Using and Managing vRealize Automation Cloud Assembly

October 2022

vRealize Automation 8.0

vmware®

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

https://docs.vmware.com/

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

Contents

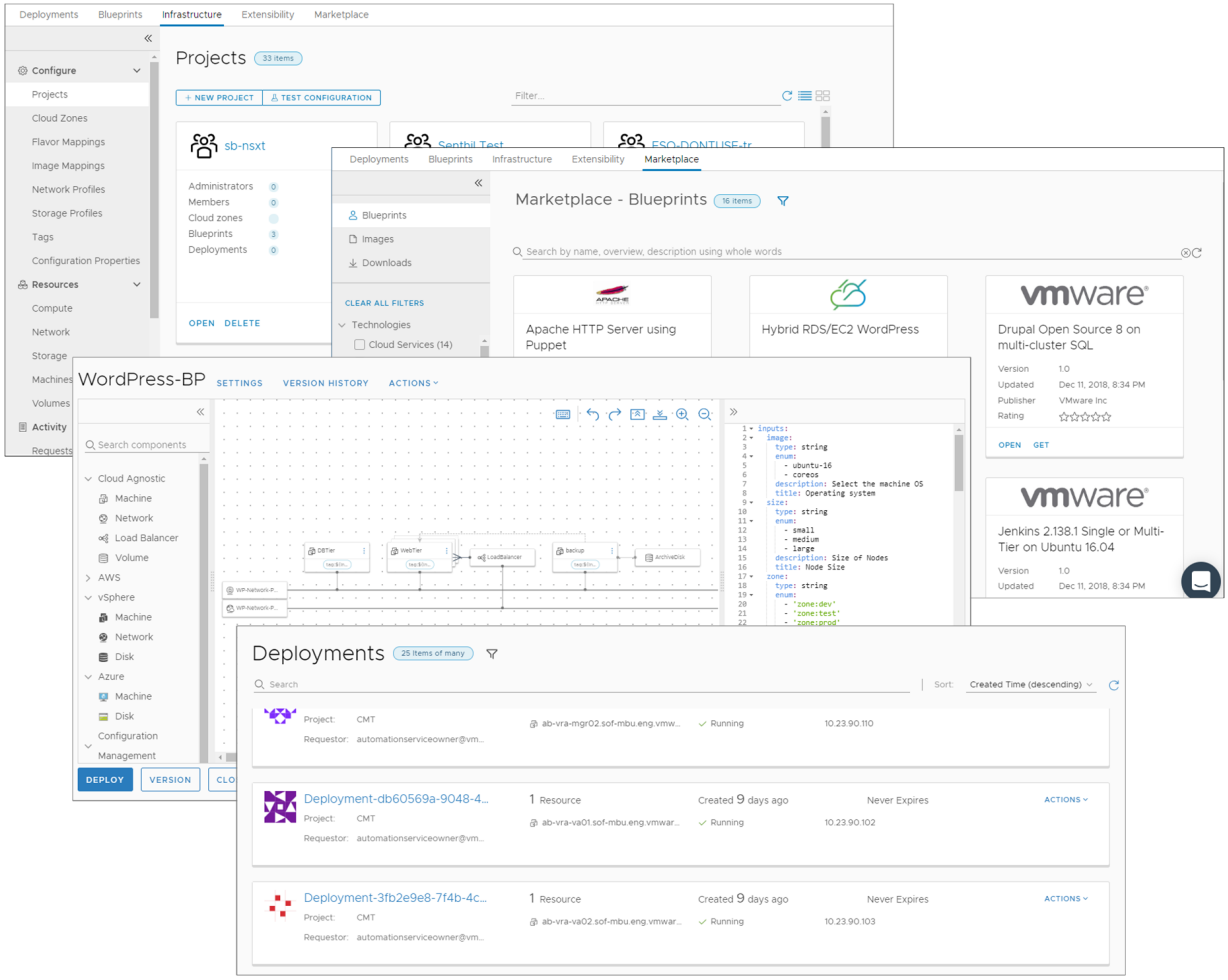
What is vRealize Automation Cloud Assembly

