette so that you can use it in the cloud template. |

   c. To enable this resource type in the cloud template resource type list, verify that Activate option is toggled on.

   d. Select the Scope setting that makes the resource type available to any project.

   e. Under Based on, verify that vRO Inventory is selected as the lifecycle action provider.
f Select the workflows that define the resource and the day 2 actions.

**Note** The selected day 2 workflows must have an input parameter that is of the same type as the external type. The external type input is not displayed in the day 2 custom form requested by the user, as it is automatically bound to the custom resource.

<table>
<thead>
<tr>
<th>Setting</th>
<th>Sample Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lifecycle Actions - Create</td>
<td>Select the Create a user with a password in an organizational unit workflow. If you have multiple vRealize Orchestrator integrations, select the workflow on the integration instance you use to run these custom resources. After selecting the workflow, the external type drop-down menu becomes available and is automatically set to AD:User. <strong>Note</strong> An external source type can be used only once if shared and once per project. In this use case, you are providing the same custom resource for all the projects. It does mean that you cannot use AD:User for any other resource types for all projects. If you have other workflows that require the AD:User type, you must create individual custom resources for each project.</td>
</tr>
<tr>
<td>Lifecycle Actions - Destroy</td>
<td>Select the Destroy a user workflow.</td>
</tr>
<tr>
<td>Additional Actions</td>
<td>Select the Change a user password workflow. On the Add Action window, add a name for the action, such as password_change and click Add. To modify the action request form that the user responds to when they request the action, click the icon in the Request Parameters column. <strong>Note</strong> For additional action workflows, verify that the workflow has a input parameter that is of the same type as the external type.</td>
</tr>
</tbody>
</table>

In this example, there is no appropriate application of an update workflow. A common example of an update workflow, which makes changes to the provisioned custom resource, is scaling in or scaling out a deployment.
Review the schema key and type values in the Properties tab so that you understand the workflow inputs so that you can configure the inputs in the cloud template.

The schema lists the required and optional input values defined in the workflow. The required input values are included in the cloud template YAML.

In the Create a user workflow, accountName, displayName, and ouContainer are required input values. The other schema properties are not required. You can also use the schema to determine where you want to create bindings to other field values, workflows, or actions. Bindings are not included in this use case.

To finish creating your custom resource, click Create.

2 Create a cloud template that adds the user to a machine when you deploy it.
   a Select Design > Cloud Templates, and click New from > Blank canvas.
   b Name the cloud template Machine with an AD user.
   c Select the DevOpsTesting project and click Create.
   d Add and configure a vSphere machine.
   e From the custom resource list on the left of the cloud template design page, drag the AD user resource type onto the canvas.

**Note** You can select the custom resource by either scrolling down and selecting it from the left pane, or searching for it in the Search Resource Types text box. If the custom resource does not appear, click the refresh button next to the Search Resource Types text box.
On the right, edit the YAML code to add the mandatory input values and the password. Add an inputs section in the code so that users can provide the name of the users that they are adding. In the following example, some of these values are sample data. Your values might be different.

```
inputs:
  accountName:
    type: string
    title: Account name
    encrypted: true
  displayName:
    type: string
    title: Display name
  password:
    type: string
    title: Password
    encrypted: true
  confirmPassword:
    type: string
    title: Password
    encrypted: true
  ouContainer:
    type: object
    title: AD OU container
    $data: 'vro/data/inventory/AD:OrganizationalUnit'
    properties:
      id:
        type: string
      type:
        type: string
```

In the resources section, add `${input.input-name}` code to prompt for the user selection.

```
resources:
  Custom_ADUser_1:
    type: Custom.ADUser
    properties:
      accountName: '${input.accountName}'
      displayName: '${input.displayName}'
      ouContainer: '${input.ouContainer}'
      password: '${input.password}'
      confirmPassword: '${input.confirmPassword}'
```

3 Deploy the cloud template.
   a On the cloud template designer page, click **Deploy**.
   b Enter the Deployment Name **AD User Scott**.
   c Select the Cloud Template Version and click **Next**.
d Complete the deployment inputs.

e Click **Deploy**.

4 Monitor the provisioning request on the **Deployments** tab to ensure that the user is added to Active Directory and that the deployment is successful.

**What to do next**

When your tested cloud template is working, you can then begin using the **AD user** custom resource with other cloud templates.

**How to create a Cloud Assembly template that includes SSH**

You can create custom resources that you can use to build cloud templates using vRealize Orchestrator workflows. In this use case, you add a custom resource that adds an SSH host. You can then include the resource in cloud templates. This procedure also adds an update workflow so that users change the SSH configuration after deployment rather than perform individual day 2 actions.

Custom resources are vRealize Orchestrator or extensibility action objects that you manage through vRealize Automation Cloud with the lifecycle actions defined in the custom resource. The cloud template service automatically calls up the appropriate vRealize Orchestrator workflows or extensibility actions when the operation associated with a specific lifecycle action is triggered. You can extend the functionality of the resource type by also selecting vRealize Orchestrator workflows or extensibility actions that can be used as day 2 operations.

This use case uses built-in workflows provided in the vRealize Orchestrator library. It includes prescriptive values or strings to demonstrate how to perform the process. You can modify them to suit your environment.

For reference purposes, this use case uses a project named **DevOpsTesting**. You can replace the project with one that you already have.

**Prerequisites**

- Verify that you configured a vRealize Orchestrator integration. See **Configure a vRealize Orchestrator integration in Cloud Assembly**.

- Verify that the workflows that you are using for the create, update, destroy, and day 2 actions exist in vRealize Orchestrator and run successfully from there.

- In vRealize Orchestrator, locate the resource type used by the workflows. The workflows included in this custom resource must all use the same resource type. In this use case, the resource type is **SSH:Host**. For more information on resource type validation, see **Custom resource types for Cloud Assembly cloud templates**.

- Verify that you know how to configure and deploy a machine cloud template.
Procedure

1. Create an SSH host custom resource for adding SSH to a cloud template.

   This step adds the custom resource to the cloud template design canvas as a resource type.

   a. In Cloud Assembly, select **Design > Custom Resources**, and click **New Custom Resource**.

   b. Provide the following values.

      Remember, except for the workflow names, these are sample values.

      **Table 6-3.**

<table>
<thead>
<tr>
<th>Setting</th>
<th>Sample Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td><strong>SSH Host - DevOpsTesting Project</strong>&lt;br&gt;This is the name that appears in the cloud template resource type palette.</td>
</tr>
<tr>
<td>Resource Type</td>
<td><strong>Custom.SSHHost</strong>&lt;br&gt;The resource type must begin <strong>Custom.</strong> and each resource type must be unique.&lt;br&gt;Although the inclusion of <strong>Custom.</strong> is not validated in the text box, the string is automatically added if you remove it.&lt;br&gt;This resource type is added to the design canvas so that you can use it in the cloud template.</td>
</tr>
</tbody>
</table>

   c. To enable this resource type in the cloud template resource type list, verify that **Activate** option is toggled on.

   d. Select the **Scope** setting that makes the resource type available to the **DevOpsTesting** project.

   e. Under **Based on**, verify that **vRO Inventory** is selected as the lifecycle action provider.

   f. Select the workflows that define the resource.

<table>
<thead>
<tr>
<th>Setting</th>
<th>Setting</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lifecycle Actions - Create</td>
<td>Select the <strong>Add SSH Host</strong> workflow.&lt;br&gt;If you have multiple vRealize Orchestrator integrations, select the workflow on the integration instance you use to run these custom resources.&lt;br&gt;After select the workflow, the external type drop-down menu becomes available and is automatically set to <strong>SSH:Host</strong>. An external source type can be used only once if share and once per project. In this use case, you are providing the custom resource for only the <strong>DevOpsTesting</strong> project. If you had other workflows that require the <strong>SSH:Host</strong> type, you must create individual custom resources for each project.</td>
</tr>
<tr>
<td>Lifecycle Actions - Update</td>
<td>Select the <strong>Update SSH Host</strong> workflow.</td>
</tr>
<tr>
<td>Lifecycle Actions - Destroy</td>
<td>Select the <strong>Remove SSH Host</strong> workflow.</td>
</tr>
</tbody>
</table>
Review the schema key and type values in the Properties tab so that you understand the workflow inputs so that you can configure the inputs in the cloud template.

The schema lists the required and optional input values defined in the workflow. The required input values are included in the cloud template YAML.

In the Add SSH Host workflow, hostname, port, and username are required input values. The other schema properties are not required. You can also use the schema to determine where you want to create bindings to other field values, workflows, or actions. Bindings are not included in this use case.

To finish creating your custom resource, click Create.

Create a cloud template that adds the SSH host when you deploy it.

a Select Design > Cloud Templates, and click New from > Blank canvas.
b Name the cloud template Machine with SSH Host.
c Select the DevOpsTesting project, and click Create.
d Add and configure a vSphere machine.
e From the custom resource list on the left of the cloud template design page, drag the SSH Host - DevOpsTesting Project resource type onto the canvas.

**Note** You can select the custom resource by either scrolling down and selecting it from the left pane, or searching for it in the Search Resource Types text box. If the custom resource does not appear, click the refresh button next to the Search Resource Types text box.

A reminder that the resource type is available because it was configured for the project. If you were creating a cloud template for another project, you cannot see the resource type.
On the right, edit the YAML code to add the mandatory input values.

Add an `inputs` section in the code so that users can provide the user name and the host name at deployment time. In this example, the port default is 22. In the following example, some of these values are sample data. Your values might be different.

```yaml
inputs:
  hostname:
    type: string
    title: The hostname of the SSH Host
  username:
    type: string
    title: Username
```

In the `resources` section, add `${input.input-name}` code to prompt for the user selection.

```yaml
resources:
  Custom_SSHHost_1:
    type: Custom.SSHHost
    properties:
      port: 22
      hostname: '${input.hostname}'
      username: '${input.username}'
```

3 Deploy the cloud template.
   a On the cloud template designer page, click Deploy.
   b Enter the Deployment Name SSH Host Test.
   c Select the Cloud Template Version and click Next.
   d Complete the deployment inputs.
   e Click Deploy.

4 Monitor the provisioning request on the Deployments tab to ensure that the SSH host is included in the deployment and that the deployment is successful.

What to do next

When your tested cloud template is working, you can then begin using the SSH Host custom resource with other cloud templates.

Cloud Assembly designs that prepare for day 2 changes

In addition to the day 2 actions already associated with Cloud Assembly resource types, you have design options that let you prepare in advance for custom updates that users might need to make.
Day 2 preparation can involve the Cloud Assembly design interface or make direct use of cloud template code, or both.

- You can add inputs to cloud template code. Then, any update action to a deployment or deployed resource asks for fresh input values.
- You can use Cloud Assembly to design a custom action based on a vRealize Orchestrator workflow or action. Running the custom action results in vRealize Orchestrator making changes to the deployment or deployed resource.

How to use cloud template inputs for vRealize Automation Cloud day 2 updates

When designing cloud templates, vRealize Automation Cloud input parameters allow day 2 users to re-enter selections from the initial deployment request.

**Caution** Some property changes cause a resource to be re-created. For example, changing the `connection_string.name` under a `Cloud.Service.Azure.App.Service` deletes the existing resource and creates a new one.

When designing inputs to support day 2 changes, decide whether to allow inputs that delete and re-create resources. To learn which properties re-create a resource, follow the schema link at [vRealize Automation Cloud resource property schema](#).

For information on how to create inputs, see [User input in vRealize Automation Cloud requests](#).

For a specific day 2 example, see the next section.

How to move a deployed machine to another network

While maintaining deployments and networks, you might need the ability to relocate machines that you deployed with Cloud Assembly.

For example, you might deploy to a test network first, then move to a production network. The technique described here lets you design a cloud template in advance to prepare for such day 2 actions. Note that the machine is moved. It isn't deleted and redeployed.

This procedure only applies to `Cloud.vSphere.Machine` resources. It won't work for cloud agnostic machines deployed to vSphere.

**Prerequisites**

- The Cloud Assembly network profile must include all subnets that the machine will connect to. In Cloud Assembly, you can check networks by going to **Infrastructure > Configure > Network Profiles**.

  The network profile must be in an account and region that are part of the appropriate Cloud Assembly project for your users.

- Tag the two subnets with different tags. The example that follows assumes that `test` and `prod` are the tag names.
The deployed machine must keep the same IP assignment type. It can’t change from static to DHCP, or vice versa, while moving to another network.

Procedure

1. In Cloud Assembly, go to Design, and create a cloud template for the deployment.

2. In the inputs section of the code, add an entry that lets the user select a network.

   ```
   inputs:
   net-tagging:
   type: string
   enum:
   - test
   - prod
   title: Select a network
   ```

3. In the resources section of the code, add the Cloud.Network and connect the vSphere machine to it.

4. Under the Cloud.Network, create a constraint that references the selection from the inputs.

   ```
   resources:
   ABCServer:
   type: Cloud.vSphere.Machine
   properties:
   name: abc-server
   networks:
   - network: '${resource[“ABCNet”].id}’

   ABCNet:
   type: Cloud.Network
   properties:
   name: abc-network
   constraints:
   - tag: '${input.net-tagging}’
   ```

5. Continue with your design, and deploy it as you normally would. At deployment, the interface prompts you to select the test or prod network.

6. When you need to make a day 2 change, go to Deployments > Deployments, and locate the deployment associated with the cloud template.

7. To the right of the deployment, click Actions > Update.

8. In the Update panel, the interface prompts you the same way, to select the test or prod network.

9. To change networks, make your selection, click Next, and click Submit.
How to create a Cloud Assembly resource action to vMotion a virtual machine

After you deploy a cloud template, you can run day 2 actions that change the deployment. Cloud Assembly includes many day 2 actions, but you might want to provide others. You can create custom resource actions and make them available to users as day 2 actions.

The custom resource actions are based on vRealize Orchestrator workflows.

This example of a custom day 2 resource action is meant to introduce you to the creation process. To use resource actions effectively, you must be able to create vRealize Orchestrator workflows and actions that run the tasks you need.

Prerequisites

- Verify that you configured a vRealize Orchestrator integration. See Configure a vRealize Orchestrator integration in Cloud Assembly.
- Verify that the workflow that you are using for the day 2 action exists in vRealize Orchestrator and runs successfully there.

Procedure

1. Create a custom resource action that uses vMotion to move a vSphere virtual machine from one host to another.
   b. Provide the following values.

   Remember, except for the workflow names, these are sample values.

<table>
<thead>
<tr>
<th>Setting</th>
<th>Sample Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>vSphere_VM_vMotion&lt;br&gt;This is the name that appears in the resource actions list.</td>
</tr>
<tr>
<td>Display name</td>
<td>Move VM&lt;br&gt;This is the name that users see in the deployment actions menu.</td>
</tr>
</tbody>
</table>
c Click the **Activate** option to enable this action in the day 2 actions menu for resources that matches the resource type.

d Select the resource type and workflow that define the day 2 action.

<table>
<thead>
<tr>
<th>Setting</th>
<th>Sample Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Resource Type</td>
<td>Select the <strong>Cloud.vSphere.Machine</strong> resource type. This is the resource type that is deployed as a cloud template component, not necessarily what is in the cloud template. For example, you might have a cloud agnostic machine in your cloud template, but when it is deployed on a vCenter Server, the machine is <strong>Cloud.vSphere.Machine</strong>. Because the action applies to the deployed type, do not use cloud agnostic types when you define your resource actions. In this example, vMotion only works for vSphere machines, but you might have other actions that you want to run on multiple resource types. You must create an action for each resource type.</td>
</tr>
<tr>
<td>Workflow</td>
<td>Select the <strong>Migrate virtual machine with vMotion</strong> workflow. If you have multiple vRealize Orchestrator integrations, select the workflow on the integration instance you use to run these custom resource actions.</td>
</tr>
</tbody>
</table>

2 Create a binding for the vRealize Orchestrator properties to the Cloud Assembly schema properties. Cloud Assembly day 2 actions support three types of bindings.

<table>
<thead>
<tr>
<th>Binding type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>in request</td>
<td>The default value binding type. When selected, the input property is displayed in the request form and its value must be provided by the user at the request time.</td>
</tr>
</tbody>
</table>
| with binding action   | This option is available only for reference type inputs such as:  
  - VC:VirtualMachine  
  - VC:Folder  
  The user selects an action that performs the binding. The selected action must return the same type as the input parameter. The correct property definition is ${{properties.someProperty}}. |
| direct                | This option is available for input properties that use primitive data types. When selected, the property, with the suitable type, is mapped directly from the schema of the input property. The user selects the property from the schema tree. Properties with different types are disabled. |

In this use case, the binding is a vRealize Orchestrator action that makes the connection between vRealize Orchestrator **VC:VirtualMachine** input type used in the workflow and the
Cloud Assembly Cloud.vSphere.Machine resource type. By setting up the binding, you make the day 2 action seamless for the user requesting the vMotion action on a vSphere VM machine. The system provides the name in the workflow so that the user does not have to do it.

a. After selecting the Migrate virtual machine with vMotion workflow, navigate to the Property Binding pane.

b. Select the binding of the vm input property.

c. Under Binding, select with binding action.

The findVcVmByVcAndVmUuid action is automatically selected. This action comes preconfigured with your vRealize Orchestrator integration in Cloud Assembly.

d. Click Save.

3. To save the changes to your day 2 action, click Create.
4 To account for the other input parameters in the workflow, you can customize the request form that users see when they request the action.

a From **Resource Actions**, select your recently created day 2 action.

b Click **Edit Request Parameters**.

You can customize how the request page is presented to users.

<table>
<thead>
<tr>
<th>Default Field Name</th>
<th>Appearance</th>
<th>Values</th>
<th>Constraints</th>
</tr>
</thead>
<tbody>
<tr>
<td>Destination resource pool for the virtual machine. Default is the current resource pool.</td>
<td>Label = Target resource pool, Display type = Value Picker</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Destination host to which to migrate the virtual machine</td>
<td>Label = Target host, Display type = Value Picker</td>
<td>Required = Yes</td>
<td></td>
</tr>
<tr>
<td>Priority of the migration task</td>
<td>Label = Priority of the task</td>
<td>Value options: Constant in the text box, enter a comma-separated list.</td>
<td>Required = Yes</td>
</tr>
<tr>
<td>(Optional) Only migrate the virtual machine if its power on state matches the specified state</td>
<td>Delete this text box. vMotion can move machines in any power state.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Default Field Name**: Destination resource pool for the virtual machine. Default is the current resource pool.

**Appearance**: Label = Target resource pool, Display type = Value Picker.

**Values**: Required = Yes.

**Constraints**: Constant in the text box, enter a comma-separated list.

5 To limit when the action is available, you can configure the conditions.

For example, you only want the vMotion action to be available when the machine has four or fewer CPUs.

a Toggle on **Requires condition**.

b Enter the condition.

<table>
<thead>
<tr>
<th>Key</th>
<th>Operator</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>${properties.cpuCount}</td>
<td>lessThan</td>
<td>4</td>
</tr>
</tbody>
</table>

If you need complex conditions, see **How to build advanced conditions for Cloud Assembly custom actions**.

c Click **Update**.
6 Verify that the Move VM action is available for deployed machines that match the criteria.
   a Select **Deployments**.
   b Locate a deployment that includes a deployed machine that matches the defined criteria.
   c Open the deployment and select the machine.
   d Click actions in the right pane and verify that the Move VM action exists.

   e Run the action.

**How to build advanced conditions for Cloud Assembly custom actions**

As an alternative to the simple conditions list in Cloud Assembly, the advanced editor lets you assemble more complex criteria expressions to control when the action is available.

When creating a new resource action, select **Requires condition** and **Use advanced editor**. Then, enter the criteria expression that you want.

The expression is a clause or list of clauses, each of which is in key-operator-value format. The preceding figure shows criteria where the target must be powered on and present.
### Clauses

<table>
<thead>
<tr>
<th>Clause</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>and</td>
<td>All subclauses need to be true for the expression result to be true.</td>
<td>Evaluate as true only when both properties.powerState is ON and syncStatus is not MISSING.</td>
</tr>
</tbody>
</table>

```json
matchExpression:
  - and:
    - key: properties.powerState
      operator: eq
      value: ON
    - key: syncStatus
      operator: notEq
      value: MISSING
```

| or     | One or more of the subclauses need to be true for the expression result to be true. | Evaluate as true whether properties.powerState is ON or OFF. |

```json
matchExpression:
  - or:
    - key: properties.powerState
      operator: eq
      value: ON
    - key: properties.powerState
      operator: eq
      value: OFF
```

### Operators

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>eq</td>
<td>Equal. Look for an exact match.</td>
<td>Evaluate as true when properties.powerState is ON.</td>
</tr>
</tbody>
</table>

```json
matchExpression:
  - and:
    - key: properties.powerState
      operator: eq
      value: ON
```

| notEq    | Not equal. Avoid an exact match. | Evaluate as true when properties.powerState is not OFF. |

```json
matchExpression:
  - and:
    - key: properties.powerState
      operator: notEq
      value: OFF
```
<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>hasAny</td>
<td>Look for a match in a collection of objects.</td>
<td>Evaluate as true when the storage.disks array includes a 100 IOPS EBS object. matchExpression: - key: storage.disks operator: hasAny value: matchExpression: - and: - key: iops operator: eq value: 100 - key: service operator: eq value: ebs</td>
</tr>
<tr>
<td>in</td>
<td>Look for a match in a set of values.</td>
<td>Evaluate as true when properties.powerState is either OFF or SUSPEND. matchExpression: - and: - key: properties.powerState operator: in value: OFF, SUSPEND</td>
</tr>
<tr>
<td>notIn</td>
<td>Avoid matching a set of values.</td>
<td>Evaluate as true when properties.powerState is neither OFF nor SUSPEND. matchExpression: - and: - key: properties.powerState operator: notIn value: OFF, SUSPEND</td>
</tr>
<tr>
<td>greaterThan</td>
<td>Look for a match over a given threshold. Only applies to numeric values.</td>
<td>Evaluate as true when the first object in the storage.disks array has IOPS over 50. matchExpression: - and: - key: storage.disks[0].iops operator: greaterThan value: 50</td>
</tr>
<tr>
<td>lessThan</td>
<td>Look for a match under a given threshold. Only applies to numeric values.</td>
<td>Evaluate as true when the first object in the storage.disks array has IOPS under 200. matchExpression: - and: - key: storage.disks[0].iops operator: lessThan value: 200</td>
</tr>
<tr>
<td>Operator</td>
<td>Description</td>
<td>Example</td>
</tr>
<tr>
<td>-----------------------</td>
<td>-------------------------------------------------------</td>
<td>------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>greaterThanEquals</td>
<td>Look for a match at or above a given threshold. Only applies to numeric values.</td>
<td>Evaluate as true when the first object in the storage.disks array has IOPS of 100 or higher.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>matchExpression:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- and:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- key: storage.disks[0].iops operator: greaterThanEquals value: 100</td>
</tr>
<tr>
<td>lessThanEquals</td>
<td>Look for a match at or below a given threshold. Only applies to numeric values.</td>
<td>Evaluate as true when the first object in the storage.disks array has IOPS of 100 or lower.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>matchExpression:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- and:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- key: storage.disks[0].iops operator: lessThanEquals value: 100</td>
</tr>
<tr>
<td>matchesRegex</td>
<td>Use a regular expression to look for a match.</td>
<td>Evaluate as true when the properties.zone is us-east-1a or us-east-1c.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>matchExpression:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- and:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- key: properties.zone operator: matchesRegex value: (us-east-1)+(a</td>
</tr>
</tbody>
</table>

**Examples**

The following criteria expression evaluates as true when properties.tags includes a tag of key `key1` and value `value1`.

The outer expression uses `hasAny` because properties.tags is an array, and you want to evaluate as true whenever `key1=value1` appears in any of the key-value pairs in the array.

In the inner expression, there are two clauses, one for the key field and one for the value field. The properties.tags array holds key-value tagging pairs, and you need to match both the key and value fields.

```yaml
matchExpression:
  - key: properties.tags
    operator: hasAny
    value:
      matchExpression:
        - and:
          - key: key
            operator: eq
            value: key1
          - key: value
            operator: eq
            value: value1
```
The following criteria expression is similar to the previous example, but now evaluates as true whenever properties.tags includes either a tag of key1=value1 or key2=value2.

```yaml
matchExpression:
  - or:
    - key: properties.tags
      operator: hasAny
      value:
        matchExpression:
          - and:
            - key: key
              operator: eq
              value: key1
            - key: value
              operator: eq
              value: value1
    - key: properties.tags
      operator: hasAny
      value:
        matchExpression:
          - and:
            - key: key
              operator: eq
              value: key2
            - key: value
              operator: eq
              value: value2
```

Other Cloud Assembly code examples

Cloud template code in Cloud Assembly can be almost limitless in combination and application. Often, an example of successful code is your best starting point for further development. When following an example, make substitutions in order to apply your site settings in terms of resource names, values, and so on.

Documented Cloud Assembly template example

By including a thorough set of comments, this example lets you review the structure and purpose of the sections in a Cloud Assembly template, formerly called a blueprint.

```yaml
# *************************************************************************  
# This WordPress cloud template is enhanced with comments to explain its  
# parameters.  
# Try cloning it and experimenting with its YAML code. If you're new to  
# YAML, visit yaml.org for general information.  
# The cloud template deploys a minimum of 3 virtual machines and runs scripts  
# to install packages.  
```
name: WordPress Template with Comments
formatVersion: 1
version: 1

inputs:

platform:
type: string
title: Deploy to
oneOf:
  - title: AWS
    const: aws
  - title: Azure
    const: azure
  - title: vSphere
    const: vsphere
default: vsphere

osimage:
type: string
title: Operating System
description: Which OS to use
enum:
  - Ubuntu

Set the number of machines in the database cluster. Small and large correspond to 1 or 2 machines, respectively, which you see later, down in the resources section.
dbenvsize:
  type: string
  title: Database cluster size
  enum:
    - Small
    - Large
#
# Dynamically tag the machines that will be created. The # 'array' of objects means you can create as many key-value pairs as # needed. To see how array input looks when it's collected, # open the cloud template and click TEST.
#
Mtags:
  type: array
  title: Tags
  description: Tags to apply to machines
  items:
    type: object
    properties:
      key:
        type: string
        title: Key
      value:
        type: string
        title: Value
#
# Create machine credentials. These credentials are needed in # remote access configuration later, in the resources section.
#
username:
  type: string
  minLength: 4
  maxLength: 20
  pattern: '[a-z]+'
  title: Database Username
userpassword:
  type: string
  pattern: '[a-z0-9A-Z@#$]+'
  encrypted: true
  title: Database Password
#
# Set the database storage disk size.
#
databaseDiskSize:
  type: number
  default: 4
  maximum: 10
  title: MySQL Data Disk Size
  description: Size of database disk
# Set the number of machines in the web cluster. Small, medium, and large
# correspond to 2, 3, and 4 machines, respectively, which you see later,
# in the WebTier part of the resources section.

clusterSize:
  type: string
  enum:
    - small
    - medium
    - large
  title: Wordpress Cluster Size
  description: Wordpress Cluster Size

# Set the archive storage disk size.

archiveDiskSize:
  type: number
  default: 4
  maximum: 10
  title: Wordpress Archive Disk Size
  description: Size of Wordpress archive disk

# The resources section configures the deployment of machines, disks,
# networks, and other objects. In several places, the code pulls from
# the preceding interactive user inputs.

resources:

# Create the database server. Choose a cloud agnostic machine 'type' so
# that it can deploy to AWS, Azure, or vSphere. Then enter its property
# settings.

DBTier:
  type: Cloud.Machine
  properties:

# Descriptive name for the virtual machine. Does not become the hostname
# upon deployment.

  name: mysql

# Hard-coded operating system image to use. To pull from user input above,
# enter the following instead.

  image: '${input.osimage}'

# Hard-coded capacity to use. Note that the Cloud Assembly
Infrastructure must also have AWS, Azure, and vSphere flavors such as small, medium, and large mapped.

```yaml
flavor: small
```

Tag the database machine to deploy to the cloud vendor chosen from the user input. Tags are case-sensitive, so 'to_lower' forces the tag to lowercase to ensure a match with a site's tagging convention. It's important if platform input were to contain any upper case characters.

```yaml
constraints:
  - tag: '${"env:" + to_lower(input.platform)}'
```

Also tag the database machine with any free-form tags that were created during user input.

```yaml
tags: '${input.Mtags}'
```

Set the database cluster size by referencing the dbenvsize user input. Small is one machine, and large defaults to two.

```yaml
count: '${input.dbenvsize == "Small" ? 1 : 2}''
```

Add a variable to connect the machine to a network resource based on a property binding to another resource. In this case, it's the 'WP_Network' network that gets defined further below.

```yaml
networks:
  - network: '${resource.WP_Network.id}'
```

Enable remote access to the database server. Reference the credentials from the user input.

```yaml
remoteAccess:
  authentication: usernamePassword
  username: '${input.username}'
  password: '${input.userpassword}'
```

You are free to add custom properties, which might be used to initiate an extensibility subscription, for example.

```yaml
ABC-Company-ID: 9393
```

Run OS commands or scripts to further configure the database machine, via operations such as setting a hostname, generating SSH private keys, or installing packages.

```yaml
cloudConfig:
```
#cloud-config
repo_update: true
repo_upgrade: all
packages:
- mysql-server
runcmd:
- sed -e '/bind-address/ s/^#*/#/' -i /etc/mysql/mysql.conf.d/mysqld.cnf
- service mysql restart
- mysql -e "GRANT ALL PRIVILEGES ON *.* TO 'root'@'%' IDENTIFIED BY 'mysqlpassword';"
- mysql -e "FLUSH PRIVILEGES;"
attachedDisks: []
#
# Create the web server. Choose a cloud agnostic machine 'type' so that it
# can deploy to AWS, Azure, or vSphere. Then enter its property settings.
# ------------------------------------------------------------------------
WebTier:
  type: Cloud.Machine
  properties:

# Descriptive name for the virtual machine. Does not become the hostname
# upon deployment.
# ------------------------------------------------------------------------
  name: wordpress

# Hard-coded operating system image to use. To pull from user input above,
# enter the following instead:
# image: '${input.osimage}'
# ------------------------------------------------------------------------
  image: Ubuntu

# Hard-coded capacity to use. Note that the Cloud Assembly
# Infrastructure must also have AWS, Azure, and vSphere flavors
# such as small, medium, and large mapped.
# ------------------------------------------------------------------------
  flavor: small

# Set the web server cluster size by referencing the clusterSize user
# input. Small is 2 machines, medium is 3, and large defaults to 4.
# ------------------------------------------------------------------------
  count: '${input.clusterSize== "small" ? 2 : (input.clusterSize == "medium" ? 3 : 4)}'

# Set an environment variable to display object information under the
# Properties tab, post-deployment. Another example might be
# {env.blueprintID}
# ------------------------------------------------------------------------
  tags:
    - key: cas.requestedBy
      value: '${env.requestedBy}'
# You are free to add custom properties, which might be used to initiate
# an extensibility subscription, for example.

ABC-Company-ID: 9393

# Tag the web server to deploy to the cloud vendor chosen from the
# user input. Tags are case-sensitive, so 'to_lower' forces the tag to
# lowercase to ensure a match with your site's tagging convention. It's
# important if platform input were to contain any upper case characters.

constraints:
  - tag: "${"env": to_lower(input.platform)}"

# Add a variable to connect the machine to a network resource based on
# a property binding to another resource. In this case, it's the
# 'WP_Network' network that gets defined further below.

networks:
  - network: "${resource.WP_Network.id}"

# Run OS commands or scripts to further configure the web server,
# with operations such as setting a hostname, generating SSH private keys,
# or installing packages.

cloudConfig: |
  #cloud-config
  repo_update: true
  repo_upgrade: all
  packages:
    - apache2
    - php
    - php-mysql
    - libapache2-mod-php
    - php-mcrypt
    - mysql-client
  runcmd:
    - i=0; while [ $i -le 5 ]; do mysql --connect-timeout=3 -h ${DBTier.networks[0].address} -u root -pmysqlpassword -e "SHOW STATUS;" && break || sleep 15; i=$((i+1)); done
    - mysql -u root -pmysqlpassword -h ${resource.DBTier.networks[0].address} -e "CREATE DATABASE wordpress_blog;"
wp-config.php && sed -i -e "s/"define('DB_HOST', 'localhost');"/"define('DB_HOST', '${resource.DBTier.networks[0].address}');"/ /var/www/html/mywordpresssite/wp-config.php

# Create the network that the database and web servers connect to.
# Choose a cloud agnostic network 'type' so that it can deploy to AWS,
# Azure, or vSphere. Then enter its property settings.
# WP_Network:
  # type: Cloud.Network
  properties:
  
  # Descriptive name for the network. Does not become the network name
  # upon deployment.
  # name: WP_Network
  
  # Set the networkType to an existing network. You could also use a
  # constraint tag to target a specific, like-tagged network.
  # The other network types are private or public.
  # networkType: existing

# VMware hopes that you found this commented template useful. Note that
# you can also access an API to create templates, or query for input
# schema that you intend to request. See the following Swagger
documentation.
# www.mgmt.cloud.vmware.com/blueprint/api/swagger/swagger-ui.html

vSphere resource examples in Cloud Assembly

These code examples illustrate vSphere machine resources within Cloud Assembly cloud templates.
<table>
<thead>
<tr>
<th>Resource</th>
<th>Example Cloud Template</th>
</tr>
</thead>
</table>
| vSphere virtual machine with CPU, memory, and operating system | resources:  
demo-machine:  
  type: Cloud.vSphere.Machine  
  properties:  
    name: demo-machine  
    cpuCount: 1  
    totalMemoryMB: 1024  
    image: ubuntu |
| vSphere machine with a datastore resource | resources:  
demo-vsphere-disk-001:  
  type: Cloud.vSphere.Disk  
  properties:  
    name: DISK_001  
    type: 'HDD'  
    capacityGb: 10  
    dataStore: 'datastore-01'  
    provisioningType: thick |
| vSphere machine with an attached disk | resources:  
demo-vsphere-disk-001:  
  type: Cloud.vSphere.Disk  
  properties:  
    name: DISK_001  
    type: HDD  
    capacityGb: 10  
    dataStore: 'datastore-01'  
    provisioningType: thin  
demo-machine:  
  type: Cloud.vSphere.Machine  
  properties:  
    name: demo-machine  
    cpuCount: 2  
    totalMemoryMB: 2048  
    imageRef: >-  
    attachedDisks:  
      - source: '${demo-vsphere-disk-001.id}''
<table>
<thead>
<tr>
<th>Resource</th>
<th>Example Cloud Template</th>
</tr>
</thead>
</table>
| vSphere machine with a dynamic number of disks | inputs:
  disks:
    type: array
    title: disks
    items:
      title: disk
      type: object
      properties:
        size:
          type: integer
          title: size
          maxItems: 15
  resources:
    Cloud Machine 1:
      type: Cloud.vSphere.Machine
      properties:
        image: centos7
        flavor: small
        attachedDisks: '{map_to_object(resource.Cloud_Volume_1[*].id, "source")}'
    Cloud Volume 1:
      type: Cloud.Volume
      allocatePerInstance: true
      properties:
        capacityGb: '${input.disks[count.index].size}'
        count: '${length(input.disks)}' |
| vSphere machine from a snapshot image. Append a forward slash and the snapshot name. The snapshot image can be a linked clone. | resources:
  demo-machine:
    type: Cloud.vSphere.Machine
    properties:
      imageRef: 'demo-machine/snapshot-01'
      cpuCount: 1
      totalMemoryMB: 1024 |
| vSphere machine in a specific folder in vCenter | resources:
  demo-machine:
    type: Cloud.vSphere.Machine
    properties:
      name: demo-machine
      cpuCount: 2
      totalMemoryMB: 1024
      imageRef: ubuntu
      resourceGroupName: 'myFolder' |
<table>
<thead>
<tr>
<th>Resource</th>
<th>Example Cloud Template</th>
</tr>
</thead>
<tbody>
<tr>
<td>vSphere machine with multiple NICs</td>
<td>resources:</td>
</tr>
<tr>
<td></td>
<td>demo-machine:</td>
</tr>
<tr>
<td></td>
<td>type: Cloud.vSphere.Machine</td>
</tr>
<tr>
<td></td>
<td>properties:</td>
</tr>
<tr>
<td></td>
<td>image: ubuntu</td>
</tr>
<tr>
<td></td>
<td>flavor: small</td>
</tr>
<tr>
<td></td>
<td>networks:</td>
</tr>
<tr>
<td></td>
<td>- network: '${network-01.name}'</td>
</tr>
<tr>
<td></td>
<td>deviceIndex: 0</td>
</tr>
<tr>
<td></td>
<td>- network: '${network-02.name}'</td>
</tr>
<tr>
<td></td>
<td>deviceIndex: 1</td>
</tr>
<tr>
<td></td>
<td>network-01:</td>
</tr>
<tr>
<td></td>
<td>type: Cloud.vSphere.Network</td>
</tr>
<tr>
<td></td>
<td>properties:</td>
</tr>
<tr>
<td></td>
<td>name: network-01</td>
</tr>
<tr>
<td></td>
<td>network-02:</td>
</tr>
<tr>
<td></td>
<td>type: Cloud.vSphere.Network</td>
</tr>
<tr>
<td></td>
<td>properties:</td>
</tr>
<tr>
<td></td>
<td>name: network-02</td>
</tr>
<tr>
<td></td>
<td>vSphere machine with an attached tag in vCenter</td>
</tr>
<tr>
<td>Resource</td>
<td>Example Cloud Template</td>
</tr>
<tr>
<td>----------</td>
<td>------------------------</td>
</tr>
</tbody>
</table>
| vSphere machine with a customization spec | resources:  
  demo-machine:  
  type: Cloud.vSphere.Machine  
  properties:  
  name: demo-machine  
  image: ubuntu  
  flavor: small  
  customizationSpec: Linux |
| vSphere machine with remote access | inputs:  
  username:  
  type: string  
  title: Username  
  description: Username  
  default: testUser  
  password:  
  type: string  
  title: Password  
  default: VMware@123  
  encrypted: true  
  description: Password for the given username  
  resources:  
  demo-machine:  
  type: Cloud.vSphere.Machine  
  properties:  
  flavor: small  
  imageRef: >-  
  https://cloud-images.ubuntu.com/releases/16.04/release-20170307/ubuntu-16.04-server-cloudimg-amd64.ova  
  cloudConfig: |  
  ssh_pwauth: yes  
  chpasswd:  
  list: |  
  ${input.username}:${input.password}  
  expire: false  
  users:  
  - default  
  - name: ${input.username}  
  lock_passwd: false  
  sudo: ['ALL=(ALL) NOPASSWD:ALL']  
  groups: [wheel, sudo, admin]  
  shell: '/bin/bash'  
  runcmd:  
  - echo "Defaults:${input.username} ! requiretty" >> /etc/sudoers.d/${input.username} |

Cores per socket and CPU count in Cloud Assembly

Cloud Assembly template code lets you specify a number of cores per socket for a vSphere machine resource.

You can specify the number of cores per virtual socket or the total number of sockets. For example, your licensing terms might restrict software that is licensed per socket or available operating systems might only recognize a certain number of sockets so that additional CPUs must be provisioned as additional cores.
Add the `coreCount` property to a cloud template in the vSphere machine resource.

The `coreCount` value must be less than or equal to the CPU count (`cpuCount`) value specified in the flavor mapping or in the vSphere machine resource code in the cloud template. For related information, see Setting the number of cores per CPU in a virtual machine (1010184).

The `coreCount` property is optional and available only for vSphere machine resources.

An example vSphere machine resource snippet is shown below.

```yaml
Cloud_vSphere_Machine_1:
  type: Cloud.vSphere.Machine
  properties:
    cpuCount: 8
    coreCount: 4
```

Additional information about sockets and cores per socket settings is available in blog article Virtual Machine vCPU and vNUMA Rightsizing – Guidelines.

**Networks, security resources, and load balancers in vRealize Automation Cloud**

You can use networking, security, and load balancer resources and settings in cloud template designs and deployments.

For a summary of cloud template design code options, see vRealize Automation Resource Type Schema.

For related information, see:

- More about network resources in vRealize Automation Cloud cloud templates
- More about security group and tag resources in vRealize Automation Cloud cloud templates
- More about load balancer resources in vRealize Automation Cloud cloud templates

These examples illustrate network, security, and load balancer resources within basic cloud template designs.
## Networks

<table>
<thead>
<tr>
<th>Resource scenario</th>
<th>Example cloud template design code</th>
</tr>
</thead>
<tbody>
<tr>
<td>vSphere machine with multiple NICs associated to an NSX network resource.</td>
<td>```</td>
</tr>
<tr>
<td></td>
<td>resources:</td>
</tr>
<tr>
<td></td>
<td>demo-machine:</td>
</tr>
<tr>
<td></td>
<td>type: Cloud.vSphere.Machine</td>
</tr>
<tr>
<td></td>
<td>properties:</td>
</tr>
<tr>
<td></td>
<td>image: ubuntu</td>
</tr>
<tr>
<td></td>
<td>flavor: small</td>
</tr>
<tr>
<td></td>
<td>networks:</td>
</tr>
<tr>
<td></td>
<td>- network: '${resource.Cloud_vSphere_Network_1.id}'</td>
</tr>
<tr>
<td></td>
<td>Cloud_vSphere_Network_1:</td>
</tr>
<tr>
<td></td>
<td>type: Cloud.vSphere.Network</td>
</tr>
<tr>
<td></td>
<td>properties:</td>
</tr>
<tr>
<td></td>
<td>networkType: existing</td>
</tr>
<tr>
<td></td>
<td>Cloud_vSphere_Network_2:</td>
</tr>
<tr>
<td></td>
<td>type: Cloud.NSX.Network</td>
</tr>
<tr>
<td></td>
<td>properties:</td>
</tr>
<tr>
<td></td>
<td>networkType: existing</td>
</tr>
<tr>
<td></td>
<td>Add a private network with a static IP address for an Azure VM deployment</td>
</tr>
<tr>
<td></td>
<td>formatVersion: 1</td>
</tr>
<tr>
<td></td>
<td>inputs: {}</td>
</tr>
<tr>
<td></td>
<td>resources:</td>
</tr>
<tr>
<td></td>
<td>Cloud_Azure_Machine_1:</td>
</tr>
<tr>
<td></td>
<td>type: Cloud.Azure.Machine</td>
</tr>
<tr>
<td></td>
<td>properties:</td>
</tr>
<tr>
<td></td>
<td>image: photon</td>
</tr>
<tr>
<td></td>
<td>flavor: Standard_B1ls</td>
</tr>
<tr>
<td></td>
<td>networks:</td>
</tr>
<tr>
<td></td>
<td>- network: '${resource.Cloud_Network_1.id}'</td>
</tr>
<tr>
<td></td>
<td>assignment: static</td>
</tr>
<tr>
<td></td>
<td>address: 10.0.0.45</td>
</tr>
<tr>
<td></td>
<td>assignPublicIpAddress: false</td>
</tr>
<tr>
<td></td>
<td>Cloud_Network_1:</td>
</tr>
<tr>
<td></td>
<td>type: Cloud.Network</td>
</tr>
<tr>
<td></td>
<td>properties:</td>
</tr>
<tr>
<td></td>
<td>networkType: existing</td>
</tr>
<tr>
<td></td>
<td>Using and Managing VMware Cloud Assembly</td>
</tr>
</tbody>
</table>
Add or edit NAT and DNAT port forwarding rules in a Cloud.NSX.NAT resource for an existing deployment.

```resources:
gw:
  type: Cloud.NSX.NAT
  properties:
    networks:
      - '${resource.akout.id}'
    natRules:
      - translatedInstance: '${resource.centos.networks[0].id}'
        index: 0
        protocol: TCP
        kind: NAT44
        type: DNAT
        sourceIPs: any
        sourcePorts: 80
        translatedPorts: 8080
        destinationPorts: 8080
        description: edit
      - translatedInstance: '${resource.centos.networks[0].id}'
        index: 1
        protocol: TCP
        kind: NAT44
        type: DNAT
        sourceIPs: any
        sourcePorts: 90
        translatedPorts: 9090
        destinationPorts: 9090
        description: add

centos:
  type: Cloud.vSphere.Machine
  properties:
    image: WebTinyCentOS65x86
    flavor: small
    customizationSpec: Linux
    networks:
      - network: '${resource.akout.id}'
        assignment: static

akout:
  type: Cloud.NSX.Network
  properties:
    networkType: outbound
    constraints:
      - tag: nsxt-nat-1-M2```

Public cloud machine to use an internal IP instead of a public IP. This example uses a specific network ID.

Note: The network: option is used in the networks: setting to specify a target network ID. The name: option in the networks: setting has been deprecated and should not be used.

```resources:
wf_proxy:
  type: Cloud.Machine
  properties:
    image: ubuntu 16.04
    flavor: small
    constraints:
      - tag: 'platform:vsphere'
    networks:
      - network: '${resource.wf_net.id}'
        assignPublicIpAddress: false```
<table>
<thead>
<tr>
<th>Resource scenario</th>
<th>Example cloud template design code</th>
</tr>
</thead>
</table>
| Routed network for NSX-V or NSX-T using the NSX network resource type.            | Cloud_NSX_Network_1:  
  type: Cloud.NSX.Network  
  properties:  
    networkType: routed                                                                                                                                                                                                          |
| Add a tag to a machine NIC resource in the cloud template.                       | formatVersion: 1  
  inputs: {}  
  resources:  
    Cloud_Machine_1:  
      type: Cloud.vSphere.Machine  
      properties:  
        flavor: small  
        image: ubuntu  
        networks:  
          - name: '${resource.Cloud_Network_1.name}'  
            deviceIndex: 0  
            tags:  
              - key: 'nic0'  
                value: null  
              - key: internal  
                value: true  
          - name: '${resource.Cloud_Network_2.name}'  
            deviceIndex: 1  
            tags:  
              - key: 'nic1'  
                value: null  
              - key: internal  
                value: false                                                                                                                                                     |
| Tag NSX-T logical switches for an outbound network.                              | Cloud_NSX_Network_1:  
  type: Cloud.NSX.Network  
  properties:  
    networkType: outbound  
  tags:  
    - key: app  
      value: opencart                                                                                                                                                                                                               |
| Tagging is supported for NSX-T and VMware Cloud on AWS.                          |                                                                                                                                                                                                                                |
| For more information on this scenario, see community blog post Creating Tags in NSX with Cloud Assembly.                                                                                                                             |
Security groups

<table>
<thead>
<tr>
<th>Resource scenario</th>
<th>Example cloud template design code</th>
</tr>
</thead>
</table>
| **Existing security group with a constraint tag applied to a machine NIC.**       | ```formatVersion: 1
inputs: {}
resources:
  allowSsh_sg:
    type: Cloud.SecurityGroup
    properties:
      securityGroupType: existing
      constraints:
        - tag: allowSsh
  compute:
    type: Cloud.Machine
    properties:
      image: centos
      flavor: small
      networks:
        - network: '${resource.prod-net.id}'
      securityGroups:
        - '${resource.allowSsh_sg.id}'
prod-net:
  type: Cloud.Network
  properties:
    networkType: existing``` |
| To use an existing security group, enter `existing` for the `securityGroupType` property. |                                                                                                                                                                |
| You can assign tags to a `Cloud.SecurityGroup` resource to allocate existing security groups by using tag constraints. Security groups that do not contain tags cannot be used in the cloud template design. |                                                                                                                                                                |
| Constraint tags must be set for `securityGroupType: existing` `security group` resources. Those constraints must match the tags set on the existing security groups. Constraint tags cannot be set for `securityGroupType: new` `security group resources`. |                                                                                                                                                                |
| **On-demand security group with two firewall rules illustrating the Allow and Deny access options.** | ```resources:
Cloud_SecurityGroup_1:
  type: Cloud.SecurityGroup
  properties:
    securityGroupType: new
    rules:
      - ports: 5000
        source:
        'fc00:10:0000:0000:0000:56ff:fe89:48b4'
        access: Allow
        direction: inbound
        name: allow_5000
        protocol: TCP
      - ports: 7000
        source:
        'fc00:10:0000:0000:0000:56ff:fe89:48b4'
        access:Deny
        direction: inbound
        name: deny_7000
        protocol: TCP
Cloud_vSphere_Machine_1:
  type: Cloud.vSphere.Machine
  properties:
    image: photon
    cpuCount: 1
    totalMemoryMB: 256
    networks:
      - network: '${resource.Cloud_Network_1.id}'
assignIPv6Address: true``` |
Complex cloud template with 2 security groups, including:
- 1 existing security group
- 1 on-demand security group with multiple firewall rule examples
- 1 vSphere machine
- 1 existing network

This sample illustrates different combinations of protocols and ports, services, IP CIDR as source and destination, IP range as source or destination, and the options for any, IPv6, and (::/0).

For machine NICs, you can specify the connected network, and security group(s). You can also specify the NIC index or an IP address.