1

You use vRealize Automation Cloud Assembly to connect to your public and private cloud providers so that you can deploy machines, applications, and services that you create to your those resources. You and your teams develop blueprints-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 vRealize Automation 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 blueprints.
- The Marketplace tab provides VMware Solution Exchange blueprints and images that help you build your blueprint library and access supporting OVA or OVFs.
- The Blueprints tab is your development home. You use the canvas and the YAML editor to develop and then deploy your machines and applications.
- 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 vRealize Automation Cloud Assembly work

How does vRealize Automation Cloud Assembly work

vRealize Automation Cloud Assembly is a blueprint 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.

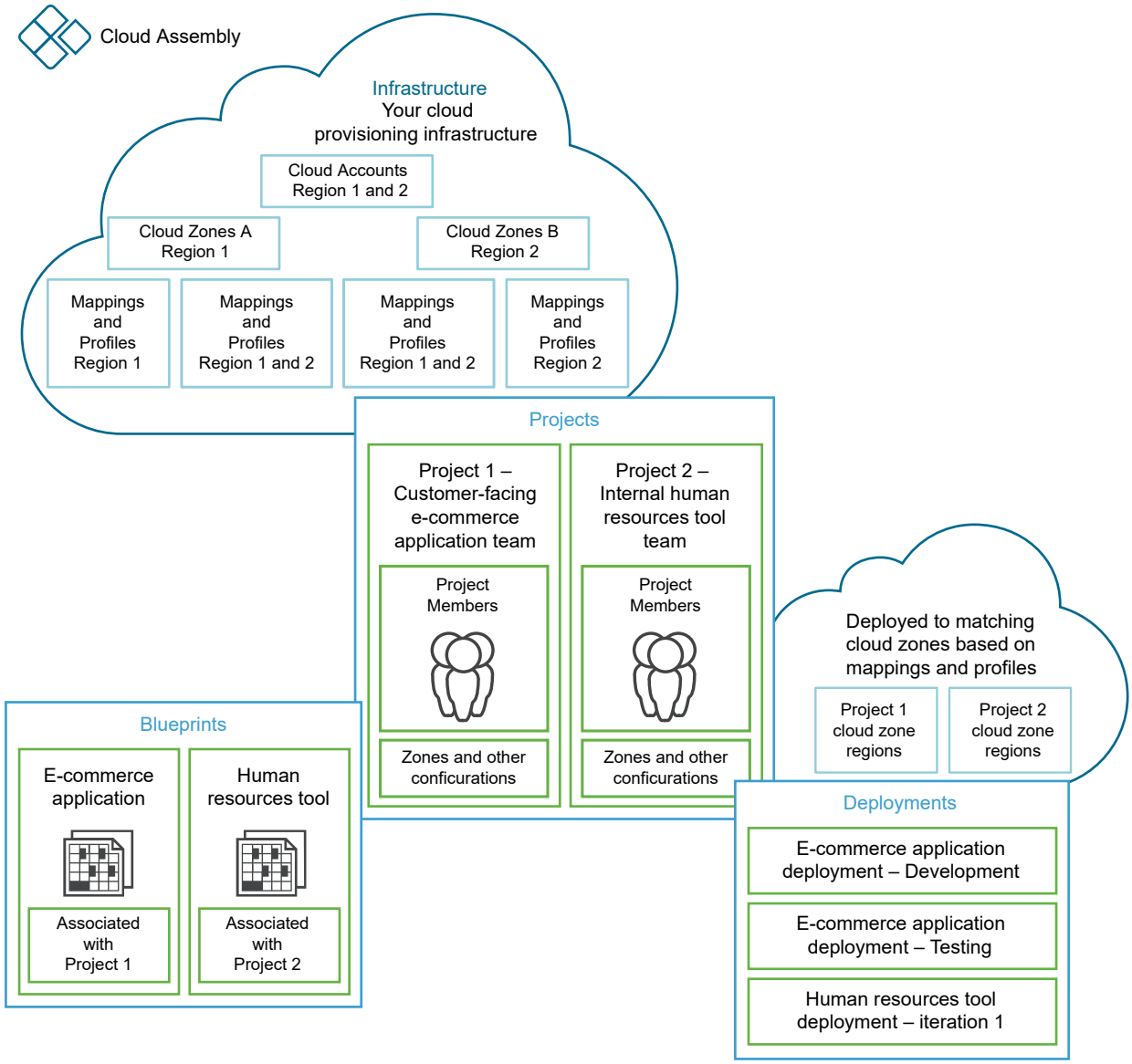
- Add your cloud vendor accounts. See Adding cloud accounts to vRealize Automation Cloud Assembly.
- Determine which regions or datastores are the cloud zones that you want your developers deploying to. See Learn more about vRealize Automation Cloud Assembly cloud zones.
- Create policies that define the cloud zones. See Chapter 4 Building your vRealize Automation Cloud Assembly resource infrastructure.