<table>
<thead>
<tr>
<th>Resource scenario</th>
<th>Example cloud template design code</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>assignment: static securityGroups:</td>
</tr>
<tr>
<td></td>
<td>- '${resource.Cloud_SecurityGroup_1.id}'</td>
</tr>
<tr>
<td></td>
<td>Cloud_Network_1: type: Cloud.Network properties: networkType: existing</td>
</tr>
</tbody>
</table>

```
formatVersion: 1
inputs: {}
resources:
  DEMO_ESG: existing security group - security group 1
    type: Cloud.SecurityGroup
    properties:
      constraints:
        - tag: BlockAll
      securityGroupType: existing (designation of existing for security group 1)
  DEMO_ODSG: (on-demand security group - security group 2)
    type: Cloud.SecurityGroup
    properties:
      rules: (multiple firewall rules in this section)
      - name: IN-ANY (rule 1)
        source: any
        service: any
        direction: inbound
        access: Deny
      - name: IN-SSH (rule 2)
        source: any
        service: SSH
        direction: inbound
        access: Allow
      - name: IN-SSH-IP (rule 3)
        source: 33.33.33.1-33.33.33.250
        protocol: TCP
        ports: 223
        direction: inbound
        access: Allow
      - name: IPv-6-ANY-SOURCE (rule 4)
        source: '::/0'
        protocol: TCP
        ports: 223
        direction: inbound
        access: Allow
      - name: IN-EXISTING-SG (rule 6)
        source: '${resource["DEMO_ESG"].id}'
        protocol: ICMPv6
        direction: inbound
        access: Allow
      - name: OUT-ANY (rule 7)
        destination: any
        service: any
      ```
### Resource scenario

<table>
<thead>
<tr>
<th>Example cloud template design code</th>
</tr>
</thead>
<tbody>
<tr>
<td>direction: outbound</td>
</tr>
<tr>
<td>access: Deny</td>
</tr>
<tr>
<td>- name: OUT-TCP-IPv6 (rule 8)</td>
</tr>
<tr>
<td>destination:</td>
</tr>
<tr>
<td>'2001:0db8:85a3::8a2e:0370:7334/64'</td>
</tr>
<tr>
<td>protocol: TCP</td>
</tr>
<tr>
<td>ports: 22</td>
</tr>
<tr>
<td>direction: outbound</td>
</tr>
<tr>
<td>access: Allow</td>
</tr>
<tr>
<td>- name: IPv6-ANY-DESTINATION (rule 9)</td>
</tr>
<tr>
<td>destination: ':/:0'</td>
</tr>
<tr>
<td>protocol: UDP</td>
</tr>
<tr>
<td>ports: 23</td>
</tr>
<tr>
<td>direction: outbound</td>
</tr>
<tr>
<td>access: Allow</td>
</tr>
<tr>
<td>- name: OUT-UDP-SERVICE (rule 10)</td>
</tr>
<tr>
<td>destination: any</td>
</tr>
<tr>
<td>service: NTP</td>
</tr>
<tr>
<td>direction: outbound</td>
</tr>
<tr>
<td>access: Allow</td>
</tr>
</tbody>
</table>

securityGroupType: new (designation of on-demand for security group 2)

DEMO_VC_MACHINE: (machine resource)

type: Cloud.vSphere.Machine

properties:
- image: PHOTON
- cpuCount: 1
- totalMemoryMB: 1024

networks: (Machine network NICs)
- network: '${resource.DEMO_NW.id}'
  securityGroups:
  - '${resource.DEMO_ODSG.id}'
  - '${resource.DEMO_ESG.id}'

DEMO_NETWORK: (network resource)

type: Cloud.vSphere.Network

properties:
- networkType: existing

constraints:
- tag: nsx62
Load balancers

<table>
<thead>
<tr>
<th>Resource scenario</th>
<th>Example cloud template design code</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specify a load balancer logging level, algorithm, and size.</td>
<td>Sample NSX load balancer showing use of logging level, algorithm, and size:</td>
</tr>
<tr>
<td></td>
<td>resources:</td>
</tr>
<tr>
<td></td>
<td>Cloud_LoadBalancer_1:</td>
</tr>
<tr>
<td></td>
<td>type: Cloud.NSX.LoadBalancer</td>
</tr>
<tr>
<td></td>
<td>properties:</td>
</tr>
<tr>
<td></td>
<td>name: myapp-lb</td>
</tr>
<tr>
<td></td>
<td>network: '${appnet-public.name}'</td>
</tr>
<tr>
<td></td>
<td>instances: '${wordpress.id}'</td>
</tr>
<tr>
<td></td>
<td>routes:</td>
</tr>
<tr>
<td></td>
<td>- protocol: HTTP port: '80'</td>
</tr>
<tr>
<td></td>
<td>loggingLevel: CRITICAL</td>
</tr>
<tr>
<td></td>
<td>algorithm: LEAST_CONNECTION</td>
</tr>
<tr>
<td></td>
<td>type: MEDIUM</td>
</tr>
<tr>
<td>Associate a load balancer with a named machine or a named machine NIC. You can specify either machine ID or machine network ID to add the machine to the load balancer pool. The instances property supports both machines (machine by ID) and NICs (machine by network ID).</td>
<td>You can use the instances property to define a machine ID or a machine network ID:</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Cloud_LoadBalancer_1:</td>
</tr>
<tr>
<td></td>
<td>type: Cloud.LoadBalancer</td>
</tr>
<tr>
<td></td>
<td>properties:</td>
</tr>
<tr>
<td></td>
<td>network: '${resource.Cloud_Network_1.id}'</td>
</tr>
<tr>
<td></td>
<td>instances: '${resource.Cloud_Machine_1.id}'</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Cloud_LoadBalancer_1:</td>
</tr>
<tr>
<td></td>
<td>type: Cloud.LoadBalancer</td>
</tr>
<tr>
<td></td>
<td>properties:</td>
</tr>
<tr>
<td></td>
<td>network: '${resource.Cloud_Network_1.id}'</td>
</tr>
<tr>
<td></td>
<td>instances: '${resource.Cloud_Machine_1.networks[0].id}'</td>
</tr>
<tr>
<td></td>
<td>- resource.Cloud_Machine_1.id</td>
</tr>
<tr>
<td>One machine specified for load balancer inclusion and another machine NIC specified for load balancer inclusion:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>instances:</td>
</tr>
<tr>
<td></td>
<td>- resource.Cloud_Machine_1.id</td>
</tr>
</tbody>
</table>
Add health check settings to an NSX load balancer. Additional options include `httpMethod`, `requestBody`, and `responseBody`.

```
myapp-lb:
  type: Cloud.NSX.LoadBalancer
  properties:
    name: myapp-lb
    network: '${appnet-public.name}'
    instances: '${wordpress.id}'
  routes:
    - protocol: HTTP
      port: '80'
      algorithm: ROUND_ROBIN
      instanceProtocol: HTTP
      instancePort: '80'
      healthCheckConfiguration:
        protocol: HTTP
        port: '80'
        urlPath: /mywordpresssite/wp-admin/install.php
  intervalSeconds: 60
  timeoutSeconds: 10
  unhealthyThreshold: 10
  healthyThreshold: 2
  connectionLimit: '50'
  connectionRateLimit: '50'
  maxConnections: '500'
  minConnections: ''
  internetFacing: true
```
### Resource scenario

<table>
<thead>
<tr>
<th>Example cloud template design code</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>On-demand network with a 1-arm load balancer.</strong></td>
</tr>
</tbody>
</table>
| inputs: {}  
resources:  
  mp-existing:  
    type: Cloud.Network  
    properties:  
      name: mp-existing  
      networkType: existing  
  mp-wordpress:  
    type: Cloud.vSphere.Machine  
    properties:  
      name: wordpress  
      count: 2  
      flavor: small  
      image: tiny  
      customizationSpec: Linux  
    networks:  
      - network: '${resource["mp-private"].id}'  
  mp-private:  
    type: Cloud.NSX.Network  
    properties:  
      name: mp-private  
      networkType: private  
      constraints:  
        - tag: nsxt  
  mp-wordpress-lb:  
    type: Cloud.LoadBalancer  
    properties:  
      name: wordpress-lb  
      internetFacing: false  
      network: '${resource.mp-existing.id}'  
    instances: '${resource["mp-wordpress"]}.id'}  
    routes:  
      - protocol: HTTP  
        port: '80'  
        instanceProtocol: HTTP  
        instancePort: '80'  
        healthCheckConfiguration:  
          protocol: HTTP  
          port: '80'  
          urlPath: /index.pl  
          intervalSeconds: 60  
          timeoutSeconds: 30  
          unhealthyThreshold: 5  
          healthyThreshold: 2  
| **Existing network with a load balancer.** |
| formatVersion: 1  
inputs:  
  count:  
    type: integer  
    default: 1  
resources:  
  ubuntu-vm:  
    type: Cloud.Machine  
    properties:  
      name: ubuntu  
      flavor: small  
      image: tiny  
    count: '${input.count}'  
    networks:  
      - network: '${resource.mp-existing.id}'}
Learn more

For network and security group implementation scenarios, see VMware blogs such as these:

- **vRealize Automation Cloud Assembly Load Balancer with NSX-T Deep Dive**
- **Network Automation with Cloud Assembly and NSX – Part 1** (includes use of NSX-T and vCenter cloud accounts and network CIDR)
- **Network Automation with Cloud Assembly and NSX – Part 2** (includes use of existing and outbound network types)
- **Network Automation with Cloud Assembly and NSX – Part 3** (includes use of existing and on-demand security groups)
- **Network Automation with Cloud Assembly and NSX – Part 4** (includes use of existing and on-demand load balancers)

**More about network resources in vRealize Automation Cloud cloud templates**

As you create or edit your vRealize Automation Cloud cloud templates, use the most appropriate network resources for your objectives. Learn about the NSX and cloud-agnostic network options that are available in the cloud template.

Select one of the available network resource types based on machine and related conditions in your vRealize Automation Cloud cloud template.
Cloud agnostic network resource

You add a cloud agnostic network by using the **Cloud Agnostic > Network** resource on the cloud template **Design** page. The resource displays in the cloud template code as a `Cloud.Network` resource type. The default resource displays as:

```yaml
Cloud_Network_1:
  type: Cloud.Network
  properties:
    networkType: existing
```

Use a cloud agnostic network when you want to specify networking characteristics for a target machine type that is not, or might not, be connected to an NSX network.

The cloud agnostic network resource is available for these resource types:

- Cloud agnostic machine
- vSphere
- Google Cloud Platform (GCP)
- Amazon Web Services (AWS)
- Microsoft Azure
- VMware Cloud on AWS (VMC)

The cloud agnostic network resource is available for these network type (`networkType`) settings:

- public
- private
- outbound
- existing

vSphere network resource

You add a vSphere network by using the **vSphere > Network** resource on the cloud template **Design** page. The resource displays in the cloud template code as a `Cloud.vSphere.Network` resource type. The default resource displays as:

```yaml
Cloud_vSphere_Network_1:
  type: Cloud.vSphere.Network
  properties:
    networkType: existing
```

Use a vSphere network when you want to specify networking characteristics for a vSphere machine type (`Cloud.vSphere.Machine`).

The vSphere network resource is only available for a `Cloud.vSphere.Machine` machine type.

The vSphere resource is available for these network type (`networkType`) settings:

- public
private

existing

For examples, see Using network settings in network profiles and cloud templates in vRealize Automation Cloud.

**NSX network resource**

You add an NSX network by using the **NSX > Network** resource on the cloud template **Design** page. The resource displays in the cloud template code as a **Cloud.NSX.Network** resource type. The default resource displays as:

```
Cloud_NSX_Network_1:
  type: Cloud.NSX.Network
  properties:
    networkType: existing
```

Use an NSX network when you want to attach a network resource to one or more machines that have been associated to an NSX-V or NSX-T cloud account. The NSX network resource allows you to specify NSX networking characteristics for a vSphere machine resource that is associated to an NSX-V or NSX-T cloud account.

The **Cloud.NSX.Network** resource is available for these network type (**networkType**) settings:

- public
- private
- outbound
- existing
- routed - Routed networks are only available for NSX-V and NSX-T.

If you want multiple outbound or routed networks to share the same NSX-T Tier-1 router or NSX-V Edge Service Gateway (ESG), connect a single NSX gateway resource (**Cloud.NSX.Gateway**) to the connected networks in the template prior to initial deployment. If you add the gateway after deployment as a Day 2 or iterative development operation, each network creates its own router.

You can use the NSX NAT resource in the template to support NAT and DNAT port forwarding rules.

**Cloud agnostic network resource with Azure, AWS, or GCP deployment intent**

Public cloud provider VMs can require specific cloud template property combinations that are not necessarily required in NSX or vSphere-based machine deployments. For examples of cloud template code that support some of these scenarios, see Networks, security resources, and load balancers in vRealize Automation Cloud.
NSX gateway resource

You can reuse or share a single NSX-T Tier-1 router or NSX-V Edge Service Gateway (ESG) within a single deployment by using a gateway resource (Cloud.NSX.Gateway) in the cloud template. The gateway resource represents the Tier-1 or ESG and can be connected to multiple networks in the deployment. The gateway resource can be used with outbound or routed networks only.

The Cloud.NSX.Gateway resource allows you to share the NSX-T Tier-1 router or NSX-V Edge Service Gateway (ESG) among connected outbound or routed networks in a deployment.

The gateway is often attached to a single outbound or routed network. However, if the gateway is attached to multiple networks, the networks must be of the same type, for example all outbound or all routed. The gateway can be connected to multiple machines or load balancers that are connected to the same outbound or routed networks. The gateway must be connected to a load balancer on the shared on-demand network so that it can reuse the NSX-T Tier-1 router or NSX-V Edge Service Gateway (ESG) created by the gateway.

To allow multiple outbound or routed networks to share the same T1 router or Edge, connect a single Cloud.NSX.Gateway gateway resource to all the networks initially. All the intended networks and the single gateway must be connected together before you deploy the cloud template, otherwise each network creates its own router.

For an NSX network that contains an associated compute gateway resource, the gateway settings are applied to all associated networks in the deployment. A single NSX-T Tier-1 logical router is created for each deployment and shared by all on-demand networks and load balancers in the deployment. A single NSX-V Edge is created for each deployment and shared by all the on-demand networks and load balancers in the deployment.

You can attach the gateway resource to a network as an iterative deployment update. However, doing so does not create a T1 or Edge router - the initial network deployment creates the router.

For NSX-T networks that do not use an associated gateway resource, multiple on-demand networks in the cloud template continue to create multiple Tier-1 logical routers in the deployment.

If the gateway contains NAT rules, you can reconfigure or delete the NAT or DNAT rules for the Tier 1 router or Edge router. If the gateway is initially deployed with no NAT rules, it has no available Day 2 actions.

NSX NAT resource

The Cloud.NSX.NAT resource allows DNAT rules and port forwarding to be attached to all the connected outbound networks by way of the gateway resource. You can attach a NAT resource to a gateway resource for which the DNAT rules need to be configured.

Note The Cloud.NSX.Gateway resource was originally available for DNAT rules. However, use of the Cloud.NSX.Gateway as a means of defining DNAT rules and port forwarding has been deprecated. It does remain available for backward compatibility. Use the Cloud.NSX.NAT cloud template resource for DNAT rules and port forwarding. A warning appears in the cloud template if you attempt to use the Cloud.NSX.Gateway resource type with NAT rule specifications.
The **Cloud.NSX.NAT** resource supports DNAT rules and port forwarding when connected to an outbound NSX-V or NSX-T network.

The NAT rules setting in the resource is `natRules`. You can attach the NAT resource to the gateway resource to configure the `natRules` entries on the gateway. DNAT rules that are specified in the resource use the associated machines or load balancers as their target.

You can reconfigure a machine NIC or compute gateway in an existing deployment to modify its `natRules` settings by adding, reordering, editing or deleting DNAT port forwarding rules. You cannot use DNAT rules with clustered machines. You can specify DNAT rules for individual machines within the cluster as part of a Day 2 operation.

**External IPAM integration options**

For information about properties that are available for use with your Infoblox IPAM integrations in cloud template designs and deployments, see [Using Infoblox-specific properties and extensible attributes for IPAM integrations in vRealize Automation Cloud cloud templates](#).

**Available day 2 operations**

For a list of common day 2 operations that are available for cloud template and deployment resources, see [What actions can I run on Cloud Assembly deployments](#).

For an example of how to move from one network to another, see [How to move a deployed machine to another network](#).

**Learn more**

For related information and examples, see [Networks, security resources, and load balancers in vRealize Automation Cloud](#).

For information about defining network resources, see [Network resources in vRealize Automation Cloud](#).

For information about defining network profiles, see [Learn more about network profiles in vRealize Automation Cloud](#).

For examples of cloud template designs that illustrate sample network resources and settings, see [Networks, security resources, and load balancers in vRealize Automation Cloud](#).

**More about security group and tag resources in vRealize Automation Cloud cloud templates**

As you create or edit your vRealize Automation Cloud cloud templates, use the most appropriate security resource options to meet your objectives.
Cloud agnostic security group resource

You add a security group resource by using the **Cloud Agnostic > Security Group** resource on the cloud template Design page. The resource displays in the cloud template code as a `Cloud.SecurityGroup` resource type. The default resource displays as:

```yaml
Cloud_SecurityGroup_1:
  type: Cloud.SecurityGroup
  properties:
    constraints: []
    securityGroupType: existing
```

You specify a security group resource in a cloud template design as either existing (`securityGroupType: existing`) or on-demand (`securityGroupType: new`).

You can add an existing security group to your cloud template or you can use an existing security group that has been added to a network profile.

For NSX-V and NSX-T, as well as NSX-T with the policy manager switch enabled in combination with VMware Cloud on AWS, you can add an existing security group or define a new security group as you design or modify your cloud template. On-demand security groups are supported for NSX-T and NSX-V, and VMware Cloud on AWS when used with NSX-T policy manager.

For all cloud account types except Microsoft Azure, you can associate one or more security groups to a machine NIC. A Microsoft Azure virtual machine NIC (`machineName`) can only be associated to one security group.

By default, the security group property `securityGroupType` is set to `existing`. To create an on-demand security group, enter `new` for the `securityGroupType` property. To specify firewall rules for an on-demand security group, use the `rules` property in the `Cloud.SecurityGroup` section of the security group resource.

**Existing security groups**

Existing security groups are created in a source cloud account resource such as NSX-T or Amazon Web Services. They are data collected by vRealize Automation Cloud from the source. You can select an existing security group from a list of available resources as part of a vRealize Automation Cloud network profile. In a cloud template design, you can specify an existing security group either inherently by its membership in a specified network profile or specifically by name using the `securityGroupType: existing` setting in a security group resource. If you add a security group to a network profile, add at least one capability tag to the network profile. On-demand security group resources require a constraint tag when used in a cloud template design.

You can associate a security group resource in your cloud template design to one or more machine resources.

**Note** If you intend to use a machine resource in your cloud template design to provision to a Microsoft Azure virtual machine NIC (`machineName`), you should only associate the machine resource to a single security group.
On-demand security groups

You can define on-demand security groups as you define or modify a cloud template design by using the `securityGroupType: new` setting in the security group resource code.

You can use an on-demand security group for NSX-V and NSX-T, as well as Amazon Web Services when used with NSX-T Policy type, to apply a specific set of firewall rules to a networked machine resource or set of grouped resources. Each security group can contain multiple named firewall rules. You can use an on-demand security group to specify services or protocols and ports. Note that you can specify either a service or a protocol but not both. You can specify a port in addition to a protocol. You cannot specify a port if you specify a service. If the rule contains neither a service or a protocol, the default service value is Any.

You can also specify IP addresses and IP ranges in firewall rules. Some firewall rule examples are shown in Networks, security resources, and load balancers in vRealize Automation Cloud.

When you create firewall rules in an NSX-V or NSX-T on-demand security group, the default is to allow the specified network traffic but to also allow other network traffic. To control network traffic, you must specify an access type for each rule. The rule access types are:

- **Allow** (default) - Allows the network traffic that is specified in this firewall rule.
- **Deny** - Blocks the network traffic that is specified in this firewall rule. Actively tells the client that the connection is rejected.
- **Drop** - Rejects the network traffic that is specified in this firewall rule. Silently drops the packet as if the listener is not online.

For an example design that uses an `access: Allow` and an `access: Deny` firewall rule, see Networks, security resources, and load balancers in vRealize Automation Cloud.

**Note** A cloud administrator can create a cloud template design that contains only an NSX on-demand security group and can deploy that design to create a reusable existing security group resource that members of the organization can add to network profiles and cloud template designs as an existing security group.

Firewall rules support either IPv4 or IPv6 format CIDR values for source and destination IP addresses. For an example design that uses IPv6 CIDR values in a firewall rule, see Networks, security resources, and load balancers in vRealize Automation Cloud.

On-demand and existing security groups for VMware Cloud on AWS

You can define an on-demand security group for a VMware Cloud on AWS machine in a cloud template by using the `securityGroupType: new` setting in the security group resource code.

A sample code snippet for an on-demand security group is shown below:

```yaml
resources:
  Cloud_SecurityGroup_1:
    type: Cloud.SecurityGroup
    properties:
      name: vmc-odsg
```
You can also define an existing security group for a networked VMware Cloud on AWS machine and optionally include constraint tagging, as shown in the following examples:

```yaml
Cloud_SecurityGroup_2:
  type: Cloud.SecurityGroup
  properties:
    constraints: [xyz]
    securityGroupType: existing

Cloud_SecurityGroup_3:
  type: Cloud.SecurityGroup
  properties:
    securityGroupType: existing
    constraints:
      - tag: xyz
```

Iterative cloud template development is supported.

- If a security group is associated with one or more machines in the deployment, a delete action displays a message stating that the security group cannot be deleted.

- If a security group is not associated with any machine in the deployment, a delete action displays a message stating that the security group will be deleted from this deployment and the action cannot be undone. An existing security group is deleted from the cloud template, while an on-demand security group is destroyed.

**Using NSX-V security tags and NSX-T VM tags**

You can see and use NSX-V security tags and NSX-T and NSX-T with Policy VM tags from managed resources in vRealize Automation Cloud cloud templates.

NSX-V and NSX-T security tags are supported for use with vSphere. NSX-T security tags are also supported for use with VMware Cloud on AWS.

**Note**  As with VMs deployed to vSphere, you can configure machine tags for a VM to be deployed on VMware Cloud on AWS. You can also update the machine tag after initial deployment. These machine tags allow vRealize Automation Cloud to dynamically assign a VM to an appropriate NSX-T security group during deployment.
You can specify NSX-V security tags by using the `key: nsxSecurityTag` and a tag value in the compute resource in the cloud template, as shown in the following example:

```yaml
- key: nsxSecurityTag
  value: security_tag_1
- key: nsxSecurityTag
  value: security_tag_2
```

The specified value must correspond to an NSX-V security tag. If there are no security tags in NSX-V that match the specified `nsxSecurityTag` key value, the deployment will fail.

NSX-T does not have a separate security tag. Any tag specified on the compute resource in the cloud template results in the deployed VM being associated with all tags that are specified in NSX-T. For NSX-T, including NSX-T with Policy, VM tags are also expressed as a key value pair in the cloud template. The `key` setting equates to the `scope` setting in NSX-T and the `value` setting equates to the `Tag Name` specified in NSX-T.

To avoid confusion, do not use a `nsxSecurityTag` key pairs when for NSX-T. If you specify an `nsxSecurityTag` key value pair for use with NSX-T, including NSX-T with Policy, the deployment creates a VM tag with an empty Scope setting with a Tag name that matches the `value` specified. When you view such tags in NSX-T, the Scope column will be empty.

**Using app isolation policies in on-demand security group firewall rules**

You can use an app isolation policy to only allow internal traffic between the resources that are provisioned by the cloud template. With app isolation, the machines provisioned by the cloud template can communicate with each other but cannot connect outside the firewall. You can create an app isolation policy in the network profile. You can also specify app isolation in a cloud template design by using an on-demand security group with a Deny firewall rule or a private or outbound network.

An app isolation policy is created with a lower precedence. If you apply multiple policies, the policies with the higher weight will take precedence.

When you create an application isolation policy, an auto-generated policy name is generated. The policy is also made available for reuse in other cloud template designs and iterations that are specific to the associated resource endpoint and project. The app isolation policy name is not visible in the cloud template but it is visible as a custom property on the project page (Infrastructure > Administration > Projects) after the cloud template design is deployed.

For the same associated endpoint in a project, any deployment that requires an on-demand security group for app isolation can use the same app isolation policy. Once the policy is created, it is not deleted. When you specify an app isolation policy, vRealize Automation Cloud searches for the policy within the project and relative to the associated endpoint - If it finds the policy it reuses it, if it does not find the policy, it creates it. The app isolation policy name is only visible after its initial deployment in the project's custom properties listing.
Using security groups in iterative cloud template development

When changing security group constraints during iterative development, where the security group is not associated to a machine in the cloud template, the security group updates in the iteration as specified. However, when the security group is already associated to a machine, redeployment fails. You must detach existing security groups and/or `securityGroupType` resource properties from associated machines during iterative cloud template development and then re-associate between each redeployment. The needed workflow is as follows, assuming that you have initially deployed the cloud template.

1. In the Cloud Assembly template designer, detach the security group from all its associated machines in the cloud template.

2. Redeploy the template by clicking **Update an existing deployment**.

3. Remove the existing security group constraint tags and/or `securityGroupType` properties in the template.

4. Add new security group constraint tags and/or `securityGroupType` properties in the template.

5. Associate the new security group constraint tags and/or `securityGroupType` property instances to the machines in the template.

6. Redeploy the template by clicking **Update an existing deployment**.

Available day 2 operations

For a list of common day 2 operations that are available for cloud template and deployment resources, see [What actions can I run on Cloud Assembly deployments](#).

Learn more

For information about using a security group for network isolation, see [Security resources in vRealize Automation Cloud](#).

For information about using security groups in network profiles, see [Learn more about network profiles in vRealize Automation Cloud](#) and [Using security group settings in network profiles and cloud template designs in vRealize Automation Cloud](#).

For examples of using security groups in cloud templates, see [Networks, security resources, and load balancers in vRealize Automation Cloud](#).

**More about load balancer resources in vRealize Automation Cloud cloud templates**

As you create or edit your vRealize Automation Cloud cloud templates, use the most appropriate load balancer resources for your objectives.

You can use NSX and cloud-agnostic load balancer resources in a cloud template to control load balancing in a deployment.
The cloud-agnostic load balancer can be deployed across multiple clouds. A cloud-specific load balancer can specify advanced settings and features that are available only to a specific cloud/topology. Cloud-specific properties are available in the NSX load balancer (Cloud.NSX.LoadBalancer) resource type. If you add these properties on a cloud-agnostic load balancer (Cloud.LoadBalancer), they are ignored if, for example, an Amazon Web Services or Microsoft Azure load balancer is provisioned, but are respected if an NSX-V or NSX-T load balancer is provisioned. Choose one of the available load balancer resource types based on conditions in your vRealize Automation Cloud cloud template.

You cannot connect a load balancer resource directly to a security group resource in the design canvas.

Cloud agnostic load balancer resource

Use a cloud agnostic load balancer when you want to specify networking characteristics for any type of target machine.

You add a cloud agnostic load balancer by using the Cloud Agnostic > Load Balancer resource on the cloud template design page. The resource displays in the cloud template code as a Cloud.LoadBalancer resource type. The default resource displays as:

```plaintext
Cloud_LoadBalancer_1:
  type: Cloud.LoadBalancer
  properties:
    routes: []
    network: ''
    instances: []
    internetFacing: false
```

NSX load balancer resource

Use an NSX load balancer when your cloud template contains characteristics that are specific to NSX-V or NSX-T (either Policy API or Manager API methods). You can attach one or more load balancers to an NSX-V or NSX-T network or to machines that are associated to an NSX-V or NSX-T network.

You add an NSX load balancer by using the NSX > Load Balancer resource. The resource displays in the cloud template code as a Cloud.NSX.LoadBalancer resource type. The default resource displays as:

```plaintext
Cloud_NSX_LoadBalancer_1:
  type: Cloud.NSX.LoadBalancer
  properties:
    routes: []
    network: ''
    instances: []
```
Load balancer options in cloud template code

Adding one or more load balancer resources to your cloud template allows you to specify the following settings. Some examples are available at Networks, security resources, and load balancers in vRealize Automation Cloud.

**Note** On-demand load balancers do not support the HTTPS protocol. When using an on-demand load balancer, the HTTP protocol is used.

- **Machine specification**
  You can specify named machine resources to participate in a load balancing pool. Alternatively you can specify that a specific machine NIC participate in the load balancer pool.
  This option is available for the **NSX load balancer resource** *(Cloud.NSX.LoadBalancer)* only.

  - `resource.Cloud_Machine_1.id`
    Specifies that the load balancer include the machine identified in the cloud template code as *Cloud_Machine_1*.

    Specifies that the load balancer only include the machine identified in the cloud template code as *Cloud_Machine_2* when it is deployed to machine NIC *Cloud_Machine_2.networks[2]*.

- **Logging level**
  The logging level value specifies a severity level for the error log. The options are NONE, EMERGENCY, ALERT, CRITICAL, ERROR, WARNING, INFO, DEBUG, and NOTICE. The logging level value applies to all load balancers in the cloud template. This option is specific to NSX. For load balancers that have a parent, the parent logging level setting overrides any logging level setting in its children.

  For related information, see topics such as **Add Load Balancers** in NSX product documentation.

- **Type**
  Use a load balancer type to specify a scaling size. The default is small. This option is specific to NSX. For load balancers that have a parent, the parent type setting overrides any type setting in its children.

  - **Small**
    Correlates to compact in NSX-V and small in NSX-T.

  - **Medium**
    Correlates to large in NSX-V and medium in NSX-T.

  - **Large**
Correlates to quad-large in NSX-V and large in NSX-T.

- Extra Large
  Correlates to xlarge in NSX-V and large in NSX-T.
  For related information, see topics such as Scaling Load Balancer Resources in NSX product documentation.

This option is available for the NSX load balancer resource (Cloud.NSX.LoadBalancer).

- Algorithm (server pool)

Use an algorithm balancing method to control how incoming connections are distributed among the server pool members. The algorithm can be used on a server pool or directly on a server. All load balancing algorithms skip servers that meet any of the following conditions:

- The Admin state is set to DISABLED.
- The Admin state is set to GRACEFUL_DISABLED and there is no matching persistence entry.
- The active or passive health check state is DOWN.
- The connection limit for the maximum server pool concurrent connections is reached.

This option is specific to NSX.

- IP_HASH
  Selects a server based on a hash of the source IP address and the total weight of all the running servers.
  Correlates to IP-HASH in NSX-V and NSX-T.

- LEAST_CONNECTION
  Distributes client requests to multiple servers based on the number of connections already on the server. New connections are sent to the server with the fewest connections. Ignores the server pool member weights even if they are configured.
  Correlates to LEASTCONN in NSX-V and LEAST_CONNECTION in NSX-T.

- ROUND_ROBIN
  Incoming client requests are cycled through a list of available servers capable of handling the request. Ignores the server pool member weights even if they are configured. Default.
  Correlates to ROUND_ROBIN in NSX-V and NSX-T.

- WEIGHTED_LEAST_CONNECTION
  Each server is assigned a weight value that signifies how that server performs relative to other servers in the pool. The value determines how many client requests are sent to a server compared to other servers in the pool. This load balancing algorithm focuses on using the weight value to distribute the load among the available server resources fairly. By default, the weight value is 1 if the value is not configured and slow start is enabled.
Correlates to WEIGHTED_LEAST_CONNECTION in NSX-T. There is no correlation in NSX-V.

- WEIGHTED_ROUND_ROBIN

Each server is assigned a weight value that signifies how that server performs relative to other servers in the pool. The value determines how many client requests are sent to a server compared to other servers in the pool. This load balancing algorithm focuses on fairly distributing the load among the available server resources.

Correlates to WEIGHTED_ROUND_ROBIN in NSX-T. There is no correlation in NSX-V.

- URI

The left part of the URI is hashed and divided by the total weight of the running servers. The result designates which server receives the request. This ensures that a URI is always directed to the same server if no server goes up or down. The URI algorithm parameter has two options uriLength=<len> and uriDepth=<dep>. The length parameter range should be \( 1 \leq \text{len} < 256 \). The depth parameter range should be \( 1 \leq \text{dep} < 10 \). Length and depth parameters are followed by a positive integer number. These options can balance servers based on the beginning of the URI only. The length parameter indicates that the algorithm should only consider the defined characters at the beginning of the URI to compute the hash. The depth parameter indicates the maximum directory depth to be used to compute the hash. One level is counted for each slash in the request. If both parameters are specified, the evaluation stops when either is reached.

Correlates to URI in NSX-V. There is no correlation in NSX-T.

- HTTPHEADER

HTTP header name is looked up in each HTTP request. The header name in parentheses is not case-sensitive. If the header is absent or does not contain any value, the round robin algorithm is applied. The HTTPHEADER algorithm parameter has one option headerName=<name>.

Correlates to HTTPHEADER in NSX-V. There is no correlation in NSX-T.

- URL

URL parameter specified in the argument is looked up in the query string of each HTTP GET request. If the parameter is followed by an equal sign = and a value, then the value is hashed and divided by the total weight of the running servers. The result designates which server receives the request. This process is used to track user identifiers in requests and ensure that a same user ID is always sent to the same server as long as no server goes up or down. If no value or parameter is found, then a round robin algorithm is applied. The URL algorithm parameter has one option urlParam=<url>.

Correlates to URL in NSX-V. There is no correlation in NSX-T.

For related information, see topics such as Add a Server Pool for Load Balancing in NSX product documentation.
Health monitor

Use the health monitor options to test whether a server is available. Active health monitoring for HTTP, ICMP, TCP, and UDP protocols is supported. Passive health monitoring is available for NSX-T only.

This option is specific to NSX.

- **httpMethod**

  HTTP method to use to detect server status for the health check request. Methods are GET, HOST, OPTIONS, HEAD, or PUT.

- **requestBody**

  Health check request body content. Used, and required, by HTTP, TCP, and UDP protocols.

- **responseBody**

  Health check expected response body content. If the received string matches this response body, the server is considered healthy. Used, and required, by HTTP, TCP, and UDP protocols.

  **Note** If you use the UDP monitor protocol, the UDP Data Sent and UDP Data Expected parameters are required. The requestBody and responseBody properties map to these parameters.

This option is available for the NSX load balancer resource (Cloud.NSX.LoadBalancer).

For related information, see topics such as Configure an Active Health Monitor in NSX product documentation.

Health check

Use health check options to specify how the load balancer performs its health checks.

This option is only available for the NSX load balancer resource (Cloud.NSX.LoadBalancer).

See Networks, security resources, and load balancers in vRealize Automation Cloud for a sample of available health check settings.

**NSX-V and NSX-T network types and load balancer options**

Load balancer options depend on the network that the load balancer resource is associated to in the cloud template. You can configure a load balancer relative to the network type and network conditions.

- **On-demand network**
If the load balancer computes are attached to an on-demand network, a new Tier-1 router is created and attached to the Tier-0 router specified in the network profile. The load balancer is then attached to the Tier-1 router. The Tier-1 router VIP advertisement is enabled if the VIP is on an existing network. If an on-demand network is configured for DHCP, the on-demand network and load balancer share the Tier-1 router.

- **Existing network**

If the load balancer is attached to an existing network, the load balancer is created with the Tier-1 router of the existing network. A new load balancer is created if there is no load balancer attached to the Tier-1 router. If the load balancer already exists, new virtual servers are attached to it. If the existing network is not attached to a Tier-1 router, a new Tier-1 router is created and attached to a Tier-0 router defined in the network profile, the Tier-1 router VIP advertisement is not enabled.

vRealize Automation Cloud does not support an NSX-T two-arm load balancer (inline load balancer) on two different existing networks. Note that in a two-arm load balancer scenario, the VIP uplink is on an existing network while the pool member machines are connected to an on-demand network. To specify load balancing when using an existing network, you must configure a one-arm load balancer where the same existing network is used for the load balancer VIP and the pool member machines. However, if you are using a load balancer that you’ve selected in the network profile, you can load balance between machines on two different existing networks if there is connectivity between the two existing networks.

- **Network isolation defined in the network profile**

For network types of outbound or private, you can specify network isolation settings in a network profile to emulate a new security group. Because machines are attached to an existing network and isolation settings are defined in the profile, this option is similar to a load balancer created on an existing network. The difference is that to enable the data path, the Tier-1 uplink port IP is added to the isolation security group.

You can specify load balancer settings for NSX-associated networks by using an NSX load balancer resource in the cloud template design.

To learn more, see VMware blog post **vRA Cloud Assembly Load Balancer with NSX-T Deep Dive**.

**Reconfiguring logging level or type settings when multiple load balancers share an NSX-T Tier 1 or NSX-V Edge**

When using a cloud template that contains multiple load balancers which share a Tier-1 router in the NSX-T endpoint or an Edge router in the NSX-V endpoint, reconfiguring the logging level or type settings in one of the load balancer resources does not update the settings for the other load balancers. Mismatched settings cause inconsistencies in NSX. To avoid inconsistencies when reconfiguring these logging level and/or type settings, use the same reconfiguration values for all the load balancer resources in the cloud template which share a Tier 1 or Edge in their associated NSX endpoint.
Available day 2 operations

When you scale in or scale out a deployment that contains a load balancer, the load balancer is configured to include newly added machines or to stop load balancing machines that are targeted for tear down.

For a list of common day 2 operations that are available for cloud templates and deployments, see What actions can I run on Cloud Assembly deployments.

Learn more

For information about defining load balancer settings in a network profile, see Learn more about network profiles in vRealize Automation Cloud.

For examples of cloud template designs that include load balancers, see Networks, security resources, and load balancers in vRealize Automation Cloud.

Puppet-enabled cloud template with username and password access

In this example, you add Puppet configuration management to a cloud template deployed on a vCenter compute resource with username and password access.

This procedure shows an example of how you might create a Puppet enabled deployable resource that requires username and password authentication. Username and password access means that the user must manually log in from the compute resource to the Puppet primary machine in order to invoke Puppet configuration management.

Optionally, you can configure remote access authentication which sets up configuration management in a cloud template so that the compute resource handles authentication with the Puppet primary machine. With remote access enabled, the compute resource automatically generates a key to satisfy password authentication. A valid username is still required.

See AWS Puppet configuration management cloud template examples and vCenter Puppet configuration cloud template examples for more examples of how you can configure different Puppet scenarios in Cloud Assembly blueprints.

Prerequisites

- Set up a Puppet Enterprise instance on a valid network.
- Add your Puppet Enterprise instance to Cloud Assembly using the Integrations feature. See Configure Puppet Enterprise integration in Cloud Assembly.
- Set up a vSphere account and a vCenter compute resource.
Procedure

1. Add a Puppet configuration management component to a vSphere compute resource on the canvas for the desired cloud template.
   
   a. Select **Infrastructure > Manage > Integrations**.
   
   b. Click **Add Integration** and select Puppet.
   
   c. Enter the appropriate information on the Puppet configuration page.

<table>
<thead>
<tr>
<th>Configuration</th>
<th>Description</th>
<th>Example Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hostname</td>
<td>Host name or IP address of the Puppet primary machine</td>
<td>Puppet-Ubuntu</td>
</tr>
<tr>
<td>SSH Port</td>
<td>SSH port for communication between Cloud Assembly and Puppet primary machine. (Optional)</td>
<td>NA</td>
</tr>
<tr>
<td>Autosign secret</td>
<td>The shared secret configured on the Puppet primary machine that nodes should provide to support autosign certificate requests.</td>
<td>User specific</td>
</tr>
<tr>
<td>Location</td>
<td>Indicate whether the Puppet primary machine is on a private or public cloud.</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Note</strong> Cross cloud deployment is supported only if there is connectivity between the deployment compute resource and the Puppet primary machine.</td>
<td></td>
</tr>
<tr>
<td>Cloud proxy</td>
<td>Not required for public cloud accounts, such as Microsoft Azure or Amazon Web Services. If you are using a vCenter based cloud account, select the appropriate cloud proxy for your account.</td>
<td>NA</td>
</tr>
<tr>
<td>Username</td>
<td>SSH and RBAC user name for Puppet primary machine.</td>
<td>User specific. YAML value is '$ (input.username)'</td>
</tr>
<tr>
<td>Password</td>
<td>SSH and RBAC password for Puppet primary machine.</td>
<td>User specific YAML value is '$ (input.password)'</td>
</tr>
<tr>
<td>Use sudo commands for this user</td>
<td>Select to use sudo commands for the procidd.</td>
<td>true</td>
</tr>
<tr>
<td>Name</td>
<td>Puppet primary machine name.</td>
<td>PEMasterOnPrem</td>
</tr>
<tr>
<td>Description</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

2. Add the username and password properties to the Puppet YAML as shown in the following example.

3. Ensure that the value for the remoteAccess property to the Puppet cloud template YAML is set to **authentication: username and password** as shown in the example below.
Example: vCenter username and password YAML code

The following example shows the representative YAML code for adding username and password authentication on a vCenter compute resource.

```yaml
inputs:
  username:
    type: string
    title: Username
    description: Username to use to install Puppet agent
    default: puppet
  password:
    type: string
    title: Password
    default: VMware@123
    encrypted: true
    description: Password for the given username to install Puppet agent
resources:
  Puppet-Ubuntu:
    type: Cloud.vSphere.Machine
    properties:
      flavor: small
      imageRef: >-
        https://cloud-images.ubuntu.com/releases/16.04/release-20170307/ubuntu-16.04-server-cloudimg-amd64.ova
      remoteAccess:
        authentication: usernamePassword
        username: '${input.username}'
        password: '${input.password}'
  Puppet_Agent:
    type: Cloud.Puppet
    properties:
      provider: PEMasterOnPrem
      environment: production
      role: 'role::linux_webserver'
      username: '${input.username}'
      password: '${input.password}'
      host: '${Puppet-Ubuntu.*}'
      useSudo: true
      agentConfiguration:
        certName: '${Puppet-Ubuntu.address}'
```

AWS Puppet configuration management cloud template examples

There are several options for configuring cloud templates to support Puppet based configuration management on AWS compute resources.
## Puppet management on AWS with username and password

### Inputs
- **Username**
  - Type: string
  - Title: Username
  - Default: puppet
- **Password**
  - Type: string
  - Title: Password
  - Encrypted: true
  - Default: VMware@123

### Resources
- **Webserver**
  - Type: Cloud.AWS.EC2.Instance
  - Properties:
    - Flavor: small
    - Image: centos
  - CloudConfig:
    ```yaml
    #cloud-config
    ssh_pwauth: yes
    chpasswd:
      list:
        - ${input.username}:${input.password}
      expire: false
    users:
      - name: ${input.username}
        lock_passwd: false
        sudo: ['(ALL=ALL NOPASSWD:ALL)']
        groups: [wheel, sudo, admin]
        shell: '/bin/bash'
        ssh-authorized-keys:
          - ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQDytVL+Q6+/vGbmkXoRpX dmettem@dmettem-m01.vmware.com
    runcmd:
      - echo "Defaults:${input.username} !requiretty"
    
    >> /etc/sudoers.d/${input.username}
    ```
- **Puppet_Agent**
  - Type: Cloud.Puppet
  - Properties:
    - Provider: PEOnAWS
    - Environment: production
    - Role: 'role::linux_webserver'
    - Host: '${Webserver.*}'
    - OsType: linux
    - Username: '${input.username}'
    - Password: '${input.password}'
    - UseSudo: true

### Example of... Authentication of cloud configuration on any supported Amazon Machine Image.
Example of... Sample Blueprint YAML

```yaml
resources:
  Webserver:
    type: Cloud.AWS.EC2.Instance
    properties:
      flavor: small
      image: centos
      cloudConfig: |
        #cloud-config
        runcmd: |
          - sudo sed -e 's/.*PasswordAuthentication no.*/PasswordAuthentication yes/' -i /etc/ssh/sshd_config
          - sudo service sshd restart
  Puppet_Agent:
    type: Cloud.Puppet
    properties:
      provider: PEOnAWS
      environment: production
      role: 'role::linux_webserver'
      host: '${Webserver.*}'
      osType: linux
      username: '${input.username}'
      password: '${input.password}'
      useSudo: true
```

Puppet management on AWS with generated PublicPrivateKey

```yaml
remoteAccess.authentication
authentication on AWS
with generatedPublicPrivateKey acces.
```

```yaml
inputs: {}
resources:
  Machine:
    type: Cloud.AWS.EC2.Instance
    properties:
      flavor: small
      imageRef: ami-a4dc46db
      remoteAccess:
        authentication: generatedPublicPrivateKey
  Puppet_Agent:
    type: Cloud.Puppet
    properties:
      provider: puppet-BlueprintProvisioningITSuite
      environment: production
      role: 'role::linux_webserver'
      host: '${Machine.*}'
      osType: linux
      username: ubuntu
      useSudo: true
      agentConfiguration:
        runInterval: 15m
        certName: '${Machine.address}'
      useSudo: true
```

vCenter Puppet configuration cloud template examples

There are several options for configuring cloud templates to support Puppet based configuration management on vCenter compute resources.
Puppet on vSphere with username and password authentication

The following example shows example YAML code for Puppet on a vSphere OVA with username and password authentication.
### Table 6-4.

<table>
<thead>
<tr>
<th>Example of...</th>
<th>Sample Blueprint YAML</th>
</tr>
</thead>
</table>
| **YAML code for Puppet on a vSphere OVA with username and password authentication.** | inputs:  
  username:  
    type: string  
    title: Username  
    default: puppet  
  password:  
    type: string  
    title: Password  
    encrypted: true  
    default: VMware@123  
  resources:  
    Puppet_Agent:  
      type: Cloud.Puppet  
      properties:  
        provider: PEonAWS  
        environment: dev  
        role: 'role::linux_webserver'  
        username: '${input.username}'  
        password: '${input.password}'  
        useSudo: true  
        host: '${Webserver.{'  
        osType: linux  
        agentConfiguration:  
          runInterval: 15m  
          certName: '${Machine.address}'  
    Webserver:  
      type: Cloud.vSphere.Machine  
      properties:  
        cpuCount: 1  
        totalMemoryMB: 1024  
        imageRef: https://cloud-images.ubuntu.com/releases/16.04/release-20170307/ubuntu-16.04-server-cloudimg-amd64.ova  
      cloudConfig: |  
        #cloud-config  
        ssh_pwauth: yes  
        chpasswd:  
          list: |  
            ${input.username}:${input.password}  
          expire: false  
          users:  
            - default  
            - name: ${input.username}  
              lock_passwd: false  
              sudo: ['ALL=(ALL) NOPASSWD:ALL']  
              groups: [wheel, sudo, admin]  
              shell: '/bin/bash'  
          ssh-authorized-keys:  
            - ssh-rsa  
            AAAAB3NzaC1yc2EAAAADQABAAABABQdYtVL+Q6+vGbmkX0RrX  
            dmettem@dmettem-m01.vmware.com  
          runcmd:  
            - echo "$Defaults:${input.username}" |

| **YAML code for Puppet on a vSphere OVA with username and password authentication on the compute resource.** | inputs:  
  username:  
    type: string  
    title: Username  
    default: puppet |
### Table 6-4. (continued)

<table>
<thead>
<tr>
<th>Example of...</th>
<th>Sample Blueprint YAML</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>password:</strong></td>
<td>type: string</td>
</tr>
<tr>
<td></td>
<td>title: Password</td>
</tr>
<tr>
<td></td>
<td>encrypted: true</td>
</tr>
<tr>
<td></td>
<td>default: VMware@123</td>
</tr>
<tr>
<td><strong>resources:</strong></td>
<td></td>
</tr>
<tr>
<td>Puppet_Agent:</td>
<td>type: Cloud.Puppet</td>
</tr>
<tr>
<td></td>
<td>properties:</td>
</tr>
<tr>
<td></td>
<td>provider: PEonAWS</td>
</tr>
<tr>
<td></td>
<td>environment: dev</td>
</tr>
<tr>
<td></td>
<td>role: 'role::linux_webserver'</td>
</tr>
<tr>
<td></td>
<td>username: '${input.username}'</td>
</tr>
<tr>
<td></td>
<td>password: '${input.password}'</td>
</tr>
<tr>
<td></td>
<td>useSudo: true</td>
</tr>
<tr>
<td></td>
<td>host: '${Webserver. *}'</td>
</tr>
<tr>
<td></td>
<td>osType: linux</td>
</tr>
<tr>
<td></td>
<td>agentConfiguration:</td>
</tr>
<tr>
<td></td>
<td>runInterval: 15m</td>
</tr>
<tr>
<td></td>
<td>certName: '$(Machine.address)'</td>
</tr>
<tr>
<td>Webserver:</td>
<td>type: Cloud.vSphere.Machine</td>
</tr>
<tr>
<td></td>
<td>properties:</td>
</tr>
<tr>
<td></td>
<td>cpuCount: 1</td>
</tr>
<tr>
<td></td>
<td>totalMemoryMB: 1024</td>
</tr>
</tbody>
</table>
|               | imageRef: $<-
|               | https://cloud-images.ubuntu.com/releases/16.04/
|               | release-20170307/ubuntu-16.04-server-cloudimg-amd64.ova
|               | cloudConfig: |
|               | #cloud-config
|               | ssh_pwauth: yes         |
|               | chpasswd:               |
|               | list: |                  |
|               | ${input.username}:${input.password} |
|               | expire: false           |
|               | users:                  |
|               | - default               |
|               | - name: ${input.username}
|               | lock_passwd: false      |
|               | sudo: ['ALL=(ALL) NOPASSWD:ALL'] |
|               | groups: [wheel, sudo, admin]
|               | shell: '/bin/bash'      |
|               | ssh-authorized-keys:    |
|               | - ssh-rsa               |
|               | AAAAB3NzaClyc2EAAAADAQABAAABAQAQDytdVL+Q6+vGbkX0RpX |
|               | dmettem@dmettem-m01.vmware.com |
|               | runcmd:                 |
|               | - echo "Defaults:${input.username}" |

**YAML code for Puppet on a vCenter with remote access enabled password authentication on the compute resource.**

**inputs:**

- **username:**
  - type: string
  - title: Username
  - description: Username to use to install Puppet agent
  - default: puppet

- **password:**
  - type: string
  - title: Password
  - default: VMware@123
  - encrypted: true
### Table 6-4. (continued)

<table>
<thead>
<tr>
<th>Example of...</th>
<th>Sample Blueprint YAML</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>description: Password for the given username to install Puppet agent</td>
</tr>
<tr>
<td></td>
<td>resources:</td>
</tr>
<tr>
<td></td>
<td>Puppet-Ubuntu:</td>
</tr>
<tr>
<td></td>
<td>type: Cloud.vSphere.Machine</td>
</tr>
<tr>
<td></td>
<td>properties:</td>
</tr>
<tr>
<td></td>
<td>flavor: small</td>
</tr>
<tr>
<td></td>
<td>imageRef: &gt;- <a href="https://cloud-images.ubuntu.com/releases/16.04/release-20170307/ubuntu-16.04-server-cloudimg-amd64.ova">https://cloud-images.ubuntu.com/releases/16.04/release-20170307/ubuntu-16.04-server-cloudimg-amd64.ova</a></td>
</tr>
<tr>
<td></td>
<td>remoteAccess:</td>
</tr>
<tr>
<td></td>
<td>authentication: usernamePassword</td>
</tr>
<tr>
<td></td>
<td>username: '${input.username}'</td>
</tr>
<tr>
<td></td>
<td>password: '${input.password}'</td>
</tr>
<tr>
<td></td>
<td>Puppet_Agent:</td>
</tr>
<tr>
<td></td>
<td>type: Cloud.Puppet</td>
</tr>
<tr>
<td></td>
<td>properties:</td>
</tr>
<tr>
<td></td>
<td>provider: PEMasterOnPrem</td>
</tr>
<tr>
<td></td>
<td>environment: production</td>
</tr>
<tr>
<td></td>
<td>role: 'role::linux_webserver'</td>
</tr>
<tr>
<td></td>
<td>username: '${input.username}'</td>
</tr>
<tr>
<td></td>
<td>password: '${input.password}'</td>
</tr>
<tr>
<td></td>
<td>host: '${Puppet-Ubuntu.*}'</td>
</tr>
<tr>
<td></td>
<td>osType: linux</td>
</tr>
<tr>
<td></td>
<td>username: ubuntu</td>
</tr>
<tr>
<td></td>
<td>useSudo: true</td>
</tr>
<tr>
<td></td>
<td>agentConfiguration:</td>
</tr>
<tr>
<td></td>
<td>certName: '${Puppet-Ubuntu.address}'</td>
</tr>
</tbody>
</table>

Puppet on vSphere with generated PublicPrivateKey authentication

### Table 6-5.

<table>
<thead>
<tr>
<th>Example of...</th>
<th>Sample Blueprint YAML</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>inputs: {}</td>
</tr>
<tr>
<td></td>
<td>resources:</td>
</tr>
<tr>
<td></td>
<td>Machine:</td>
</tr>
<tr>
<td></td>
<td>type: Cloud.vSphere.Machine</td>
</tr>
<tr>
<td></td>
<td>properties:</td>
</tr>
<tr>
<td></td>
<td>flavor: small</td>
</tr>
<tr>
<td></td>
<td>imageRef: &gt;- <a href="https://cloud-images.ubuntu.com/releases/16.04/release-20170307/ubuntu-16.04-server-cloudimg-amd64.ova">https://cloud-images.ubuntu.com/releases/16.04/release-20170307/ubuntu-16.04-server-cloudimg-amd64.ova</a></td>
</tr>
<tr>
<td></td>
<td>remoteAccess:</td>
</tr>
<tr>
<td></td>
<td>authentication: generatedPublicPrivateKey</td>
</tr>
<tr>
<td></td>
<td>Puppet_Agent:</td>
</tr>
<tr>
<td></td>
<td>type: Cloud.Puppet</td>
</tr>
<tr>
<td></td>
<td>properties:</td>
</tr>
<tr>
<td></td>
<td>provider: puppet-BlueprintProvisioningITSuite</td>
</tr>
<tr>
<td></td>
<td>environment: production</td>
</tr>
<tr>
<td></td>
<td>role: 'role::linux_webserver'</td>
</tr>
<tr>
<td></td>
<td>host: '${Machine.*}'</td>
</tr>
<tr>
<td></td>
<td>osType: linux</td>
</tr>
<tr>
<td></td>
<td>username: ubuntu</td>
</tr>
<tr>
<td></td>
<td>useSudo: true</td>
</tr>
<tr>
<td></td>
<td>agentConfiguration:</td>
</tr>
<tr>
<td></td>
<td>runInterval: 15m</td>
</tr>
<tr>
<td></td>
<td>certName: '${Machine.address}'</td>
</tr>
<tr>
<td></td>
<td>- echo &quot;Defaults:${input.username}&quot;</td>
</tr>
</tbody>
</table>
vRealize Automation Cloud resource property schema

The vRealize Automation Cloud infrastructure-as-code editor lets you click or hover for syntax and code completion help. To view the complete set of cloud template resource properties though, sometimes called custom properties, refer to the consolidated resource schema.

The schema is available from the VMware {code} site. Follow the link, and click Models to list the resource objects that are available for cloud templates, formerly called blueprints.

- vRealize Automation Resource Type Schema on VMware {code}

Special Cloud Assembly properties

Cloud Assembly supports a small number of properties that might be useful outside of production environments or in other special situations. The properties do not appear in the schema.

**Caution** The following properties should only be applied in cases where guest OS customization isn't being tested or expected.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>awaitIp</td>
<td>By default, vRealize Automation Cloud provisioning status isn't reported as Finished until the guest OS is fully powered on and configuration has completed. Use of awaitIp: false allows provisioning to finish even though full configuration did not occur. CAUTION: Use of this setting completes the provisioning process sooner but might result in an unconfigured machine with no IP address.</td>
</tr>
<tr>
<td>awaitHostName</td>
<td>Similar to awaitIp, use of awaitHostName: false allows provisioning to finish even though the machine might not have been configured with a host name.</td>
</tr>
</tbody>
</table>

Other ways to create Cloud Assembly templates

In addition to building a Cloud Assembly template from a blank canvas, you can take advantage of existing code.

Cloud template cloning

To clone a template, go to Design, select a source, and click Clone. You clone a cloud template to create a copy based on the source, then assign the clone to a new project or use it as starter code for a new application.
Uploading and downloading

You can upload, download, and share cloud template YAML code in any way that makes sense for your site. You can even modify template code using external editors and development environments.

**Note** A good way to validate shared template code is to inspect it in the Cloud Assembly code editor on the design page.

The Marketplace

To jumpstart your resource library, get files from the Cloud Assembly Marketplace. The Marketplace provides finished cloud templates and open virtualization images.

Accessing the Marketplace

In Cloud Assembly, select **Infrastructure > Connections > Integrations**. Click **Add Integration**, click **My VMware**, and provide your My VMware account credentials.

Downloading and using Marketplace cloud template files

In the **Marketplace** tab, click **Get**, and accept the cloud template EULA. Then, you can add the template to a Cloud Assembly project, or simply download it. You can upload a cloud template in the **Design** tab.

For a project-based example, imagine that you are a project administrator for a Big Data effort. To assist your team, you locate a Marketplace Hadoop template that you add to the team project. You then customize the cloud template for your resource environment, and release it. Then, you import the template into the Service Broker catalog so that your team can deploy it.

Downloading and using Marketplace image files

In the **Marketplace** tab, click **Get**, and accept the OVF or OVA image EULA. Afterward, you can download the OVF or OVA image and reference it in cloud template code.

Continuing with the previous example, your team might need access to a version of Hadoop itself. You download a Hadoop OVF and add it to cloud account resources such as a vCenter Server Content Library. You then update any template code that needs to point to the OVF image.

Extending and automating application life cycles with extensibility

You can extend your application life cycles by using either extensibility actions or vRealize Orchestrator workflows with extensibility subscriptions.
With Cloud Assembly Extensibility, you can assign an extensibility action or vRealize Orchestrator workflow to an event by using subscriptions. When the specified event occurs, the subscription initiates the action or workflow to run, and all subscribers are notified.

**Extensibility Actions**

Extensibility actions are small, lightweight scripts of code used to specify an action and how that action is to perform. You can import extensibility actions from pre-defined Cloud Assembly action templates or from a ZIP file. You can also use the action editor to create custom scripts for your extensibility actions. When multiple action scripts are linked together in one script, you create an action flow. By using action flows, you can create a sequence of actions. For information on using action flows, see *What is an action flow*.

**vRealize Orchestrator Workflows**

By integrating Cloud Assembly with your existing vRealize Orchestrator environment, you can use workflows in your extensibility subscriptions.

**Extensibility action subscriptions**

You can assign an extensibility action to a Cloud Assembly subscription to extend your application life cycle.

**Note** The following subscriptions are use case examples and do not cover all extensibility action functionality.

**How do I integrate Cloud Assembly with ServiceNow using extensibility actions**

Using extensibility actions you can integrate Cloud Assembly with an Enterprise ITSM, like ServiceNow.
Enterprise users commonly integrate their Cloud Management Platform with an IT Service Management (ITSM) and Configuration Management Database (CMDB) platform for compliance. Following this example, you can integrate Cloud Assembly with ServiceNow for CMDB and ITSM by using extensibility action scripts.

**Note** You can also integrate ServiceNow with Cloud Assembly by using vRealize Orchestrator workflows. For information about integrating ServiceNow by using workflows, see [How do I integrate Cloud Assembly for ITSM with ServiceNow using vRealize Orchestrator workflows](#).

To create this integration, you use four extensibility action scripts. The first three scripts are initiated in sequence during provisioning, at the compute provision post event. The fourth script triggers at the compute removal post event.

For more information on event topics, refer to [Event topics provided with Cloud Assembly](#).

---

**Get VM Details**

The Get VM details script acquires additional payload details required for CI creation and an identity token that is stored in Amazon Web Services Systems Manager Parameter Store (SSM). Also, this script updates `customProperties` with additional properties for later use.

**Create ServiceNow CMDB CI**

The Create ServiceNow CMDB CI script passes the ServiceNow instance URL as an input and stores the instance in SSM to meet security requirements. This script also reads the ServiceNow CMDB unique record identifier response (sys_id). It passes it as an output and writes the custom property `serviceNowSysId` during creation. This value is used to mark the CI as retired when the instance is destroyed.

**Note** Additional permissions might need to be allocated to your cloud automation service Amazon Web Services role to allow Lambda to access the SSM Parameter Store.

**Create ServiceNow Change**

This script finishes the ITSM integration by passing the ServiceNow instance URL as an input and storing the ServiceNow credentials as SSM to meet security requirements.
The retire ServiceNow CMDB CI script prompts the ServiceNow to stop and marks the CI as retired based on the custom property `serviceNowSysId` that was created in the creation script.

**Prerequisites**

- Before configuring this integration, filter all event subscriptions with the conditional cloud template property: `event.data["customProperties"]["enable_servicenow"] === "true"

  **Note** This property exists on cloud templates that require a ServiceNow integration.

- Download and install Python.

For more information on filtering subscriptions, see .Create an extensibility subscription.

**Procedure**

1. Open a command-line prompt from your Virtual Machine.
2. Run the Get VM details script.