- Create projects that group the developers with the cloud zones. See Using vRealize Automation Cloud Assembly project tags and custom properties .

As a blueprint developer, you are a member of one or more projects. You create and deploy blueprints to the cloud zones associated with one of your projects.

- Develop blueprints for projects using the canvas. Your project administrator can use the marketplace to download blueprints and supporting images from the VMware Solution Exchange. See Chapter 6 Designing your vRealize Automation Cloud Assembly deployments and How to use the vRealize Automation Cloud Assembly Marketplace .
- Deploy your blueprints to project cloud zones based on policies and constraints.
- Manage your deployments, including deleting unused applications. See Chapter 7 Managing vRealize Automation Cloud Assembly deployments.

Welcome to vRealize Automation Cloud Assembly. If you want an example of how to define the infrastructure, and then create an deploy a blueprint, see The WordPress use case.


Cloud Assembly
Infrastructure
Your cloud provisioning infrastructure
Cloud Accounts Region 1 and 2
Cloud Zones A Region 1
Cloud Zones B Region 2
Mappings and Profiles Region 1
Mappings and Profiles Region 1 and 2
Mappings and Profiles Region 1 and 2
Mappings and Profiles Region 2
Projects
Project 1 – Customer-facing e-commerce application team
Project 2 – Internal human resources tool team
Project Members
Project Members
Zones and other conficurations
Zones and other conficurations
Blueprints
E-commerce application
Human resources tool
Associated with Project 1
Associated with Project 2
Deployed to matching cloud zones based on mappings and profiles
Project 1 cloud zone regions
Project 2 cloud zone regions
Deployments
E-commerce application deployment – Development
E-commerce application deployment – Testing
Human resources tool deployment – iteration 1

Setting up vRealize Automation Cloud Assembly for your organization

2

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 system.

This chapter includes the following topics:

What are the vRealize Automation Cloud Assembly user roles

User roles determine what you can see and do in vRealize Automation Cloud Assembly. Some roles are defined at the organization level, and some are specific to vRealize Automation Cloud Assembly.

User Roles

User roles are defined for the organization in the vRealize Automation 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.

For more information about the organization roles, see Administering vRealize Automation.

The vRealize Automation Cloud Assembly service roles, which are service-specific permissions, are also assigned at the organization level in the console.

Table 2-1. Service Roles

Role	Description
Cloud Assembly Administrator	Must have 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.
Cloud Assembly User	Any user who does not have the Cloud Assembly Administrator role. In a vRealize Automation Cloud Assembly project, the administrator adds users to projects as project members. The administrator can also add a project administrator. The permission for these two roles are defined below.

Project roles and permissions

Project roles, project administrator and project member, are defined in vRealize Automation Cloud Assembly and can vary between projects.

In the following tables, where the permissions are defined, remember that the cloud administrator has full permission on all areas of the UI.