```python
from botocore.vendored import requests
import json
import boto3

client = boto3.client('ssm','ap-southeast-2')

def handler(context, inputs):
    baseUri = inputs['url']
    casToken = client.get_parameter(Name="casToken",WithDecryption=True)

    url = baseUri + "/iaas/login"
    headers = {
        "Accept": "application/json",
        "Content-Type": "application/json"
    }
    payload = {
        "refreshToken": casToken['Parameter']['Value']
    }

    results = requests.post(url,json=payload,headers=headers)

    bearer = "Bearer 
    bearer = bearer + results.json()['token']

    deploymentId = inputs['deploymentId']
    resourceId = inputs['resourceIds'][0]

    print("deploymentId: "+ deploymentId)
    print("resourceId:" + resourceId)

    machineUri = baseUri + "/iaas/machines/" + resourceId
    headers = {
        "Accept": "application/json",
        "Content-Type": "application/json",
        "Authorization": bearer
    }
    resultMachine = requests.get(machineUri,headers=headers)
    print("machine: " + resultMachine.text)

    print( "serviceNowCPUCount: "+ json.loads(resultMachine.text)['customProperties']["cpuCount"] )
    print( "serviceNowMemoryInMB: "+ json.loads(resultMachine.text)['customProperties']["memoryInMB"] )
```
#update customProperties
outputs = {}
outputs['customProperties'] = inputs['customProperties']
outputs['customProperties']['serviceNowCPUCount'] = int(json.loads(resultMachine.text)['customProperties']['cpuCount'])
outputs['customProperties']['serviceNowMemoryInMB'] = json.loads(resultMachine.text)['customProperties']['memoryInMB']
return outputs

3 Run the CMDB configuration item creation action.

```python
from botocore.vendored import requests
import json
import boto3
client = boto3.client('ssm','ap-southeast-2')

def handler(context, inputs):
    snowUser = client.get_parameter(Name="serviceNowUserName",WithDecryption=False)
snowPass = client.get_parameter(Name="serviceNowPassword",WithDecryption=True)
table_name = "cmdb_ci_vmware_instance"
url = "https://" + inputs['instanceUrl'] + "/api/now/table/{0}".format(table_name)
headers = {'Content-type': 'application/json', 'Accept': 'application/json'}
payload = {
    'name': inputs['customProperties']['serviceNowHostname'],
    'cpus': int(inputs['customProperties']['serviceNowCPUCount']),
    'memory': inputs['customProperties']['serviceNowMemoryInMB'],
    'correlation_id': inputs['deploymentId'],
    'disks_size': int(inputs['customProperties']['provisionGB']),
    'location': "Sydney",
    'vcenter_uuid': inputs['customProperties']['vcUuid'],
    'state': 'On',
    'sys_created_by': inputs['__metadata']['userName'],
    'owned_by': inputs['__metadata']['userName']
}
results = requests.post(
    url,
    json=payload,
    headers=headers,
    auth=(snowUser['Parameter']['Value'], snowPass['Parameter']['Value'])
)
print(results.text)

#parse response for the sys_id of CMDB CI reference
if json.loads(results.text)['result']:
    serviceNowResponse = json.loads(results.text)['result']
    serviceNowSysId = serviceNowResponse['sys_id']
    print(serviceNowSysId)

    #update the serviceNowSysId customProperty
```
4 Run the Creation action script.

```python
from botocore.vendored import requests
import json
import boto3
client = boto3.client('ssm','ap-southeast-2')

def handler(context, inputs):
    snowUser = client.get_parameter(Name="serviceNowUserName",WithDecryption=False)
    snowPass = client.get_parameter(Name="serviceNowPassword",WithDecryption=True)
    table_name = "change_request"
    url = "https://" + inputs['instanceUrl'] + "/api/now/table/{0}".format(table_name)
    headers = {'Content-type': 'application/json', 'Accept': 'application/json'}
    payload = {
        'short_description': 'Provision CAS VM Instance'
    }
    results = requests.post(  
        url,  
        json=payload,  
        headers=headers,  
        auth=(snowUser['Parameter']['Value'], snowPass['Parameter']['Value'])
    )
    print(results.text)
```

Results

Cloud Assembly is successfully integrated with ITSM ServiceNow.

What to do next

When desired, you can retire your CI by using the CMDB configuration item retire action:

```python
from botocore.vendored import requests
import json
import boto3
client = boto3.client('ssm','ap-southeast-2')

def handler(context, inputs):
    snowUser = client.get_parameter(Name="serviceNowUserName",WithDecryption=False)
    snowPass = client.get_parameter(Name="serviceNowPassword",WithDecryption=True)
    tableName = "cmdb_ci_vmware_instance"
    sys_id = inputs['customProperties']['serviceNowSysId']
    url = "https://" + inputs['instanceUrl'] + "/api/now\+{0}".format(sys_id)
    headers = {'Content-type': 'application/json', 'Accept': 'application/json'}
    payload = {
        'state': 'Retired'
    }
    results = requests.put(  
        url,
```

Using and Managing VMware Cloud Assembly

VMware, Inc. 564
How do I tag virtual machines during provisioning by using extensibility actions

You can use extensibility actions along with subscriptions to automate and simplify tagging VMs.

As a cloud administrator, you can create deployments that are automatically tagged with specified inputs and outputs by using extensibility actions and extensibility subscriptions. When a new deployment is created against the project containing the tag VM subscription, the deployment event triggers the Tag VM script to run and the tags are automatically applied. This saves time and promotes efficiency while allowing for easier deployment management.

Prerequisites

- Access to cloud administrator credentials.
- Amazon Web Services role for Lambda functions.

Procedure

1. Navigate to **Extensibility** > **Library** > **Actions** > **New Action** and create an action with the following parameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Action Name</td>
<td>Extensibility action name, preferably with <code>TagVM</code> as a prefix or suffix.</td>
</tr>
<tr>
<td>Project</td>
<td>Project to test the extensibility action against.</td>
</tr>
<tr>
<td>Action Template</td>
<td><code>Tag VM</code></td>
</tr>
<tr>
<td>Runtime</td>
<td>Python</td>
</tr>
<tr>
<td>Script Source</td>
<td>Write Script</td>
</tr>
</tbody>
</table>

2. Enter **Handler** as the **Main function**.

3. Add tagging inputs for testing the extensibility action.

   For example, `resourceNames = ["DB_VM"]` and `target = world`.

4. To save your action, click **Save**.

5. To test your action, click **Test**.

6. To exit the action editor, click **Close**.
7 Navigate to Extensibility > Subscriptions.

8 Click New Subscription.

9 Enter the following subscription details.

<table>
<thead>
<tr>
<th>Detail</th>
<th>Setting</th>
</tr>
</thead>
<tbody>
<tr>
<td>Event Topic</td>
<td>Select an event topic related to the tagging phase of the VM. For example, Compute Allocation.</td>
</tr>
<tr>
<td></td>
<td><strong>Note</strong> Tags must be part of the event parameters of the selected event topic.</td>
</tr>
<tr>
<td>Blocking</td>
<td>Set the timeout for the subscription to 1 minute.</td>
</tr>
<tr>
<td>Action/Workflow</td>
<td>Select an extensibility action runnable type, and select your custom extensibility action.</td>
</tr>
</tbody>
</table>

10 To save your custom extensibility action subscription, click Save.

11 Navigate to Design > Cloud Templates, and create a cloud template from a blank canvas.

12 Add two virtual machines to the cloud template: Application_VM and DB_VM.

13 To deploy the VMs, click Deploy.

14 During deployment, verify that the event is initiated and the extensibility action is run.

15 To verify that the tags were applied correctly, navigate to Infrastructure > Resources > Machines.

**Learn more about extensibility actions**

Action-based extensibility uses streamlined scripts of code within Cloud Assembly to automate extensibility actions.

Action-based extensibility provides a lightweight and flexible run-time engine interface where you can define small scriptable actions and configure them to initiate when events specified in extensibility subscriptions occur.
You can create these extensibility action scripts of code within Cloud Assembly, or on your local environment, and assign them to subscriptions. Extensibility action scripts are used for more lightweight and simple automation of tasks and steps. For more information on integrating Cloud Assembly with a vRealize Orchestrator server, see Configure a vRealize Orchestrator integration in Cloud Assembly.

Action-based extensibility provides:

- An alternative to vRealize Orchestrator workflows, using small and reusable scriptable actions, for lightweight integrations and customizations.
- A way to reuse action templates, which contain reusable parameterized actions.

You can create extensibility actions by either writing a user-defined action script code or importing a predefined script code as a .ZIP package. Action-based extensibility supports Node.js, Python, and PowerShell run-time environments. The Node.js and Python run-times rely on Amazon Web Services Lambda. Therefore, you must have an active subscription with Amazon Web Services Identity and Access Management (IAM), and configure Amazon Web Services as an endpoint in Cloud Assembly. For information on getting started with Amazon Web Services Lambda, see ABX: Serverless Extensibility of Cloud Assembly Services.

**Note** To use PowerShell in your extensibility actions, you must have an active on-premises extensibility action integration. See Configure an on-premises action-based extensibility integration in Cloud Assembly.

**Note** Extensibility actions are project-specific.

**How do I create extensibility actions**

With Cloud Assembly, you can create extensibility actions for use in extensibility subscriptions.

Extensibility actions are highly customizable, lightweight, and flexible ways to extend application life cycles by using user-defined script code and action templates. Action templates contain predefined parameters that help set up the foundation of your extensibility action.

There are two methods of creating an extensibility action:

- Writing user-defined code for an extensibility action script.
- Importing a deployment package as a ZIP package for an extensibility action. For information on creating a ZIP package for extensibility actions, see Create a ZIP package for Python runtime extensibility actions, Create a ZIP package for Node.js runtime extensibility actions, or Create a ZIP package for PowerShell runtime extensibility actions.

The following steps describe the procedure for creating an extensibility action that uses Amazon Web Services as a FaaS provider.

**Prerequisites**

- Membership in an active and valid project.
- Configured Amazon Web Services role for Lambda functions. For example, AWSLambdaBasicExecutionRole.
- Cloud administrator role or iam:PassRole permissions enabled.
- To use the PowerShell runtime, configure an on-premises action-based extensibility integration. See Configure an on-premises action-based extensibility integration in Cloud Assembly.

Procedure

1. Select Extensibility > Library > Actions.
2. Click New Action.
3. Enter a name for your action and select a project.
4. (Optional) Add a description for your action.
5. Click Next.
6. Search and select an action template.

**Note** To create a custom action without using an action template, select Custom script.

New configurable parameters appear.
7. Select Write script or Import package.
8. Select the action runtime.
9. Enter an Main function name for the action's entry point.

**Note** For actions imported from a ZIP package, the main function must also include the name of the script file that contains the entry point. For example, if your main script file is titled main.py and your entry point is handler (context, inputs), the name of the main function must be main.handler.

10. Define the input and output parameters of the action.
11. (Optional) Add secrets or extensibility action constants to your default inputs.

**Note** For more information on secrets and extensibility action constants, see How can I create secrets for use in extensibility actions and How can I create extensibility action constants.
12  (Optional) Add application dependencies to the action.

**Note**  For PowerShell scripts, you can define your application dependencies so they are resolved against the PowerShell Gallery repository. To define your application dependencies so, they are resolvable from the public repository use the following format:

```powershell
@{
    Name = 'Version'
}
e.g.

@{
    Pester = '4.3.1'
}
```

**Note**  For actions imported from a ZIP package, application dependencies are added automatically.

13  To define timeout and memory limits, enable the **Set custom timeout and limits** option.

14  To test your action, click **Save** and then **Test**.

**What to do next**

After your extensibility action is created and verified, you can assign it to a subscription.

**Note**  Extensibility subscriptions use the latest released version of an extensibility action. After creating a new version of an action, click **Versions** on the top-right of the editor window. To release the version of the action you want to use in your subscription, click **Release**.

**Create a ZIP package for Python runtime extensibility actions**

You can create a ZIP package that contains the Python script and dependencies used by your VMware Cloud Assembly extensibility actions.

There are two methods of building the script for your extensibility actions:

- Writing your script directly in the extensibility action editor in Cloud Assembly.
- Creating your script on your local environment and adding it, with any relevant dependencies, to a ZIP package.

By using a ZIP package, you can create a custom preconfigured template of action scripts and dependencies that you can import to Cloud Assembly for use in extensibility actions.

You can also use a ZIP package to create extensibility actions that contain multiple Python script files. Using multiple script files can be useful for organizing the structure of your extensibility action code.
Prerequisites

If you are using Python 3.3 or earlier, download and configure the PIP package installer. See Python Package Index.

Procedure

1. On your local machine, create a folder for your action script and dependencies.
   For example, /home/user1/zip-action.
2. Add your main Python action script or scripts to the folder.
   For example, /home/user1/zip-action/main.py.
3. (Optional) Add any dependencies for your Python script to the folder.
   a. Create a requirements.txt file that contains your dependencies. See Requirements Files.
   b. Open a Linux shell.
      
      **Note** The runtime of action-based extensibility in Cloud Assembly is Linux-based. Therefore, any Python dependencies compiled in a Windows environment might make the generated ZIP package unusable for the creation of extensibility actions. Therefore, you must use a Linux shell.
   c. Install your requirements.txt file in the script folder by running the following command:
      
      ```
      pip install -r requirements.txt --target=home/user1/zip-action
      ```
4. In the assigned folder, select your script elements and, if applicable, your requirements.txt file and compress them to a ZIP package.

   **Note** Both your script and dependency elements must be stored at the root level of the ZIP package. When creating the ZIP package in a Linux environment, you might encounter a problem where the package content is not stored at the root level. If you encounter this problem, create the package by running the `zip -r` command in your command-line shell.

   ```
   cd your_script_and_dependencies_folder
   zip -r ../your_action_ZIP.zip *
   ```

What to do next

Use the ZIP package to create an extensibility action script. See How do I create extensibility actions.

Create a ZIP package for Node.js runtime extensibility actions
You can create a ZIP package that contains the Node.js script and dependencies used by your VMware Cloud Assembly extensibility actions.
There are two methods of building the script for your extensibility actions:

- Writing your script directly in the extensibility action editor in Cloud Assembly.
- Creating your script in your local environment and adding it, with any relevant dependencies, to a ZIP package.

By using a ZIP package, you can create a custom preconfigured template of action scripts and dependencies that you can import to Cloud Assembly for use in extensibility actions.

Also, you can use packages to create extensibility actions that contain multiple Node.js script files. Using multiple script files can be useful for organizing the structure of your extensibility action code.

**Procedure**

1. On your local machine, create a folder for your action script and dependencies.
   
   For example, /home/user1/zip-action.

2. Add your main Node.js action script or scripts to the folder.
   
   For example, /home/user1/zip-action/main.js.

3. (Optional) Add any dependencies for your Node.js script to the folder.
   
   
   b. Open a command-line shell.
   
   c. Navigate to the folder that you created for the action script and dependencies.

   ```bash
   cd /home/user1/zip-action
   ```

   d. Install your package.json file in the script folder by running the following command:

   ```bash
   npm install --production
   ```

   **Note** This command creates a node_modules directory in your folder.
In the assigned folder, select your script elements and, if applicable, your `node_modules` directory and compress them to a ZIP package.

**Note** Both your script and dependency elements must be stored at the root level of the ZIP package. When creating the ZIP package in a Linux environment, you might encounter a problem where the package content is not stored at the root level. If you encounter this problem, create the package by running the `zip -r` command in your command-line shell.

```bash
cd your_script_and_dependencies_folder
zip -r ../your_action_ZIP.zip *
```

### What to do next

Use the ZIP package to create an extensibility action script. See [How do I create extensibility actions](#).

**Create a ZIP package for PowerShell runtime extensibility actions**

You can create a ZIP package that contains your PowerShell script and dependency modules for use in extensibility actions.

There are two methods of building the script for your extensibility actions:

- Writing your script directly in the extensibility action editor in Cloud Assembly.
- Creating your script on your local environment and adding it, with any relevant dependencies, to a ZIP package.

By using a ZIP package, you can create a custom preconfigured template of action scripts and dependencies that you can import to Cloud Assembly for use in extensibility actions.

**Note** You do not need to define PowerCLI cmdlets as dependencies or bundle them into a ZIP package. PowerCLI cmdlets come preconfigured with the PowerShell runtime of your Cloud Assembly service.

You can also use a ZIP package to create extensibility actions that contain multiple PowerShell script files. Using multiple script files can be useful for organizing the structure of your extensibility action code.

### Prerequisites

Verify that you are familiar with PowerShell and PowerCLI. You can find a Docker image with PowerShell Core, PowerCLI 10, PowerNSX, and several community modules and script examples at [Docker Hub](https://hub.docker.com/).

### Procedure

1. On your local machine, create a folder for your action script and dependencies.
   
   For example, `/home/user1/zip-action`. 

2 Add your main PowerShell script with a `.psml` extension to the folder.

The following script presents a simple PowerShell function called `main.psml`:

```powershell
function handler($context, $payload) {
    Write-Host "Hello " $payload.target
    return $payload
}
```

**Note** The output of a PowerShell extensibility action is based on the last variable displayed in the body of the function. All other variables in the included function are discarded.

3 (Optional) Add a proxy configuration to your main PowerShell script by using `context` parameters. See Using context parameters to add a proxy configuration in your PowerShell script.

4 (Optional) Add any dependencies for your PowerShell script.

**Note** Your PowerShell dependency script must use the `.psml` extension. Use the same name for the script and the subfolder where the script is saved.

a Log in to a Linux PowerShell shell.

**Note** The runtime of action-based extensibility in Cloud Assembly is Linux-based. Any PowerShell dependencies compiled in a Windows environment might make the generated ZIP package unusable. Any installed third-party dependencies must be compatible with the VMware Photon OS as PowerShell scripts run on Photon OS.

b Navigate to the `/home/user1/zip-action` folder.

c Download and save the PowerShell module containing your dependencies, by running the `Save-Module` cmdlet.

```powershell
Save-Module -Name <module name> -Path ./
```

d Repeat the previous substep for any additional dependency modules.

**Important** Verify that each dependency module is located in a separate subfolder. For more information on writing and managing PowerShell modules, see How to Write a PowerShell Script Module.
5 In the assigned folder, select your script elements and, if applicable, your dependency module subfolders and compress them to a ZIP package.

**Note** Both your script and dependency module subfolders must be stored at the root level of the ZIP package. When creating the ZIP package in a Linux environment, you might encounter a problem where the package content is not stored at the root level. If you encounter this problem, create the package by running the `zip -r` command in your command-line shell.

```
cd your_script_and_dependencies_folder
zip -r ../your_action_ZIP.zip *
```

**What to do next**

Use the ZIP package to create an extensibility action script. See [How do I create extensibility actions](#).

Using context parameters to add a proxy configuration in your PowerShell script

You can enable network proxy communication in your PowerShell script by using context parameters.

Certain PowerShell cmdlets might require that you set a network proxy as an environment variable in your PowerShell function. Proxy configurations are provided to the PowerShell function with the `$context.proxy.host` and `$context.proxy.port` parameters.

You can add these context parameters in the beginning of your PowerShell script.

```powershell
$proxyString = "http://" + $context.proxy.host + ":" + $context.proxy.port
$Env:HTTP_PROXY = $proxyString
$Env:HTTPS_PROXY = $proxyString
```

If the cmdlets support the `-Proxy` parameter, you can also pass the proxy value directly to the specific PowerShell cmdlets.

**Configure cloud-specific extensibility actions**

You can configure extensibility actions to work with your cloud accounts.

When creating an extensibility action, you can configure and link it to various cloud-based accounts:

- Microsoft Azure
- Amazon Web Services

**Prerequisites**

A valid cloud account is required.

**Procedure**

1 Select **Extensibility > Library > Action.**
2 Click **New Action**.

3 Enter the action parameters as necessary.

4 In the **FaaS provider** drop-down menu, select your cloud account provider or select **Auto Select**.

   **Note** If you select **Auto**, the action automatically defines the FaaS provider.

5 Click **Save**.

**Results**

Your extensibility action is linked for use with the configured cloud account.

**Configure on-premises extensibility actions**

You can configure your extensibility actions to use an on-premises FaaS provider instead of an Amazon Web Services or Microsoft Azure cloud account.

By using an on-premises FaaS provider for your extensibility actions, you can use on-premises services like LDAP, CMDB, or vCenter data centers in your Cloud Assembly extensibility subscriptions.

**Prerequisites**

Configure an on-premises extensibility action integration. For more information, see Configure an on-premises action-based extensibility integration in Cloud Assembly.

**Procedure**

1 Select **Extensibility > Library > Actions**.

2 Click **New Action**.

3 Enter a name and project for the extensibility action.

4 (Optional) Enter a description for the extensibility action.

5 Click **Next**.

6 Create or import your extensibility action script.

7 Click the **FaaS provider** drop-down menu and select **On Prem**.

8 To save the new extensibility action, click **Save**.

**What to do next**

Use the created extensibility action in your Cloud Assembly extensibility subscriptions.

**How can I create secrets for use in extensibility actions**

You can add encrypted inputs to your extensibility action by using project level secrets.
With secrets, you can add encrypted input values to your extensibility actions. Encryption is useful for use cases where your inputs are used to manage sensitive data, such as passwords and certificates. Secrets are available for all FaaS providers and runtimes.

**Note** You can also add encrypted input values by using action constants. See How can I create extensibility action constants.

Access to secrets depends on the project that they are created in. Secrets created in Project A, for example, are accessible only to users included in Project A.

Secrets use the `context.getSecret()` function to decrypt the secret value when it is added to your script. This function uses the name of the secret as a parameter. For example, you might use a secret named `abxsecret` as an encrypted input parameter in your action. To add this input parameter to your action script, you must use `context.getSecret(inputs["abxsecret"])`.

**Procedure**

1. Create a new secret.
   a. Navigate to Infrastructure > Administration > Secrets.
   b. Select New Secret.
   c. Enter the name of the project that the secret is assigned to.

   **Note** The extensibility action you want to assign the secret to must be part of the same project as the secret.

   d. Enter a name for your secret.
   e. Enter the value you want to assign to the secret.
   f. (Optional) Enter a description.
   g. Click Create.

2. Add your secret to an extensibility action.
   a. Select an existing extensibility action or create a new extensibility action.
   b. Under Default Inputs, tick the Secret check box.
   c. Search for your secret and add it to the extensibility action inputs.
   d. Add the secret to the script of the extensibility action by using the `context.getSecret()` function.
   e. To test your secret, click Test.

**How can I create extensibility action constants**

You can create and store constants for use in extensibility actions.
With extensibility action constants, you can add encrypted input values to your extensibility actions. Encryption is useful for use cases where your inputs are used to manage sensitive data, such as passwords and certificates. Constants are available for all FaaS providers and runtimes.

**Note**  Unlike secrets, extensibility action constants can only be used for extensibility secrets. For more information on secrets, see *How can I create secrets for use in extensibility actions?*

Extensibility action constants are accessible to all users included in your organization.

Constants use the `context.getSecret()` function to run as part of your script. This function uses the name of constant as a parameter. For example, you might use an extensibility action constant named `abxconstant` as an encrypted input parameter in your action. To add this input parameter to your action script, you must use `context.getSecret(inputs["abxconstant"])`.

**Procedure**

1. Create a extensibility action constant.
   a. Navigate to Extensibility > Library > Actions.
   b. Select Action Constants.
   c. To create a constant, click New Action Constant.
   d. Enter a name and value for the constant, and click Save.

2. Add your constant to a extensibility action.
   a. Select an existing extensibility action or create a new extensibility action.
   b. Under Default Inputs, tick the Secret check box.
   c. Search for your constant and add it to the extensibility action inputs.
   d. Add the constant to the script of the extensibility action by using the `context.getSecret()` function.
   e. To test your extensibility action constant, click Test.

**Create shared extensibility actions**

As a Cloud Assembly administrator, you create extensibility actions that can be shared across projects without exporting and importing the action.

For information on exporting and importing extensibility actions, see *Export and import extensibility actions*.

**Prerequisites**

Create two or more projects in your Cloud Assembly organization.

**Procedure**

1. Select Extensibility > Library > Actions.

2. Click New Action.
3 Enter a name for your extensibility action.
4 (Optional) Enter a description for your extensibility action.
5 Select a project in which your extensibility action is created.
6 Tick the **Share with all projects in this organization** checkbox.
7 Click **Next**.
8 Create or import your action script, and save your extensibility action.

**Note** You can enable or disable sharing from **Settings**. If the extensibility action is used in subscriptions, you cannot disable sharing. To disable sharing, you must remove the extensibility action from your subscriptions.

9 Create an extensibility subscription, add the shared extensibility action, and set the subscription scope to **Any Project**.

**Note** For more information on creating extensibility subscriptions, see [Create an extensibility subscription](#).

The extensibility subscription is triggered by matching events in any of your projects.

**What to do next**

You can also import shared extensibility actions as a content source in the Service Broker catalog. When you select the source project, enter the project that the extensibility action was created in. For more information on adding extensibility actions to Service Broker, see [Add extensibility actions to the Service Broker catalog](#).

**Azure logging for Python-based extensibility actions**

You can now use Microsoft Azure 3.x logging functions in your extensibility action script.

Extensibility actions in Cloud Assembly now use the Microsoft Azure 3.x Scripting API which replaces the previous 1.x version. Microsoft Azure 3.x Scripting API is Linux-based and runs in a container environment.

Because of this version change, logging functions inserted into the script of extensibility actions that use Microsoft Azure as a FaaS (Function as a Service) provider work differently. The next two script samples demonstrate the different logging functions used in the two API versions.

**Microsoft Azure 1.x script sample.**

```python
def handler(context, inputs):
    greeting = "Hello, {0}!".format(inputs["target"])
    print(greeting)

outputs = {
    "greeting": greeting
}
```

```
Microsoft Azure 3.x script sample.

```python
import logging

def handler(context, inputs):
    greeting = "Hello, {0}!".format(inputs["target"])  
    logging.info(greeting)

    outputs = {
        "greeting": greeting
    }

    return outputs
```

The preceding sample demonstrates that the 3.x version adds the `import logging` function at the beginning of the script while replacing the `print()` function with the `logging.info()` function. To continue using logging with extensibility actions created in the Microsoft Azure 1.x API, you must change the logging functions in your script so it matches the Microsoft Azure 3.x sample.

For more information on logging, see the Azure Functions Python developer guide.

**Export and import extensibility actions**

With Cloud Assembly, you can export and import extensibility actions for use in different projects.

**Prerequisites**

An existing extensibility action.

**Procedure**

1. Export an extensibility action.
   a. Navigate to Extensibility > Library > Actions.
   b. Select an extensibility action and click Export.
      
      The action script and its dependencies are saved on your local environment as a ZIP file.

2. Import an extensibility action.
   a. Navigate to Extensibility > Library > Actions.
   b. Click Import.
   c. Select the exported extensibility action and assign it to a project.
   d. Click Import.

**Note** If the imported extensibility action is already assigned to the specified project, you are prompted to select a conflict resolution policy.
What is an action flow

Action flows are a set of extensibility action scripts that are used to extend life cycles and automation further.

All action flows begin with `flow_start` and end with `flow_end`. You can link several extensibility action scripts together, by using the following action flow elements:

- **Sequential action flows** - Multiple extensibility action scripts running sequentially.
- **Fork action flows** - Multiple extensibility action scripts or flows that split pathways to contribute to the same output.
- **Join action flows** - Multiple extensibility action scripts or flows that join together and contribute to the same output.
- **Conditional action flows** - Multiple extensibility action scripts or flows that run after a condition is satisfied.

### Sequential action flows

Multiple extensibility action scripts running sequentially.

```json
version: "1"
flow:
  flow_start:
    next: action1
  action1:
    action: <action_name>
    next: action2
  action2:
    action: <action_name>
    next: flow_end
```

**Note** You can loop back to a previous action by assigning it as the `next:` action. For instance, in this example, instead of `next: flow_end`, you can enter `next: action1` to rerun action1 and restart the sequence of actions.

### Fork action flows

Multiple extensibility action scripts or flows that split pathways to contribute to the same output.
version: "1"
flow:
  flow_start:
    next: forkAction
forkAction:
  fork:
    next: [action1, action2]
  action1:
    action: <action_name>
    next: action3
  action3:
    action: <action_name>
    next: action4
  action4:
    action: <action_name>
    next: action7
  action7:
    action: <action_name>
  action2:
    action: <action_name>

**Note**  You can loop back to a previous action by assigning it as the next: action. For example, instead of next: flow_end to end your action flow, you can enter next: action1 to rerun action1 and restart the sequence of actions.

Join action flows

Multiple extensibility action scripts or flows that join pathways together and contribute to the same output.
Conditional action flows
Multiple extensibility action scripts or flows that run when a condition is satisfied using a switch element.

In some cases, the condition must be equal to `true` in order for the action to run. Other cases, as seen in this example, require parameter values to be met before an action can run. If none of the conditions are met the action flow fails.
version: 1
id: 1234
name: Test
inputs: ...
outputs: ...
flow:
  flow_start:
    next: forkAction
forkAction:
  fork:
    next: [action1, action2]
  action1:
    action: <action_name>
    next: action3
  action2:
    action: <action_name>
    next: action4
  action3:
    action: <action_name>
    next: action7
  action4:
    action: <action_name>
    next: action7
  action7:
    action: <action_name>
    next: joinElement
  action2:
    action: <action_name>
    next: switchAction
switchAction:
  switch:
    "${1 == 1}": action5
    "${1 != 1}": action6
  action5:
    action: <action_name>
    next: action8
  action6:
    action: <action_name>
    next: action8
  action8:
    action: <action_name>

**Note**  You can loop back to a previous action by assigning it as the next: action. For example, instead of next: flow_end to end your action flow, you can enter next: action1 to rerun action1 and restart the sequence of actions.

**How do I use an error handler with action flows**

You can configure your action flow to issue an error at specified stages of the flow by using an error handler element.

An error handler element requires two inputs:

- Specified error message of the failed action.
- Action flow inputs.
If an action in your flow fails and the action flow contains an error handler element, an error message is issued alerting you of the action failure. The error handler is an action on its own. The following script is an example of an error handler that can be used in an action flow.

```python
def handler(context, inputs):
    errorMsg = inputs["errorMsg"]
    flowInputs = inputs["flowInputs"]

    print("Flow execution failed with error {0}".format(errorMsg))
    print("Flow inputs were: {0}".format(flowInputs))

    outputs = {
        "errorMsg": errorMsg,
        "flowInputs": flowInputs
    }

    return outputs
```

You can view the successful and failed runs on the Action Runs window.

In this example, the flow-with-handler action flow, which contains an error handler element, was run successfully. However, one of the actions in the flow failed, which then initiated the error handler to issue an error.
How do I track action runs

The action runs tab shows you a log of subscription triggered extensibility actions and their status.

You can view the log of action runs using Extensibility > Activity > Action Runs. You can also filter the list of action runs by one or more properties at once.

Troubleshooting failed extensibility action runs

If your extensibility action run fails, you can perform troubleshooting steps to correct it.

When an action run fails you might receive an error message, a failed status, and a failed log. If your action run fails, it is either due to a deployment or code failure.

<table>
<thead>
<tr>
<th>Problem</th>
<th>Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deployment Failure</td>
<td>These failures are a result of problems related to the cloud account configuration, action deployment, or other dependencies that can prevent the action from deploying. Ensure that the project you used is defined within the configured cloud account and granted permissions to run functions. Before initiating the action again, you can test the action against a specific project within the action's details page.</td>
</tr>
<tr>
<td>Code Failure</td>
<td>These failures are a result of invalid scripts or code. Use the Action run logs to troubleshoot and correct the invalid scripts.</td>
</tr>
</tbody>
</table>

Extensibility workflow subscriptions

You can use your vRealize Orchestrator hosted workflows with Cloud Assembly to extend application lifecycle.

How do I modify virtual machine properties using a vRealize Orchestrator workflow subscription

You can use an existing vRealize Orchestrator workflow to modify virtual machine properties and add virtual machines to the active directory.

The event topic parameters define the format of the payload for Event Broker Service (EBS) messages. To receive and use EBS message payload inside a workflow, you must define the inputProperties workflow input parameters.

Prerequisites

- Cloud administrator user role
- Existing vRealize Orchestrator on-premises workflows.
- Successful integration and connection to the vRealize Orchestrator client server.
Procedure

1. Select **Extensibility > Subscriptions**.

2. Click **New Subscription**.

3. Create a subscription with the following parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>RenameVM</td>
</tr>
<tr>
<td>Event topic</td>
<td>Select an event topic suitable for the desired vRealize Orchestrator integration. For example, compute allocation.</td>
</tr>
<tr>
<td>Blocking/Non-blocking</td>
<td>Non-blocking</td>
</tr>
<tr>
<td>Action/workflow</td>
<td>Select a vRealize Orchestrator runnable type. Select the desired workflow. For example, Set VM name.</td>
</tr>
</tbody>
</table>

4. To save your subscription, click **Save**.

5. Assign and activate your subscription by creating a cloud template or deploying an existing cloud template.

What to do next

Verify that the workflow initiated successfully by one of the following methods:

- Verify the workflow runs log, **Extensibility > Activity > Workflow Runs**.
- Open the vRealize Orchestrator client and check workflow status by navigating to the workflow and verifying the status or by opening the specific logs tab.

How do I integrate Cloud Assembly for ITSM with ServiceNow using vRealize Orchestrator workflows

Using vRealize Orchestrator hosted workflows, you can integrate Cloud Assembly with ServiceNow for ITSM compliance.
Enterprise users commonly integrate their Cloud Management Platform with an IT Service Management (ITSM) and Configuration Management Database (CMDB) platform for compliance. Following this example, you can integrate Cloud Assembly with ServiceNow for CMDB and ITSM using vRealize Orchestrator hosted workflows. When using vRealize Orchestrator integrations and workflows, capability tags are especially useful if you have multiple instances for different environments. For more information on capability tags, See Using capability tags in Cloud Assembly.

**Note**  You can also integrate ServiceNow with Cloud Assembly using extensibility action scripts. For information about integrating ServiceNow using extensibility action scripts, see How do I integrate Cloud Assembly with ServiceNow using extensibility actions.

In this example, the ServiceNow integration is composed of three top-level workflows. Each workflow has their own subscriptions so that you can update and iterate each component individually.

- Event subscription entry point - Basic logging, identifies the requesting user and vCenter VM, if applicable.
- Integration workflow - Separates objects and feeds inputs into the technical workflow, handles logging, property, and output updates.
- Technical workflow - Downstream system integration for ServiceNow API to create the CMDB CI, CR, and Cloud Assembly IaaS API with additional virtual machine properties outside of the payload.

**Prerequisites**

- A standalone or clustered vRealize Orchestrator environment.
- A vRealize Orchestrator integration in Cloud Assembly. For information on integrating a standalone vRealize Orchestrator with Cloud Assembly, see Configure a vRealize Orchestrator integration in Cloud Assembly.

**Procedure**

1. Create and save a configuration file in vRealize Orchestrator that contains common configuration used in multiple workflows.

2. Save your Cloud Assembly API token in the same location, as the configuration file from Step 1.

   **Note**  The Cloud Assembly API token has an expiration.

3. Create a workflow in vRealize Orchestrator with the provided script element. This script references and locates a REST Host. It also standardizes REST actions that use an optional parameter of a token, which is added as an extra authorization header.

   ```javascript
   var configPath = "CS"
   var configName = "environmentConfig"
   ```
var attributeName = "CASRestHost"

//get REST Host from configuration element
var restHost =
    System.getModule("au.com.cs.example").getRestHostFromConfig(configPath, configName, attributeName)

var ConfigurationElement =
    System.getModule("au.com.cs.example").getConfigurationElementByName(configName, configPath);
System.debug("ConfigurationElement:" + ConfigurationElement);
var casToken = ConfigurationElement.getAttributeWithKey("CASToken")["value"]
if(!casToken){
    throw "no CAS Token";
}

//REST Template
var opName = "casLogin";
var opTemplate = "/iaas/login";
var opMethod = "POST";

// create the REST operation:
var opLogin =
    System.getModule("au.com.cs.example").createOp(restHost, opName, opMethod, opTemplate);

//cas API Token
var contentObject = {
    "refreshToken": casToken
};
postContent = JSON.stringify(contentObject);

var loginResponse =
    System.getModule("au.com.cs.example").executeOp(opLogin, null, postContent, null);
try{
    var tokenResponse = JSON.parse(loginResponse)["token"]
    System.debug("token: " + tokenResponse);
} catch (ex) {
    throw ex + " No valid token";
}

//REST Template Machine Details
var opName = "machineDetails";
var opTemplate = "/iaas/machines/" + resourceId;
var opMethod = "GET";

var bearer = "Bearer " + tokenResponse;

var opMachine =
    System.getModule("au.com.cs.example").createOp(restHost, opName, opMethod, opTemplate);

// (Rest Operation, Params, Content, Auth Token)
var vmResponse =
    System.getModule("au.com.cs.example").executeOp(opMachine, null, 
"", bearer);

try{
    var vm = JSON.parse(vmResponse);
} catch (ex) {
    throw ex + " failed to parse vm details"
This script sends the output `cpuCount` and `memoryMB` to the parent workflow and updates the existing `customProperties` properties. These values can be used in subsequent workflows when creating the CMDB.

Add the ServiceNow CMDB Create CI script element to your workflow. This element locates the ServiceNow REST Host using the configuration item, creates a REST operation for the `cmdb_ci_vmware_instance` table, creates a string of content object based on workflow inputs for post data, and outputs the returned `sys_id`.

```javascript
var configPath = "CS";
var configName = "environmentConfig";
var attributeName = "serviceNowRestHost";
var tableName = "cmdb_ci_vmware_instance";

//get REST Host from configuration element
var restHost = System.getModule("au.com.cs.example").getRestHostFromConfig(configPath, configName, attributeName);

//REST Template
var opName = "serviceNowCreateCI";
var opTemplate = "/api/now/table/" + tableName;
var opMethod = "POST";

// create the REST operation:
var opCI = System.getModule("au.com.cs.example").createOp(restHost,opName,opMethod,opTemplate);

//cmdb_ci_vmware_instance table content to post;
var contentObject = {};
contentObject["name"] = hostname;
contentObject["cpus"] = cpuTotalCount;
contentObject["memory"] = MemoryInMB;
contentObject["correlation_id"] = deploymentId;
contentObject["disks_size"] = diskProvisionGB;
contentObject["location"] = "Sydney";
contentObject["vcenter_uuid"] = vcUuid;
contentObject["state"] = "On";
contentObject["owned_by"] = owner;

postContent = JSON.stringify(contentObject);
System.log("JSON: " + postContent);

// (Rest Operation, Params, Content, Auth Token)
```
var ciResponse = 
    System.getModule("au.com.cs.example").executeOp(opCI,null,postContent,null) ;

try{
    var cmdbCI = JSON.parse(ciResponse);
} catch (ex) {
    throw ex + " failed to parse ServiceNow CMDB response";
}

serviceNowSysId = cmdbCI['result']['sys_id'];

5 Using the output from the child workflow, create a properties object using the existing customProperties and overwrite the serviceNowSysId property with the value from ServiceNow. This unique id is used in the CMDB to mark an instance as retired on destroy.

Results

Cloud Assembly is successfully integrated with ITSM ServiceNow. For more information on how you can use workflows to integrate ServiceNow in Cloud Assembly, see Extending Cloud Assembly with vRealize Orchestrator for ServiceNow Integration.

Learn more about workflow subscriptions

By using an vRealize Orchestrator integration with Cloud Assembly, you can extend the life cycles of applications with workflows.

You can add a vRealize Orchestrator integration to Cloud Assembly, so you can use workflows in your subscriptions. These workflows are maintained on the vRealize Orchestrator server. You can create, modify, and delete workflows by using the vRealize Orchestrator client.

Best practices for creating vRealize Orchestrator workflows

A workflow subscription is based on a specific event topic and the event parameters of that topic. To ensure that the subscriptions initiate the vRealize Orchestrator workflows, you must configure them with the correct input parameters so that they work with the event data.

Workflow Input Parameters

Your custom workflow can include all the parameters or a single parameter that consumes all the data in the payload.

To use a single parameter, configure one parameter with a type of Properties and name inputProperties.

Workflow Output Parameters

Your custom workflow can include output parameters that are relevant to subsequent events necessary for a reply event topic type.

If an event topic expects a reply, the workflow output parameters must match the parameters of the reply schema.
How do I track workflow runs

The **Workflow Runs** window displays the logs of the subscription triggered workflows and their status.

You can view the logs of your workflow runs by navigating to **Extensibility > Activity > Workflow Runs**.

Troubleshooting failed workflow subscriptions

If your workflow subscription fails, you can perform troubleshooting steps to correct it.

Failed workflow runs can cause your workflow subscription not to start or complete successfully. Workflow run failure can result from several common problems.

<table>
<thead>
<tr>
<th>Problem</th>
<th>Cause</th>
<th>Solution</th>
</tr>
</thead>
</table>
| Your vRealize Orchestrator workflow subscription did not start or complete successfully. | You configured a workflow subscription to run a custom workflow when the event message is received, but the workflow does not run or complete successfully. | 1 Verify that the workflow subscription is saved correctly.  
2 Verify that the workflow subscription conditions are configured correctly.  
3 Verify that vRealize Orchestrator contains the specified workflow.  
4 Verify that the workflow is configured correctly within vRealize Orchestrator. |
| Your approval request vRealize Orchestrator workflow subscription did not run. | You configured a pre-approval or post-approval workflow subscription to run a vRealize Orchestrator workflow. The workflow does not run when a machine that matches the defined criteria is requested in the service catalog. | To successfully run an approval workflow subscription, you must verify that all the components are configured correctly.  
1 Verify that the approval policy is active and correctly applied.  
2 Verify that your workflow subscription is correctly configured and saved.  
3 Review the event logs for messages related to approvals. |
| Your approval request vRealize Orchestrator workflow subscription was rejected. | You configured a pre-approval or post-approval workflow subscription that runs a specified vRealize Orchestrator workflow, but the request is rejected on the external approval level. One possible cause is an internal workflow run error in vRealize Orchestrator. For example, the workflow is missing or the vRealize Orchestrator server is not running. | 1 Review the logs for messages related to approvals.  
2 Verify that the vRealize Orchestrator server is running.  
3 Verify that vRealize Orchestrator contains the specified workflow. |
Learn more about extensibility subscriptions

You can extend your application life cycles by using either extensibility actions or vRealize Orchestrator hosted workflows with extensibility subscriptions.

When a triggering event occurs in your environment, the subscription is initiated and the specified workflow or extensibility action is run. You can view system events on the event log, workflow runs in the workflow runs window, and action runs in the action run window. Subscriptions are project-specific, meaning they are linked to cloud templates and deployments through the specified project.

Extensibility terminology

As you work with extensibility subscriptions within Cloud Assembly, you might encounter some terminology that is specific to the subscriptions and event broker service.