Project administrators leverage the infrastructure that is created by the cloud administrator to ensure that their project members have the resources they need for their development work.

Table 2-2. Project Administrator Permissions

Tab	Node or Area	View	Create	Modify/Delete
Infrastructure	Configure - Projects	Yes (only your projects)	No	Yes (only your projects)
	Configure - Cloud Zones	No	No	No
	Configure - Flavor Mappings	Yes	No	No
	Configure - Image Mappings	Yes	No	No
	Configure - Network Profiles	Yes	No	No
	Configure - Storage Profiles	Yes	No	No
	Configure - Tags	Yes	No	No
	Resources - Compute	Yes	No	No
	Resources - Network	Yes	No	No
	Resources - Storage	Yes	No	No
	Resources - Machines	Yes (only your projects)	Yes	Yes (only your projects)

Table 2-2. Project Administrator Permissions (continued)

Tab	Node or Area	View	Create	Modify/Delete
	Resources - Volumes			
	Activity - Requests	Yes (only your projects)	N/A	Yes (only your projects)
	Activity - Events	Yes (only your projects)	N/A	Yes (only your projects)
	Connections - Cloud Accounts	No	No	No
	Connections - Integrations		No	No
	Connections - Cloud Proxies		No	No
	Cost - VMC Assessment	Yes	No	No
	Cost - Private Clouds	Yes	No	No
	Onboarding		No	No
Blueprints	Blueprints	Yes (only for your projects)	Yes (only for your projects)	Yes (only for your projects)
Deployments	Deployments	Yes (only for your projects)	N/A	Yes (only for your projects)

The project members are usually developers who create and deploy blueprints.

Table 2-3. Project Member Permissions

Tab	Node or Area	View	Create	Modify/Delete
Infrastructure	Configure - Projects	Yes (only the projects you are a member of)	No	No
	Configure - Cloud Zones	No	No	No
	Configure - Flavor Mappings	Yes	No	No
	Configure - Image Mappings	Yes	No	No
	Configure - Network Profiles	Yes	No	No
	Configure - Storage Profiles	Yes	No	No
	Configure - Tags	Yes	No	No
	Resources - Compute	Yes	No	No
	Resources - Network	Yes	No	No

Table 2-3. Project Member Permissions (continued)

Tab	Node or Area	View	Create	Modify/Delete
	Resources - Storage	Yes	No	No
	Resources - Machines	Yes (only the ones that you deployed)	Yes	Yes (only the ones that you deployed)
	Resources - Volumes			
	Activity - Requests	Yes (only the ones that you deployed)	N/A	Yes (only the ones that you deployed)
	Activity - Events	Yes (only the ones that you deployed)	N/A	Yes (only the ones that you deployed)
	Connections - Cloud Accounts	No	No	No
	Connections - Integrations			
	Connections - Cloud Proxies			
	Cost - VMC Assessment	Yes	No	No
	Cost - Private Clouds	Yes	No	No
	Onboarding			
Blueprints	Blueprints	Yes (only for your projects)	Yes (only for your projects)	Yes (only for your projects)
Deployments	Deployments	Yes (For just your deployments, unless the project deployments are share with all project members.)	N/A	Yes (For just your deployments, unless projects deployments are shared with all project members and you're entitled to run the day 2 actions.)

Adding cloud accounts to vRealize Automation Cloud Assembly

Cloud accounts are the configured permissions that vRealize Automation Cloud Assembly uses to collect data from the regions or data centers, and to deploy blueprints 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 vRealize Automation Cloud Assembly management by using an onboarding plan. See What are onboarding plans in vRealize Automation Cloud Assembly.

If you remove a cloud account that is used in a deployment, resources that are part of that deploymement become unmanaged.

Credentials required for working with cloud accounts in vRealize Automation

To configure and work with cloud accounts in vRealize Automation, verify that you have the following credentials.