Table 6-6. Extensibility Terminology

<table>
<thead>
<tr>
<th>Term</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Event Topic</td>
<td>Describes a set of events that have the same logical intent and the same structure. Every event is an instance of an event topic. You can assign blocking parameters to certain event topics. For more information, see Blocking event topics.</td>
</tr>
<tr>
<td>Event</td>
<td>Indicates a change in the state in the producer or any of the entities managed by it. The event is the entity that records information about the event occurrence.</td>
</tr>
<tr>
<td>Event Broker Service</td>
<td>The service that dispatches messages published by a producer to the subscribed consumers.</td>
</tr>
<tr>
<td>Payload</td>
<td>The event data that contains all the relevant properties related to that Event Topic.</td>
</tr>
<tr>
<td>Subscription</td>
<td>Indicates that a subscriber is interested in being notified about an event by subscribing to an event topic and defining the criteria that triggers the notification. Subscriptions link either extensibility actions or workflows to triggering events used to automate parts of the applications life cycle.</td>
</tr>
<tr>
<td>Subscriber</td>
<td>The users notified by the events published to the event broker service based on the subscription definition. The subscriber can also be called a consumer.</td>
</tr>
<tr>
<td>System Administrator</td>
<td>A user with privileges to create, read, update, and delete tenant workflow subscriptions and system workflow subscriptions using Cloud Assembly.</td>
</tr>
<tr>
<td>Workflow Subscription</td>
<td>Specifies the event topic and conditions that trigger a vRealize Orchestrator workflow.</td>
</tr>
<tr>
<td>Action Subscription</td>
<td>Specifies the event topic and conditions that trigger an extensibility action to run.</td>
</tr>
</tbody>
</table>
Table 6-6. Extensibility Terminology (continued)

<table>
<thead>
<tr>
<th>Term</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Workflow</td>
<td>A vRealize Orchestrator workflow that is integrated within Cloud Assembly. You can link these workflows to events within subscriptions.</td>
</tr>
<tr>
<td>Extensibility Action</td>
<td>A streamlined script of code that can run after an event is triggered in a subscription. Extensibility actions are similar to workflows, but are more lightweight. Extensibility actions can be customized from within Cloud Assembly.</td>
</tr>
<tr>
<td>Action Runs</td>
<td>Accessible through the Action Runs tab. An action run is a detailed log of extensibility actions that have run in response to triggering events.</td>
</tr>
</tbody>
</table>

**Blocking event topics**

Some event topics support blocking events. The behavior of an extensibility subscription depends on whether the topic supports these event types and how you configure the subscription.

Cloud Assembly extensibility subscriptions can use two broad types of event topics: non-blocking and blocking event topics. The event topic type defines the behavior of the extensibility subscription.

**Non-Blocking Event Topics**

Non-blocking event topics only allow you to create non-blocking subscriptions. Non-blocking subscriptions are triggered asynchronously and you cannot rely on the order that the subscriptions are triggered in.

**Blocking Event Topics**

Some event topics support blocking. If a subscription is marked as blocking, all messages that meet the set conditions are not received by any other subscriptions with matching conditions until the runnable item of the blocking subscription is run.

Blocking subscriptions run in priority order. The highest priority value is 0 (zero). If you have more than one blocking subscription for the same event topic with the same priority level, the subscriptions run in a reverse alphabetical order based on the name of the subscription. After all blocking subscriptions are processed, the message is sent to all the non-blocking subscriptions at the same time. Because the blocking subscriptions run synchronously, the changed event payload includes the updated event when the subsequent subscriptions are notified.

You can use blocking event topics to manage multiple subscriptions that are dependent on each other.

For example, you can have two provisioning workflow subscriptions where the second subscription depends on the results of the first subscription. The first subscription changes a property during provisioning, and the second subscription records the new property, such as a machine name, in a file system. The ChangeProperty subscription is prioritized as 0 and the RecordProperty is prioritized as 1 because the second subscription uses the results of the first
subscription. When a machine is provisioned, the ChangeProperty subscription begins running. Because the RecordProperty subscription conditions are based on a post-provisioning condition, an event triggers the RecordProperty subscription. However, because the ChangeProperty workflow is a blocking workflow, the event is not received until it is finished. When the machine name is changed and the first workflow subscription is finished, the second workflow subscription runs and records the machine name in the file system.

**Recovery Runnable Item**

For blocking event topics, you can add a recovery runnable item to the subscription. The recovery runnable item in a subscription runs if the primary runnable item fails. For example, you can create a workflow subscription where the primary runnable item is a workflow that creates records in a CMDB system such as ServiceNow. Even if the workflow subscription fails, some records might be created in the CMDB system. In this scenario, a recovery runnable item can be used to clean up the records left in the CMDB system by the failed runnable item.

For use cases that include multiple subscriptions that are dependent on each other, you can add a `ebs.recover.continuation` property to the recovery runnable item. With this property, you can direct if the Extensibility service must continue with the next subscription in your chain, if the current subscription fails.

**Event topics provided with Cloud Assembly**

Cloud Assembly includes predefined event topics.

**Event Topics**

Event topics are the categories that group similar events together. When assigned to a subscription, event topics define which event triggers the subscription. The following event topics are provided by default with Cloud Assembly. All topics can be used to add or update custom properties or tags of the resource. If a vRealize Orchestrator workflow or extensibility action fails, the corresponding task fails as well.

**Table 6-7. Cloud Assembly Event Topics**

<table>
<thead>
<tr>
<th>Event Topic</th>
<th>Blockable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cloud template configuration</td>
<td>No</td>
<td>Issued when a cloud template configuration event, such as the creation or deletion of a cloud template, occurs. This event topic can be useful for notifying external systems of such events.</td>
</tr>
<tr>
<td>Cloud template version configuration</td>
<td>No</td>
<td>Issued when a new cloud template versioning event occurs, such as the creation, release, de-release, or restoration of a version. This event topic can be useful with integrations of third-party version control systems.</td>
</tr>
<tr>
<td>Event Topic</td>
<td>Blockable</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------</td>
<td>-----------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Compute allocation</td>
<td>Yes</td>
<td>Issued before the allocation of resourcenames and hostselections. Both of these properties can be modified at this stage. Issued once for a cluster of machines.</td>
</tr>
<tr>
<td>Compute gateway post provisioning</td>
<td>Yes</td>
<td>Issued after a compute gateway resource is provisioned.</td>
</tr>
<tr>
<td>Compute gateway post removal</td>
<td>Yes</td>
<td>Issued after a compute gateway is removed.</td>
</tr>
<tr>
<td>Compute gateway provisioning</td>
<td>Yes</td>
<td>Issued before a compute gateway is provisioned.</td>
</tr>
<tr>
<td>Compute gateway removal</td>
<td>Yes</td>
<td>Issued before a compute gateway is removed.</td>
</tr>
<tr>
<td>Compute initial power on</td>
<td>Yes</td>
<td>Issued after a resource is provisioned at the hypervisor layer, but before the resource is powered on for the first time. Currently, this event topic is only supported for vSphere. Events are sent for each machine in a cluster. Note: You can skip the initial power on for the resource.</td>
</tr>
<tr>
<td>Compute nat post provisioning</td>
<td>Yes</td>
<td>Issued after a compute NAT resource is provisioned.</td>
</tr>
<tr>
<td>Compute nat post removal</td>
<td>Yes</td>
<td>Issued after a compute NAT resource is removed.</td>
</tr>
<tr>
<td>Compute nat provisioning</td>
<td>Yes</td>
<td>Issued before a compute NAT is provisioned.</td>
</tr>
<tr>
<td>Compute nat removal</td>
<td>Yes</td>
<td>Issued before a compute NAT is removed.</td>
</tr>
<tr>
<td>Compute post provision</td>
<td>Yes</td>
<td>Issued after a resource is provisioned. Events are sent for each machine in a cluster.</td>
</tr>
<tr>
<td>Compute post removal</td>
<td>Yes</td>
<td>Issued after a compute resource is removed. Events are sent for each machine in a cluster.</td>
</tr>
<tr>
<td>Compute provision</td>
<td>Yes</td>
<td>Issued before the resource is provisioned at the hypervisor layer. Events are sent for each machine in a cluster.                                                                                             Note: You can change the allocated IP address.</td>
</tr>
<tr>
<td>Event Topic</td>
<td>Blockable</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------------</td>
<td>-----------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Compute removal</td>
<td>Yes</td>
<td>Issued before the resource is removed. Events are sent for each machine in a cluster.</td>
</tr>
<tr>
<td>Compute reservation</td>
<td>Yes</td>
<td>Issued at the time of reservation. Issued once for a cluster of machines.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Note</strong> You can change the placement order.</td>
</tr>
<tr>
<td>Custom resource post provision</td>
<td>Yes</td>
<td>Issued for post provisioning events triggered by custom resource operations.</td>
</tr>
<tr>
<td>Custom resource pre provision</td>
<td>Yes</td>
<td>Issued for pre provisioning events triggered by custom resource operations.</td>
</tr>
<tr>
<td>Deployment action completed</td>
<td>Yes</td>
<td>Issued after a deployment action is finished.</td>
</tr>
<tr>
<td>Deployment action requested</td>
<td>Yes</td>
<td>Issued before a deployment action is finished.</td>
</tr>
<tr>
<td>Deployment completed</td>
<td>Yes</td>
<td>Issued after the deployment of a cloud template or catalog request.</td>
</tr>
<tr>
<td>Deployment onboarded</td>
<td>No</td>
<td>Issued when a new deployment is onboarded.</td>
</tr>
<tr>
<td>Deployment requested</td>
<td>Yes</td>
<td>Issued before the deployment of a cloud template or catalog request.</td>
</tr>
<tr>
<td>Deployment resource action</td>
<td>Yes</td>
<td>Issued after the deployment of a resource action.</td>
</tr>
<tr>
<td>Deployment resource action</td>
<td>Yes</td>
<td>Issued before the deployment of a resource action.</td>
</tr>
<tr>
<td>Deployment resource completed</td>
<td>Yes</td>
<td>Issued after the provisioning of a deployment resource.</td>
</tr>
<tr>
<td>Deployment resource requested</td>
<td>Yes</td>
<td>Issued before the provisioning of a deployment resource.</td>
</tr>
<tr>
<td>Disk allocation</td>
<td>Yes</td>
<td>Issued for the preallocation of disk resources.</td>
</tr>
</tbody>
</table>
Table 6-7. Cloud Assembly Event Topics (continued)

<table>
<thead>
<tr>
<th>Event Topic</th>
<th>Blockable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Disk attach</td>
<td>Yes</td>
<td>Issued before a disk is attached to a machine. Disk attach is a read and write event. Disk properties supported for write-back are:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- diskFullPaths</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- diskDatastoreNames</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- diskParentDirs</td>
</tr>
<tr>
<td></td>
<td></td>
<td>All three vSphere specific disk properties are required for updates. All other properties are read-only.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Note</strong> Write-back is optional for vSphere First Class Disks.</td>
</tr>
<tr>
<td>Disk detach</td>
<td>Yes</td>
<td>Issued after a disk is detached from a machine. Disk detach is a read-only event.</td>
</tr>
<tr>
<td>Disk post removal</td>
<td>Yes</td>
<td>Issued after a disk resource is deleted.</td>
</tr>
<tr>
<td>Disk post resize</td>
<td>Yes</td>
<td>Issued after a disk resource is resized.</td>
</tr>
<tr>
<td>Kubernetes cluster allocation</td>
<td>Yes</td>
<td>Issued for the preallocation of resources for a Kubernetes cluster.</td>
</tr>
<tr>
<td>Kubernetes cluster post provision</td>
<td>Yes</td>
<td>Issued after a Kubernetes cluster is provisioned.</td>
</tr>
<tr>
<td>Kubernetes cluster post removal</td>
<td>Yes</td>
<td>Issued after a Kubernetes cluster is deleted.</td>
</tr>
<tr>
<td>Kubernetes cluster provision</td>
<td>Yes</td>
<td>Issued before a Kubernetes cluster is provisioned.</td>
</tr>
<tr>
<td>Kubernetes cluster removal</td>
<td>Yes</td>
<td>Issued before the process of deleting a Kubernetes cluster is initiated.</td>
</tr>
<tr>
<td>Kubernetes namespace allocation</td>
<td>Yes</td>
<td>Issued during the preallocation for Kubernetes namespace resources.</td>
</tr>
<tr>
<td>Kubernetes namespace post provision</td>
<td>Yes</td>
<td>Issued after a Kubernetes namespace resource is provisioned.</td>
</tr>
<tr>
<td>Kubernetes namespace post removal</td>
<td>Yes</td>
<td>Issued after a Kubernetes namespace resource is removed.</td>
</tr>
<tr>
<td>Kubernetes namespace provision</td>
<td>Yes</td>
<td>Issued before a Kubernetes namespace is provisioned.</td>
</tr>
<tr>
<td>Kubernetes namespace removal</td>
<td>Yes</td>
<td>Issued before a namespace cluster resource is removed.</td>
</tr>
<tr>
<td>Kubernetes supervisor namespace allocation</td>
<td>Yes</td>
<td>Issued during the preallocation for Kubernetes supervisor namespace resources.</td>
</tr>
<tr>
<td>Event Topic</td>
<td>Blockable</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------------------</td>
<td>-----------</td>
<td>------------------------------------------------------------------</td>
</tr>
<tr>
<td>Kubernetes supervisor namespace post provision</td>
<td>Yes</td>
<td>Issued after a supervisor namespace is provisioned.</td>
</tr>
<tr>
<td>Kubernetes supervisor namespace post removal</td>
<td>Yes</td>
<td>Issued after a supervisor namespace resource is removed.</td>
</tr>
<tr>
<td>Kubernetes supervisor namespace provision</td>
<td>Yes</td>
<td>Issued before a supervisor namespace is provisioned.</td>
</tr>
<tr>
<td>Kubernetes supervisor namespace removal</td>
<td>Yes</td>
<td>Issued before a supervisor namespace resource is removed.</td>
</tr>
<tr>
<td>Load balancer post provision</td>
<td>Yes</td>
<td>Issued after the provisioning of a load balancer.</td>
</tr>
<tr>
<td>Load balancer post removal</td>
<td>Yes</td>
<td>Issued after the removal of a load balancer.</td>
</tr>
<tr>
<td>Load balancer provision</td>
<td>Yes</td>
<td>Issued before provisioning a load balancer.</td>
</tr>
<tr>
<td>Load balancer removal</td>
<td>Yes</td>
<td>Issued before removing a load balancer.</td>
</tr>
<tr>
<td>Network Configure</td>
<td>Yes</td>
<td>Issued when the network is configured during compute allocation.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Note</strong> The Network Configure topic supports multiple IP addresses/NICs.</td>
</tr>
<tr>
<td>Network post provisioning</td>
<td>Yes</td>
<td>Issued after a network resource is provisioned.</td>
</tr>
<tr>
<td>Network post removal</td>
<td>Yes</td>
<td>Issued after a network resource is removed.</td>
</tr>
<tr>
<td>Network provisioning</td>
<td>Yes</td>
<td>Issued before a network resource is provisioned.</td>
</tr>
<tr>
<td>Network removal</td>
<td>Yes</td>
<td>Issued before a network resource is removed.</td>
</tr>
<tr>
<td>Project Lifecycle Event Topic</td>
<td>No</td>
<td>Issued when a project is created, updated, or deleted.</td>
</tr>
<tr>
<td>Provisioning request</td>
<td>Yes</td>
<td>Issued before a security group is removed.</td>
</tr>
<tr>
<td>Security group post provision</td>
<td>Yes</td>
<td>Issued after a security group is provisioned.</td>
</tr>
<tr>
<td>Security group post removal</td>
<td>Yes</td>
<td>Issued after a security group is removed.</td>
</tr>
<tr>
<td>Security group provisioning</td>
<td>Yes</td>
<td>Issued before a security group is provisioned.</td>
</tr>
<tr>
<td>Security group removal</td>
<td>Yes</td>
<td>Issued before a security group is removed.</td>
</tr>
</tbody>
</table>
**Event Parameters**

After you add an event topic, you can view the parameters of that event topic. These event parameters define the structure of the event's payload, or `inputProperties`. Certain event parameters cannot be modified and are marked as read-only. You can identify these read-only parameters by clicking the info icon to the right of the parameter.

**Extensibility event log**

The extensibility events page displays a list of all events that have occurred within your environment.

You can view the extensibility event logs by navigating to **Extensibility > Events**. You can also filter the list of events by one or more properties. To view additional details of an individual event, select the event's ID.

**Create an extensibility subscription**

By using a vRealize Orchestrator integration, or extensibility actions with Cloud Assembly, you can create subscriptions to extend your applications.

Extensibility subscriptions allow you to extend your applications by triggering workflows or actions at specific life-cycle events. You can also apply filters to your subscriptions to set Boolean conditions for the specified event. For example, the event and workflow or action only triggers if the Boolean expression is `'true'`. This is helpful for scenarios where you want to control when events, actions, or workflows are triggered.

**Prerequisites**

- Verify that you have the cloud administrator user role.
- If you are using the vRealize Orchestrator integration and workflows:
  - Existing vRealize Orchestrator on-premises workflows.
  - Successful integration and connection to the on-premises vRealize Orchestrator server. For more information, see **Configure a vRealize Orchestrator integration in Cloud Assembly**.
- If you are using extensibility actions:
  - Existing extensibility action scripts. For more information, see **How do I create extensibility actions**.
Procedure

1. Select **Extensibility > Subscriptions**.
2. Click **New Subscription**.
3. Enter the details of your subscription.
4. Set the **Organization scope** for the subscription.
   
   **Note**  For more information on creating extensibility subscriptions for organization providers and tenants, see Create extensibility subscriptions for providers or tenants.

5. Select an **Event Topic**.
6. (Optional) Set conditions for the event topic.
   
   **Note**  Conditions can be created by using a JavaScript syntax expression. This expression can include Boolean operators, such as "&&" (AND), "||" (OR), "^^" (XOR), and "!" (NOT). You can also use arithmetic operators, such as "==" (equal to), "!
   
   " (not equal to), ">=" (greater than or equal), "<=" (less than or equal), ">" (greater than), and "<" (less than). More complex Boolean expressions can be built out of simpler expressions. To access the event payload according to the specified topic parameters, use 'event.data' or any of the event header properties: `sourceType`, `sourceIdentity`, `timeStamp`, `eventType`, `eventTopicId`, `correlationType`, `correlationId`, `description`, `targetType`, `targetId`, `userName`, and `orgId`.
7. Under **Action/workflow**, select a runnable item for your extensibility subscription.
8. (Optional) If applicable, configure the blocking behavior for the event topic.
9. (Optional) To define the project scope of the extensibility subscription, deselect **Any Project** and click **Add Projects**.
   
   **Note**  If the organization scope of the subscription is set to **Any tenant organization**, the project scope is always set to **Any Project** and the project scope cannot be changed. You can only change the project scope if the organization scope is set to the provider organization.
10. To save your subscription, click **Save**.

**Results**

Your subscription is created. When an event, categorized by the selected event topic occurs, the linked vRealize Orchestrator workflow or extensibility action is initiated and all subscribers are notified.
What to do next

After creating your subscription, you can create or deploy a cloud template to link and use the subscription. You can also verify the status of the workflow or extensibility action run in the Extensibility tab in Cloud Assembly. For subscriptions containing vRealize Orchestrator workflows, you can also monitor runs and workflow status from the vRealize Orchestrator Client.

Troubleshooting an extensibility subscription

Troubleshoot extensibility subscription failures.

When your subscription fails, it is commonly a result of errors with your workflow or extensibility action script.

View topic parameters and payload

You can use a dump subscription topic parameters script to view the specific parameters and payload of your virtual machine at any given event stage.

Primarily, this script is useful for debugging and verifying available inputs for your vRealize Orchestrator workflow. To view all parameters of your virtual machine, use the following script with your workflow:

```javascript
function dumpProperties(props,lvl){
    var keys = props.keys;
    var prefix = ""
    for (var i=0; i<lvl; i++){
        prefix = prefix + "";
    }
    for (k in keys){
        var key = keys[k];
        var value = props.get(keys[k])
        if ("Properties" == System.getObjectType(value)){
            System.log(prefix + key + "[
            dumpProperties(value,(lvl+2));
            System.log(prefix+ "]")
        } else{
            System.log( prefix + key + ":" + value)
        }
    }
}
dumpProperties(inputProperties, 0)
customProps = inputProperties.get("customProperties")
```

Subscription version history

If your subscription fails, you can view the version history.
Viewing Subscription Version History

The **Version History** tab of the subscription editor can show you the change history of your subscription, including the user and date of the change. You can also compare different subscription versions by clicking **Compare to**. If your subscription fails or is running incorrectly, the version history can help identify the cause.
Managing Cloud Assembly deployments

As a Cloud Assembly cloud administrator or cloud template developer, you use the Deployment tab to manage your deployments and the associated resources. You can troubleshoot failed provisioning processes, make changes to resources or, and destroy unused deployments.

The deployments include deployed cloud templates and onboarded resources. It is also possible for resources that are created using the IaaS API to appear as deployments.

If you manage a small number of deployments, the deployment cards provide a graphical view for managing them. If you manage a large number of deployments, the deployment list and the resource list provide more robust management view.

Working with deployment cards and the deployment list

You can locate and manage your deployments using the card list. You can filter or search for specific deployments, and then run actions on those deployments.

Figure 7-1. Deployments page card view

1 Filter your requests based on attributes.
For example, you can filter based on owner, projects, lease expiration date, or other filtering options. Or you might want to find all the deployments for two projects with a particular tag. When you construct the filter for the projects and tag example, the results conform to the following criteria: (Project1 OR Project2) AND Tag1.

The values that you see in the filter pane depend on the current deployments that you have permission to view or manage.

Most of the filters and how to use them are relatively obvious. Additional information about some of these filters is provided below.

2 Search for deployments based on keywords or requester.
3 Sort the list to order by time or name.
4 Switch between the deployment card and the deployment list views.
5 Run deployment-level actions on the deployment, including deleting unused deployments to reclaim resources.

You can also see deployment costs, expiration dates, and status.

You can switch between the card and list view in the upper right of the page, to the right of the Sort text box. You can use the list view to manage a large number of deployments on fewer pages.

Figure 7-2. Deployment page list view

Working with selected deployment filters

The following table is a not a definitive list of filter options. Most of them are self-evident. However, some of the filters require a little extra knowledge.
Table 7-1. Selected filter information

<table>
<thead>
<tr>
<th>Filter name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Optimizable Resources Only</td>
<td>If you integrated vRealize Operations Cloud and are using the integration to identify reclaimable resources, you can toggle on the filter to limit the list of qualifying deployments.</td>
</tr>
<tr>
<td>Deployment Lifecycle Status</td>
<td>The Deployment Lifecycle Status and Last Request Status filters can be used individually or in combination, particularly if you manage a large number of deployments. Examples are included at the end of the Last Request Status section below. Deployment Lifecycle Status filters on the current state of the deployment based on the management operations. This filter is not available for deleted deployments. The values that you see in the filter pane depend on the current state of the listed deployments. You might not see all possible values. The following list includes all the possible values. Day 2 actions are included in the Update status.</td>
</tr>
<tr>
<td></td>
<td>- Create - Successful</td>
</tr>
<tr>
<td></td>
<td>- Create - In Progress</td>
</tr>
<tr>
<td></td>
<td>- Create - Failed</td>
</tr>
<tr>
<td></td>
<td>- Update - Successful</td>
</tr>
<tr>
<td></td>
<td>- Update - In Progress</td>
</tr>
<tr>
<td></td>
<td>- Update - Failed</td>
</tr>
<tr>
<td></td>
<td>- Delete - In Progress</td>
</tr>
<tr>
<td></td>
<td>- Delete - Failed</td>
</tr>
<tr>
<td>Last Request Status filters</td>
<td>Last Request Status filters on the last operation or action that ran on the deployment. This filter is not available for deleted deployments. The values that you see in the filter pane depend on the last operations that ran on the listed deployments. You might not see all possible values. The following list is all of the possible values.</td>
</tr>
<tr>
<td></td>
<td>- Pending. The first stage of a request where the action is submitted but the deployment process has not yet started.</td>
</tr>
<tr>
<td></td>
<td>- Failed. The request experienced a failure during any stage of the deployment process.</td>
</tr>
<tr>
<td></td>
<td>- Cancelled. The request was cancelled by a user while the deployment process was processing and not yet completed.</td>
</tr>
<tr>
<td></td>
<td>- Successful. The request successfully created, updated, or deleted a deployment.</td>
</tr>
<tr>
<td></td>
<td>- In Progress. The deployment process is currently running. Additional deployment states, for example,</td>
</tr>
</tbody>
</table>
Table 7-1. Selected filter information (continued)

<table>
<thead>
<tr>
<th>Filter name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initialization and Completion</td>
<td>Initialization and Completion that you see in the deployment History tab are not provided as filters, but you can use the In Progress filter to locate deployments in those states.</td>
</tr>
<tr>
<td>Approval Pending</td>
<td>Approval Pending. The request triggered one or more approval policies. The process is waiting for a response to the approval request.</td>
</tr>
<tr>
<td>Approval Rejected</td>
<td>Approval Rejected. The request was denied by the approvers in the triggered approval policies. The request does not continue.</td>
</tr>
</tbody>
</table>

The following examples illustrate how the how to use the Deployment Lifecycle Status and Last Request Status filters individually or together.

- To find all delete requests that failed, select **Delete - Failed** in the Deployment Lifecycle Status filter.
- To find all the requests waiting for approval, select **Approval Pending** in the Last Request Status filter.
- To find the delete requests where the approval request is still pending, select **Delete - In Progress** in the Deployment Lifecycle Status filter and **Approval Pending** in the Last Request Status filter.

Working with the resource list

You can use the resource list to manage the machines, storage volumes, and networks that make up your deployments. In the resource list you can manage them in resource type groups rather than by deployments.

Similar to the deployment list view, you can filter the list, select a resource type, search, sort, and run actions.

If you click the resource name, you can work with the resource in the context of the deployment details.

You can locate and manage your deployments using the card list. You can filter or search for specific deployments, and then run actions on those deployments.
This chapter includes the following topics:

- How do I monitor deployments in Cloud Assembly
- What can I do if a Cloud Assembly deployment fails
- How do I manage the life cycle of a completed Cloud Assembly deployment
- What actions can I run on Cloud Assembly deployments

How do I monitor deployments in Cloud Assembly

After you deploy a Cloud Assembly cloud template, you can monitor your request to ensure that the resources are provisioned and running. Beginning with the deployment card, you can verify the provisioning of your resources. Next, you can examine the deployment details. Finally, you can view and filter deleted deployments for up to 90 days after deletion.

Procedure

1. Click Deployments and locate your deployment card using the filter and search, if needed.

2. Review the card status.

   If the deployment is in progress, the process bar indicates the number of tasks remaining.
   If the deployment completed successfully, the card displays the basic details about the deployment.