Required cloud account credentials

To...	You need...
Sign up for and log in to vRealize Automation Cloud Assembly	A VMware ID. ■ Set up a My VMware account by using your corporate email address.
Connect to vRealize Automation 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. For related information about required ports and protocols, see: ■ Ports and Protocols in the *Installation* help ■ Port Requirements in the *Reference Architecture* help

To...	You need...
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 ■ ec2:CreateInternetGateway ■ ec2:CreateSecurityGroup ■ ec2:DescribeVolumes

To...	You need...

- ec2:CreateSnapshot
- ec2:ModifyInstanceAttribute
- ec2:DescribeRouteTables
- ec2:DescribeInstanceStatus
- ec2:DetachVolume
- ec2:RebootInstances
- ec2:AuthorizeSecurityGroupEgress
- ec2:ModifyVolume
- ec2:TerminateInstances
- ec2:DescribeSpotFleetRequestHistory
- ec2:DescribeTags
- ec2:CreateTags
- ec2:RunInstances
- ec2:DescribeNatGateways
- ec2:StopInstances
- ec2:DescribeSecurityGroups
- ec2:CreateVolume
- ec2:DescribeSpotFleetRequests
- ec2:DescribeImages
- ec2:DescribeVpcs
- ec2:DeleteSecurityGroup
- ec2:DeleteVpc
- ec2:CreateSubnet
- ec2:DescribeSubnets
- ec2:RequestSpotFleet

  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
  - elasticloadbalancing:ConfigureHealthCheck
  - elasticloadbalancing:AddTags
  - elasticloadbalancing:CreateTargetGroup
  - elasticloadbalancing:DeleteLoadBalancerListeners
  - elasticloadbalancing:DeregisterInstancesFromLoadBalancer
  - elasticloadbalancing:RegisterInstancesWithLoadBalancer
  - elasticloadbalancing:CreateLoadBalancerListeners

To...	You need...
	■ Load balancer resources: ■ * 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:ListGroupsForUser ■ iam:ListGroupPolicies ■ iam:GetGroupPolicy ■ iam:ListAttachedGroupPolicies ■ iam:ListPolicyVersions

To...	You need...
Add a Microsoft Azure cloud account	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: ■ Subscription ID Allows you to access to your Microsoft Azure subscriptions. ■ Tenant ID The authorization endpoint for the Active Directory applications you create in your Microsoft Azure account. ■ Client application ID Provides access to Microsoft Active Directory in your Microsoft Azure individual account. ■ Client application secret key The unique secret key generated to pair with your client application ID. The following permissions are needed for creating and validating Microsoft Azure cloud accounts: ■ Microsoft Compute ■ Microsoft.Compute/virtualMachines/extensions/write ■ Microsoft.Compute/virtualMachines/extensions/read ■ Microsoft.Compute/virtualMachines/extensions/delete ■ Microsoft.Compute/virtualMachines/deallocate/action ■ Microsoft.Compute/virtualMachines/delete ■ Microsoft.Compute/virtualMachines/powerOff/action ■ Microsoft.Compute/virtualMachines/read ■ Microsoft.Compute/virtualMachines/restart/action ■ Microsoft.Compute/virtualMachines/start/action ■ Microsoft.Compute/virtualMachines/write ■ Microsoft.Compute/availabilitySets/write ■ Microsoft.Compute/availabilitySets/read ■ Microsoft.Compute/availabilitySets/delete ■ Microsoft.Compute/disks/delete ■ Microsoft.Compute/disks/read ■ Microsoft.Compute/disks/write ■ Microsoft Network ■ Microsoft.Network/loadBalancers/backendAddressPools/join/action ■ Microsoft.Network/loadBalancers/delete ■ Microsoft.Network/loadBalancers/read ■ Microsoft.Network/loadBalancers/write ■ Microsoft.Network/networkInterfaces/join/action ■ Microsoft.Network/networkInterfaces/read ■ Microsoft.Network/networkInterfaces/write ■ Microsoft.Network/networkInterfaces/delete ■ Microsoft.Network/networkSecurityGroups/join/action ■ Microsoft.Network/networkSecurityGroups/read

To...	You need...
	■ Microsoft.Network/networkSecurityGroups/write ■ Microsoft.Network/networkSecurityGroups/delete ■ Microsoft.Network/publicIPAddresses/delete ■ Microsoft.Network/publicIPAddresses/join/action ■ Microsoft.Network/publicIPAddresses/read ■ Microsoft.Network/publicIPAddresses/write ■ Microsoft.Network/virtualNetworks/read ■ Microsoft.Network/virtualNetworks/subnets/delete ■ Microsoft.Network/virtualNetworks/subnets/join/action ■ Microsoft.Network/virtualNetworks/subnets/read ■ Microsoft.Network/virtualNetworks/subnets/write ■ Microsoft.Network/virtualNetworks/write ■ Microsoft Resources ■ Microsoft.Resources/subscriptions/resourcegroups/delete ■ Microsoft.Resources/subscriptions/resourcegroups/read ■ Microsoft.Resources/subscriptions/resourcegroups/write ■ Microsoft Storage ■ Microsoft.Storage/storageAccounts/delete ■ Microsoft.Storage/storageAccounts/listKeys/action ■ Microsoft.Storage/storageAccounts/read ■ Microsoft.Storage/storageAccounts/write ■ Microsoft Web ■ Microsoft.Web/sites/read ■ Microsoft.Web/sites/write ■ Microsoft.Web/sites/delete ■ Microsoft.Web/sites/config/read ■ Microsoft.Web/sites/config/write ■ Microsoft.Web/sites/config/list/action ■ Microsoft.Web/sites/publishxml/action ■ Microsoft.Web/serverfarms/write ■ Microsoft.Web/serverfarms/delete ■ Microsoft.Web/sites/hostruntime/functions/keys/read ■ Microsoft.Web/sites/hostruntime/host/read ■ Microsoft.web/sites/functions/masterkey/read 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/config/read ■ Microsoft.Web/sites/config/write ■ Microsoft.Web/sites/config/list/action ■ Microsoft.Web/sites/publishxml/action ■ Microsoft.Web/serverfarms/write ■ Microsoft.Web/serverfarms/delete

To...	You need...
	■ Microsoft.Web/sites/hostruntime/functions/keys/read ■ Microsoft.Web/sites/hostruntime/host/read ■ Microsoft.Web/sites/functions/masterkey/read If you are using Microsoft Azure with action-based extensibility with extensions, the following permissions are also needed: ■ Microsoft.Compute/virtualMachines/extensions/write ■ Microsoft.Compute/virtualMachines/extensions/read ■ Microsoft.Compute/virtualMachines/extensions/delete

To...	You need...
Add a Google Cloud Platform (GCP) cloud account	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, 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: ■ 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. 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

To...	You need...
	■ 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 ■ compute.subnetworks.get ■ compute.subnetworks.list ■ compute.subnetworks.use ■ compute.subnetworks.useExternalIp ■ compute.targetPools.get ■ compute.targetPools.list

To...	You need...

- compute.zoneOperations.get
- compute.zoneOperations.list
- compute.zones
- resourcemanager.projects.get
- resourcemanager.projects.list
- serviceusage.quotas.get
- serviceusage.services.get
- serviceusage.services.list

- roles/compute.instanceAdmin.v1

  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.

  - compute.acceleratorTypes
  - compute.addresses.get
  - compute.addresses.list
  - compute.addresses.use
  - compute.autoscalers
  - compute.backendBuckets.get
  - compute.backendBuckets.list
  - compute.backendServices.get
  - compute.backendServices.list
  - compute.diskTypes
  - compute.disks
  - compute.firewalls.get
  - compute.firewalls.list
  - compute.forwardingRules.get
  - compute.forwardingRules.list
  - compute.globalAddresses.get
  - compute.globalAddresses.list
  - compute.globalAddresses.use
  - compute.globalForwardingRules.get
  - compute.globalForwardingRules.list
  - compute.globalOperations.get
  - compute.globalOperations.list
  - compute.healthChecks.get
  - compute.healthChecks.list
  - compute.httpHealthChecks.get
  - compute.httpHealthChecks.list
  - compute.httpsHealthChecks.get
  - compute.httpsHealthChecks.list
  - compute.images
  - compute.instanceGroupManagers
  - compute.instanceGroups
  - compute.instanceTemplates