   If an approval policy is triggered for your request, you might see the request in an in progress state with the name of at least one approver. Approval policies are defined in Service Broker by your administrator. The approvers are defined in the policy. The approvers approve requests in Service Broker. You might also encounter approvals on day 2 actions.
3. To determine where your resources were deployed, click the deployment name and review the details on the Topology page.

You will likely need the IP address for the primary component. As you click on each component, notice the information provided that is specific to the component. In this example, the IP address is highlighted.

The availability of the external link depends on the cloud provider. Where it is available, you must have the credential on that provider to access the component.

What to do next

- You can make changes to your deployment. See How do I manage the life cycle of a completed Cloud Assembly deployment.
- If your deployment fails, see What can I do if a Cloud Assembly deployment fails.

What can I do if a Cloud Assembly deployment fails

Your deployment request might fail for many reasons. It might be due to network traffic, a lack of resources on the target cloud provider, or a flawed deployment specification. Or, the deployment succeeded, but it does not appear to be working. You can use Cloud Assembly to examine your deployment, review any error messages, and determine whether the problem is the environment, the requested workload specification, or something else.

You use this workflow to begin your investigation. The process might reveal that the failure was due to a transient environmental problem. Redeploying the request after verifying the conditions have improved resolves this type of problem. In other cases, your investigation might require you to examine other areas in detail.

As a project member, you can review the request details in Cloud Assembly.

Procedure

1. To determine if a request failed, click the Deployments tab and locate the deployment card.
Failed deployments are indicated on the card.

a  Review the error message.

b  For more information, click the deployment name for the deployment details.

2  On the deployment details page, click the History tab.

   a  Review the event tree to see where the provisioning process failed. This tree is useful when you modify a deployment, but the change fails.

      The tree also shows when you run deployment actions. You can use the tree to troubleshoot failed changes.

   b  The Details provides a more verbose version of the error message.

   c  If the requested item was a Cloud Assembly cloud template, the link to the right of the message opens Cloud Assembly so that you can see the Request Details.

3  The Request Details provides the provisioning workflow for failed components so that you can research the problem.

   The request history is retained for 48 hours.

   View and filter deleted deployment history for up to 90 days after deletion.
a Review the error message.

b You can turn on the **Dev mode** to switch between the simple provisioning workflow and a more detailed flowchart.

c Click the card to review the deployment script.

4 Resolve the errors and redeploy the cloud template.

The errors might be in the template construction or they might be related to how your infrastructure is configured.

**What to do next**

When the errors are resolved and the cloud template is deployed, you can see information similar to the following example in the Request Details. To see the request details, select **Infrastructure > Activity > Requests.**
How do I manage the life cycle of a completed Cloud Assembly deployment

After a deployment is provisioned and running, you have several actions that you can run to manage the deployment. The life cycle management can include powering on or off, resizing, and deleting a deployment. You can also run various actions on individual components to manage them.
Procedure

1. Click **Deployments** and locate your deployment.

2. To access the deployment details, click the deployment name.

   You use the deployment details to understand how the resources are deployed and what changes have been made. You can also see pricing information, the current health of the deployment, and if you have any resources that need to be modified.
- **Topology** tab. You can use the Topology tab to understand the deployment structure and resources.
- **History** tab. The History tab includes all the provisioning events and any events related to actions that you run after requested item is deployed. If there are any problems with the provisioning process, the History tab events will help you with troubleshoot the failures.
- **Pricing** tab. You can use the pricing card to understand how much your deployment is costing your organization. Pricing information is based on vRealize Operations Cloud or CloudHealth integrations.
- **Monitor** tab. The Monitor tab data provides information about the health of your deployment based on data from vRealize Operations Cloud.

- **Alerts** tab. The Alerts tab provides active alerts on the deployment resources. You can dismiss the alert or add reference notes. The alerts are based on data from vRealize Operations Cloud.

- **Optimize** tab. The Optimize tab provides utilization information about your deployment and offers suggestions for reclaiming or otherwise modifying the resources to optimize resource consumption. The optimization information is based on data from vRealize Operations Cloud.

3. If you determine that a deployment is too costly in its current configuration and you want to resize a component, select the component on the topology page and then select **Actions > Resize** on the component page.

The available actions depend on the component, the cloud account, and your permissions.

4. As part of your development life cycle, one of your deployments is no longer needed. To remove the deployment and reclaim resources, select **Actions > Delete**.

The available actions depend on the state of the deployment.
To view your deleted deployments, click the filter on the Deployments tab, and then turn on Deleted Deployments Only toggle.

The list of deployments is now limited to those that are deleted. You might want to review the history of a particular deployment. For example, to retrieve the name of a deleted machine.

The deleted deployments are listed for 90 days.

What to do next

To learn more about possible actions, see What actions can I run on Cloud Assembly deployments.

What actions can I run on Cloud Assembly deployments

After you deploy cloud templates, you can run actions in Cloud Assembly to manage the resources. The available actions depend on the resource type and whether the actions are supported on a particular cloud account or integration platform.

The available actions also depend on what your administrator entitled you to run.

As an administrator or project administrator, you can set up Day 2 Actions policies in Service Broker. See How do I entitle consumers to Service Broker day 2 action policies

You might also see actions that are not included in the list. These are likely custom actions added by your administrator. For example, a How to create a Cloud Assembly resource action to vMotion a virtual machine.
## Table 7-2. List of possible actions

<table>
<thead>
<tr>
<th>Action</th>
<th>Applies to these resource types</th>
<th>Deployed resource type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add Disk</td>
<td>Machines</td>
<td>Amazon Web Service</td>
<td>Add additional disks to existing virtual machines. If you add a disk to an Azure machine, the persistent disk or non-persistent disk is deployed in the resource group that includes the machine. When you add a disk to an Azure machine, you can also encrypt the new disk using the Azure disk encryption set configured in the storage profile. When you add a disk to vSphere machines, you can select the SCSI controller, the order of which was set in the cloud template and deployed. You can also specify the unit number for the new disk. You cannot specify a unit number without a selected controller. If you do not select a controller or provide a unit number, the new disk is deployed to first available controller and assigned then next available unit number on that controller. If you add a disk to a vSphere machine for a project with defined storage limits, the added machine is not considered as part of the storage limits. Only resized disks are considered. If you use VMware Storage DRS (SDRS) and the datastore cluster is configured in the storage profile, you can add disks on SDRS to vSphere machines.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Google Cloud Platform</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Microsoft Azure</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>VMware vSphere</td>
<td></td>
</tr>
<tr>
<td>Apply Salt Configuration</td>
<td>Machines</td>
<td>VMware vSphere</td>
<td>Install a Salt minion or update an existing minion on a virtual machine. The Apply Salt Configuration option is available if you configured the SaltStack Config integration. To apply a configuration, you must select an authentication method. The <strong>Remote access with existing credentials</strong> uses the remote access credentials that are included in the deployment. If you changed the credentials on the machine after deployment, the action can fail. If you know the new credentials, use the Password authentication method. The <strong>Password</strong> and <strong>Private key</strong> use the user name and the password or key to validate your credentials and then connect to the virtual machine using SSH. If you do not provide a value for the Master ID and Minion ID, Salt creates the values for you.</td>
</tr>
</tbody>
</table>
### Table 7-2. List of possible actions (continued)

<table>
<thead>
<tr>
<th>Action</th>
<th>Applies to these resource types</th>
<th>Deployed resource type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cancel</td>
<td>Deployments</td>
<td>Amazon Web Service</td>
<td>Cancel a deployment or a day 2 action on a deployment or a resource while the request is being processed.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Google Cloud Platform</td>
<td>You can cancel the request on the deployment card or in the deployment details. After you cancel the request, it appears as a failed request on the Deployments tab. Use the Delete action to release any deployed resources and clean up your deployment list.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Microsoft Azure</td>
<td>Canceling a request that you think has been running too long is one method for managing deployment time. However, it is more efficient to set the Request Timeout in the projects. The default timeout is two hours. You can set it for a longer period of time if the workload deployment for a project requires more time.</td>
</tr>
<tr>
<td></td>
<td>Various resource types in deployments</td>
<td>VMware vSphere</td>
<td></td>
</tr>
<tr>
<td>Change Lease</td>
<td>Deployment s</td>
<td>Amazon Web Service</td>
<td>Change the lease expiration date and time.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Microsoft Azure</td>
<td>When a lease expires, the deployment is destroyed and the resources are reclaimed. Lease policies are set in Service Broker.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>VMware vSphere</td>
<td></td>
</tr>
<tr>
<td>Change Owner</td>
<td>Deployment s</td>
<td>Amazon Web Service</td>
<td>Changes to deployment owner to the selected user. The selected user must be a member of the same project that deployed the request.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Google Cloud Platform</td>
<td>If you want to assign a service administrator or project administrator as the owner, you must add them as a project member.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Microsoft Azure</td>
<td>When a cloud template designer deploys a template, the designer is both the requester and the owner. However, a requester can make another project member the owner.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>VMware vSphere</td>
<td>You can use policies to control what an owner can do with a deployment, giving them permissions that are more restrictive or less restrictive.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Action</td>
<td>Deployed resource types</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>-------------------------</td>
<td>-------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td></td>
</tr>
</tbody>
</table>
| Change Project Deployments | Amazon Web Service, Google Cloud Platform, Microsoft Azure, VMware vSphere | The change project action is only available for deployments with onboarded resources. The onboarded deployments can include only machines and disks. The action is not available for deployed cloud templates nor migrated deployments. If you make any changes to the deployment resources, for example, add a disk, you cannot run the change project action. Change the project of an onboarded deployment. This action allows you to change individual deployments from the onboarding project to a different project. Action constraints:  
  - The initiating user must have permission to run the change project action.  
  - If you are an administrator moving the deployment, you could move the deployment to a project where the owner is not a member and therefore loses access. You can add the user to the target project or move the deployment to a project where they are a member.  
  - The target project cloud zones must be the same as the source project cloud zones. If they are not, any future day 2 actions involving cloud account / region resources that you run might not work. |
| Change Security Groups Machines | VMware vSphere | You can associate and dissociate security groups with machine networks in a deployment. The change action applies to existing and on-demand security groups for NSX-V and NSX-T. This action is available only for single machines, not machine clusters. To associate a security group with the machine network, the security group must be present in the deployment. Dissociating a security group from all networks of all machines in a deployment does not remove the security group from the deployment. These changes do not affect security groups applied as part of the network profiles. This action changes the machine's security group configuration without recreating the machine. This is a non-destructive change.  
  - To change the machine's security group configuration, select the machine in the topology pane, then click the Action menu in the right pane and select Change Security Groups. You can now add or remove the association on the security groups with the machine networks. |
<table>
<thead>
<tr>
<th>Action</th>
<th>Applies to these resource types</th>
<th>Deployed resource type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connect to Remote Console</td>
<td>Machines</td>
<td>VMware vSphere</td>
<td>Open a remote session on the selected machine. Review the following requirements for a successful connection.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- As a cloud administrator, verify that the ESXi host is accessible over SSL from the network that your consumers are using.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- As a deployment consumer, verify that the provisioned machine is powered on. If your connection attempt indicates that a connection cannot be established with the host, you might be required to accept the self-signed certificate.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1. If you receive a certificate error, click accept the certificate and use the method provided by your browser to open the URL for the VMware ESXi page.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Opening the VMware ESXi page is sufficient to accept the certificate. You can close the ESXi page.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2. Refresh or reopen the remote connection page that previously displayed the warning.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The console for the target machine opens. As an administrator, if you prefer to ensure trusted certificates, see KB2113926 to configure CA signed certificates for ESXi hosts.</td>
</tr>
<tr>
<td>Create Disk Snapsho...</td>
<td>Machines and disks</td>
<td>Microsoft Azure</td>
<td>Create a snapshot of a virtual machine disk or a storage disk.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- For machines, you create snapshots for individual machine disks, including boot disk, image disks, and storage disks.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- For storage disks, you create snapshots of independent managed disks, not unmanaged disks.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>In addition to providing a snapshot name, you can also provide the following information for the snapshot:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Incremental Snapshot. Select the check box to create a snapshot of the changes since the last snapshot rather full snapshot.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Resource Group. Enter the name of the target resource group where you want to create the snapshot. By default, the snapshot is created in the same resource group that is used by the parent disk.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Encryption Set Id. Select the encryption key for the snapshot. By default, the snapshot is encrypted with the same key that is used by the parent disk.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Tags. Enter any tags that will help you manage the snapshots in Microsoft Azure.</td>
</tr>
</tbody>
</table>
### Table 7-2. List of possible actions (continued)

<table>
<thead>
<tr>
<th>Action</th>
<th>Applies to these resource types</th>
<th>Deployed resource type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create Snapshot</td>
<td>Machines</td>
<td>■ Google Cloud Platform</td>
<td>Create a snapshot of the virtual machine.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ VMware vSphere</td>
<td>If you are allowed only two snapshots in vSphere and you already have them, this command is not available until you delete a snapshot.</td>
</tr>
<tr>
<td>Delete Deployments</td>
<td>Deployment services</td>
<td>■ Amazon Web Service</td>
<td>Destroy a deployment. All the resources are deleted and the reclaimed.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ Google Cloud Platform</td>
<td>If a delete fails, you can run the delete action on a deployment a second time. During the second attempt, you can select Ignore Delete Failures. If you select this option, the deployment is deleted, but the resources might not be reclaimed. You should check the systems on which the deployment was provisioned to ensure that all resources are removed. If they are not, you must manually delete the residual resources on those systems.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ Microsoft Azure</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ VMware vSphere</td>
<td></td>
</tr>
<tr>
<td>NSX Gateway</td>
<td></td>
<td>■ NSX</td>
<td>Delete the NAT port forwarding rules from an NSX-T or NSX-V gateway.</td>
</tr>
<tr>
<td>Delete Machines and load balancers</td>
<td></td>
<td>■ Amazon Web Service</td>
<td>Delete a machine or load balancer from a deployment. This action might result in an unusable deployment.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ Microsoft Azure</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ VMware vSphere</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ VMware NSX</td>
<td></td>
</tr>
<tr>
<td>Security groups</td>
<td></td>
<td>■ NSX-T</td>
<td>If the security is not associated with any machine in the deployment, the process removes the security group from the deployment.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ NSX-V</td>
<td></td>
</tr>
<tr>
<td>Delete Disk Snapshot</td>
<td>Machines and disks</td>
<td>■ Microsoft Azure</td>
<td>Delete an Azure virtual machine disk or managed disk snapshot. This action is available when there is at least one snapshot.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ VMware vSphere</td>
<td>Delete a snapshot of the virtual machine.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ Google Cloud Platform</td>
<td></td>
</tr>
<tr>
<td>Disable Boot Diagnostics</td>
<td>Machines</td>
<td>■ Microsoft Azure</td>
<td>Turn off the Azure virtual machine debugging feature. The Disable option is only available if the feature is turned on.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ VMware vSphere</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ Google Cloud Platform</td>
<td></td>
</tr>
<tr>
<td>Edit Tags</td>
<td>Deployment services</td>
<td>■ Amazon Web Service</td>
<td>Add or modify resource tags that are applied to individual deployment resources.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ Microsoft Azure</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ VMware vSphere</td>
<td></td>
</tr>
</tbody>
</table>
### Table 7-2. List of possible actions (continued)

<table>
<thead>
<tr>
<th>Action</th>
<th>Applies to these resource types</th>
<th>Deployed resource type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enable Boot Diagnostics</td>
<td>Machines</td>
<td>Microsoft Azure</td>
<td>Turn on the Azure virtual machine debugging feature to diagnose virtual machine boot failures. The boot diagnostics information is available in your Azure console. The Enable option is only available if the feature is not currently turned on.</td>
</tr>
<tr>
<td>Get Terraform State</td>
<td>Terraform Configuration</td>
<td>Amazon Web Service</td>
<td>Display the Terraform state file. To view any changes that were made to the Terraform machines on the cloud platforms that they were deployed on and update the deployment, you first run the Refresh Terraform State action, and then run this Get Terraform State action. When the file is displayed in a dialog box. The file is available for approximately 1 hour before you need to run a new refresh action. You can copy it if you need it for later. You can also view the file on the deployment History tab. Select the Get Terraform State event on the Events tab, and then click Request Details. If the file is not expired, click View content. If the file is expired, run the Refresh and Get actions again.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Google Cloud Platform</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Microsoft Azure</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>VMware vSphere</td>
<td></td>
</tr>
<tr>
<td>Power Off</td>
<td>Deployment</td>
<td>Amazon Web Service</td>
<td>Power off the deployment without shutting down the guest operating systems.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Microsoft Azure</td>
<td>Power off the machine without shutting down the guest operating systems.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>VMware vSphere</td>
<td>Power on the deployment. If the resources were suspended, normal operation resumes from the point at which they were suspended.</td>
</tr>
<tr>
<td></td>
<td>Machines</td>
<td>Amazon Web Service</td>
<td>Power off the deployment without shutting down the guest operating systems.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Google Cloud Platform</td>
<td>Power off the machine without shutting down the guest operating systems.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Microsoft Azure</td>
<td>Power on the deployment. If the resources were suspended, normal operation resumes from the point at which they were suspended.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>VMware vSphere</td>
<td></td>
</tr>
</tbody>
</table>
### Table 7-2. List of possible actions (continued)

<table>
<thead>
<tr>
<th>Action</th>
<th>Applies to these resource types</th>
<th>Deployed resource type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Machines</td>
<td>- Amazon Web Service</td>
<td></td>
<td>Power on the machine. If the machine was suspended, normal operation resumes from the point at which the machine was suspended.</td>
</tr>
<tr>
<td></td>
<td>- Google Cloud Platform</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Microsoft Azure</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>- VMware vSphere</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reboot</td>
<td>Machines</td>
<td>- Amazon Web Service</td>
<td>Reboot the guest operating system on a virtual machine. For a vSphere machine, VMware Tools must be installed on the machine to use this action.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- VMware vSphere</td>
<td></td>
</tr>
<tr>
<td>Reconfigure</td>
<td>Load Balancers</td>
<td>- Amazon Web Service</td>
<td>Change the load balancer size and logging level. You can also add or remove routes, and change the protocol, port, health configuration, and member pool settings.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Microsoft Azure</td>
<td>For NSX load balancers, you can enable or disable the health check and modify the health options. For NSX-T, you can set the check to active or passive. NSX-V does not support passive health checks.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- VMware NSX</td>
<td></td>
</tr>
<tr>
<td>NSX Gateway</td>
<td>Gateway port forwarding</td>
<td>- NSX-T</td>
<td>Add, edit, or delete the NAT port forwarding rules from an NSX-T or NSX-V gateway.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- NSX-V</td>
<td></td>
</tr>
<tr>
<td>Security</td>
<td>Groups</td>
<td>- NSX-T</td>
<td>Add, edit, or remove firewall rules or constraints based on whether the security group is an on-demand or an existing security group.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- NSX-V</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>- VMware Cloud</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>- VMware vSphere</td>
<td></td>
</tr>
<tr>
<td>On-demand</td>
<td>security group</td>
<td>- NSX-T</td>
<td>Add, edit, or remove firewall rules for NSX-T and VMware Cloud on-demand security groups.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- NSX-V</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>- VMware vSphere</td>
<td></td>
</tr>
<tr>
<td>Existing</td>
<td>security group</td>
<td>- NSX-V</td>
<td>Add, edit, or remove constraints for existing NSX-V, NSX-T, and VMware Cloud security groups.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- VMware vSphere</td>
<td></td>
</tr>
<tr>
<td></td>
<td>To add or remove a rule, select the security group in the topology pane, click the <strong>Action</strong> menu in the right pane, and select <strong>Reconfigure</strong>. You can now add, edit, or remove the rules.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>To add or remove a constraint, select the security group in the topology pane, click the <strong>Action</strong> menu in the right pane, and select <strong>Reconfigure</strong>. You can now add, edit, or remove the constraints.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Action</td>
<td>Applies to these resource types</td>
<td>Deployed resource type</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------</td>
<td>---------------------------------</td>
<td>------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Refresh Terraform State</td>
<td>Terraform Configuration</td>
<td>■ Amazon Web Service</td>
<td>Retrieve the latest iteration of the Terraform state file. To retrieve any changes that were made to the Terraform machines on the cloud platforms that they were deployed on and update the deployment, you first run this Refresh Terraform State action. To view the file, run the Get Terraform State action on the configuration. Use the deployment history tab to monitor the refresh process.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ Google Cloud Platform</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ Microsoft Azure</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ VMware vSphere</td>
<td></td>
</tr>
<tr>
<td>Remove Disk</td>
<td>Machines</td>
<td>■ Amazon Web Service</td>
<td>Remove disks from existing virtual machines. If you run the day 2 action on a deployment that is deployed as vSphere machines and disks, the disk count is reclaimed as it applies to project storage limits. The project storage limits do not apply to additional disks that you added after deployment as a day 2 action.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ Google Cloud Platform</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ Microsoft Azure</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ VMware vSphere</td>
<td></td>
</tr>
<tr>
<td>Reset</td>
<td>Machines</td>
<td>■ Amazon Web Service</td>
<td>Force a virtual machine restart without shutting down the guest operating system.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ Google Cloud Platform</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ VMware vSphere</td>
<td></td>
</tr>
<tr>
<td>Resize</td>
<td>Machines</td>
<td>■ Amazon Web Service</td>
<td>Increase or decrease the CPU and memory of a virtual machine.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ Microsoft Azure</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ Google Cloud Platform</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ VMware vSphere</td>
<td></td>
</tr>
<tr>
<td>Resize Boot Disk</td>
<td>Machines</td>
<td>■ Amazon Web Service</td>
<td>Increase or decrease the size of your boot disk medium. If you run the day 2 action on a deployment that is deployed as vSphere machines and disks, and the action fails with a message similar to “The requested storage is more than the available storage placement,” it is likely due to the defined storage limits on your vSphere VM templates that are defined in the project. The project storage limits do not apply to additional disks that you added after deployment as a day 2 action.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ Google Cloud Platform</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ Microsoft Azure</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ VMware vSphere</td>
<td></td>
</tr>
<tr>
<td>Resize Disk</td>
<td>Storage disk</td>
<td>■ Amazon Web Service</td>
<td>Increase the capacity of a storage disk. If you run the day 2 action on a deployment that is deployed as vSphere machines and disks, and the action fails with a message similar to “The requested storage is more than the available storage placement,” it is likely due to the defined storage limits on your vSphere VM templates that are defined in the project. The project storage limits do not apply to additional disks that you added after deployment as a day 2 action.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ Google Cloud Platform</td>
<td></td>
</tr>
</tbody>
</table>
### Table 7-2. List of possible actions (continued)

<table>
<thead>
<tr>
<th>Action</th>
<th>Applies to these resource types</th>
<th>Deployed resource type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Machines</td>
<td></td>
<td>Amazon Web Service</td>
<td>Increase or decrease the size of disks included in the machine image template and any attached disks.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Google Cloud Platform</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Microsoft Azure</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>VMware vSphere</td>
<td></td>
</tr>
<tr>
<td>Restart</td>
<td>Machines</td>
<td>Microsoft Azure</td>
<td>Shut down and restart a running machine.</td>
</tr>
<tr>
<td>Revert to Snapshot</td>
<td>Machines</td>
<td>VMware vSphere</td>
<td>Revert to a previous snapshot of the machine. You must have an existing snapshot to use this action.</td>
</tr>
<tr>
<td>Run Puppet Task</td>
<td>Managed resources</td>
<td>Puppet Enterprise</td>
<td>Run the selected task on machines in your deployment. The tasks are defined in your Puppet instance. You must be able to identify the task and provide the input parameters.</td>
</tr>
<tr>
<td>Shutdown</td>
<td>Machines</td>
<td>VMware vSphere</td>
<td>Shut down the guest operating system and power off the machine. VMware Tools must be installed on the machine to use this action.</td>
</tr>
<tr>
<td>Suspend</td>
<td>Machines</td>
<td>Microsoft Azure</td>
<td>Pause the machine so that it cannot be used and does not consume any system resources other than the storage it is using.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>VMware vSphere</td>
<td></td>
</tr>
<tr>
<td>Update Deployments</td>
<td>Deployment</td>
<td>Amazon Web Service</td>
<td>Change the deployment based on the input parameters. For an example, see <a href="https://example.com">How to move a deployed machine to another network</a>.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Microsoft Azure</td>
<td>If the deployment is based on vSphere resources, and the machine and disks include the count option, storage limits defined in the project might apply when you increase the count. If the action fails with a message similar to “The requested storage is more than the available storage placement,” it is likely due to the defined storage limits on your vSphere VM templates that are defined in the project. The project storage limits do not apply to additional disks that you added after deployment as a day 2 action.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>VMware vSphere</td>
<td></td>
</tr>
</tbody>
</table>
Table 7-2. List of possible actions (continued)

<table>
<thead>
<tr>
<th>Action</th>
<th>Applies to these resource types</th>
<th>Deployed resource type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Update Tags</td>
<td>Machines and disks</td>
<td>Amazon Web Service</td>
<td>Add, modify, or delete a tag that is applied to an individual resource.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Microsoft Azure</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>VMware vSphere</td>
<td></td>
</tr>
<tr>
<td>Unregister</td>
<td>Machines</td>
<td>Amazon Web Service</td>
<td>The unregister action is only available for onboarded deployment machines.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Google Cloud Platform</td>
<td>Unregistered machines are removed from the deployment, along with any attached disks. By removing the resources, you can then re-run the onboarding workflow for the unregistered machine. You might want to onboard the resource again, this time to a new project. If you make any changes to the machine, for example, add a disk, before unregistering the machine, the unregister action fails.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Microsoft Azure</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>VMware vSphere</td>
<td></td>
</tr>
</tbody>
</table>