To...	You need...
	■ compute.instances ■ compute.interconnectAttachments.get ■ compute.interconnectAttachments.list ■ compute.interconnectLocations ■ compute.interconnects.get ■ compute.interconnects.list ■ compute.licenseCodes ■ compute.licenses ■ compute.machineTypes ■ compute.networkEndpointGroups ■ compute.networks.get ■ compute.networks.list ■ compute.networks.use ■ compute.networks.useExternalIp ■ compute.projects.get ■ compute.projects.setCommonInstanceMetadata ■ compute.regionBackendServices.get ■ compute.regionBackendServices.list ■ compute.regionOperations.get ■ compute.regionOperations.list ■ compute.regions ■ compute.reservations.get ■ compute.reservations.list ■ compute.resourcePolicies ■ compute.routers.get ■ compute.routers.list ■ compute.routes.get ■ compute.routes.list ■ compute.snapshots ■ compute.sslCertificates.get ■ compute.sslCertificates.list ■ compute.sslPolicies.get ■ compute.sslPolicies.list ■ compute.sslPolicies.listAvailableFeatures ■ compute.subnetworks.get ■ compute.subnetworks.list ■ compute.subnetworks.use ■ compute.subnetworks.useExternalIp ■ compute.targetHttpProxies.get ■ compute.targetHttpProxies.list ■ compute.targetHttpsProxies.get ■ compute.targetHttpsProxies.list ■ compute.targetInstances.get ■ compute.targetInstances.list ■ compute.targetPools.get

To...	You need...
	■ compute.targetPools.list ■ compute.targetSslProxies.get ■ compute.targetSslProxies.list ■ compute.targetTcpProxies.get ■ compute.targetTcpProxies.list ■ compute.targetVpnGateways.get ■ compute.targetVpnGateways.list ■ compute.urlMaps.get ■ compute.urlMaps.list ■ compute.vpnTunnels.get ■ compute.vpnTunnels.list ■ compute.zoneOperations.get ■ compute.zoneOperations.list ■ compute.zones ■ resourcemanager.projects.get ■ resourcemanager.projects.list ■ serviceusage.quotas.get ■ serviceusage.services.get ■ serviceusage.services.list
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 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 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 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 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 2-4. Permissions Required for vSphere Agent to Manage vCenter Server Instance

Attribute Value	Permission
Datastore	■ Allocate space ■ Browse datastore ■ Low level file operations
Datastore Cluster	Configure a datastore cluster
Folder	■ Create folder ■ Delete folder
Global	■ Manage custom attributes ■ Set custom attribute
Network	Assign network

Table 2-4. Permissions Required for vSphere Agent to Manage vCenter Server Instance (continued)

Attribute Value	Permission
Permissions	Modify permission
Resource	■ Assign VM to Res Pool ■ Migrate powered off virtual machine ■ Migrate powered on virtual machine
Content Library	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. ■ Add library item ■ Create local library ■ Create subscribed library ■ Delete library item ■ Delete local library ■ Delete subscribed library ■ Download files ■ Evict library item ■ Evict subscribed library ■ Probe subscription information ■ Read storage ■ Sync library item ■ Sync subscribed library ■ Type introspection ■ Update configuration settings ■ Update files ■ Update library ■ Update library item ■ Update local library ■ Update subscribed library ■ View configuration settings
Tags	■ Assign or unassign vSphere tag ■ Create a vSphere tag ■ Create a vSphere tag category ■ Delete vSphere tag ■ Delete vSphere tag category ■ Edit vSphere tag ■ Edit vSphere tag category ■ Modify UsedBy field for category ■ Modify UsedBy field for tag

Table 2-4. Permissions Required for vSphere Agent to Manage vCenter Server Instance (continued)

Attribute Value	Permission
vApp	■ Import ■ vApp application configuration 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.
Virtual Machine - Inventory	■ Create from existing ■ Create new ■ Move ■ Remove
Virtual Machine - Interaction	■ Configure CD media ■ Console interaction ■ Device connection ■ Power off ■ Power on ■ Reset ■ Suspend ■ Tools install
Virtual Machine - Configuration	■ Add existing disk ■ Add new disk ■ Remove disk ■ Advanced ■ Change CPU count ■ Change resource ■ Extend virtual disk ■ Disk change tracking ■ Memory ■ Modify device settings ■ Rename ■ Set annotation ■ Settings ■ Swapfile placement
Virtual Machine - Provisioning	■ Customize ■ Clone template ■ Clone virtual machine ■ Deploy template ■ Read customization specs
Virtual Machine - State	■ Create snapshot ■ Remove snapshot ■ Revert to snapshot

Configure Microsoft Azure for use with vRealize Automation Cloud Assembly

You must gather some information and perform some configuration in order to create a Microsoft Azure cloud account in vRealize Automation Cloud Assembly.

Procedure

1 Locate and record your Microsoft Azure subscritption 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. Altenatively, 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 resoruce 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 your 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.

Create a Microsoft Azure cloud account in vRealize Automation

As a cloud administrator, you can create a Microsoft Azure cloud account for account regions to which your team will deploy vRealize Automation blueprints.

To view an example use case of how Microsoft Azure cloud account works in vRealize Automation see The WordPress use case.

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.
- Verify that you have the required user role. See What are the vRealize Automation Cloud Assembly user roles.
- Configure a Microsoft Azure account for use with vRealize Automation. See Configure Microsoft Azure for use with vRealize Automation Cloud Assembly.
- If you do not have external Internet access, configure an Internet server proxy. See How do I configure an Internet proxy server for vRealize Automation.

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 vRealize Automation Cloud Assembly resources and deployments and Creating a Tagging Strategy.

7 Click **Save**.

Results

The account is added to vRealize Automation, and the selected regions are available for the specified cloud zone.

What to do next

Create infrastructure resources for this cloud account.

Create an Amazon Web Services cloud account in vRealize Automation

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 blueprints.

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.
- Verify that you have the required user role. See What are the vRealize Automation Cloud Assembly user roles.
- Verify that you have required AWS administrator credentials.
- If you do not have external Internet access, configure an Internet server proxy. See How do I configure an Internet proxy server for vRealize Automation.

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 vRealize Automation Cloud Assembly resources and deployments and Creating a Tagging Strategy.

7 Click **Add**.

Results

The account is added to vRealize Automation, 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

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 blueprints.

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.
- Verify that you have the required user role. See What are the vRealize Automation Cloud Assembly 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.
- If you do not have external Internet access, configure an Internet server proxy. See How do I configure an Internet proxy server for vRealize Automation.

Procedure

1 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. To successfully create the cloud account in vRealize Automation, the source GCP account must have the compute engine service enabled.

In vRealize Automation, 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 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 vRealize Automation Cloud Assembly resources and deployments and Creating a Tagging Strategy.

7 Click **Add**.

Results

The account is added to vRealize Automation, and the selected regions are available for the specified cloud zone.

What to do next

Create infrastructure resources for this cloud account.

Create a vCenter cloud account in vRealize Automation Cloud Assembly

You add a vCenter cloud account for the account regions that you want to deploy vRealize Automation Cloud Assembly blueprints to.

For network and security purposes, you can associate an NSX-T or NSX-V cloud account with the 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.
- Verify that you have the cloud administrator user role. See What are the vRealize Automation Cloud Assembly user roles.
- Verify that you have properly configured your ports and protocols to support the cloud account. See the *Ports and Protocols for vRealize Automation* topic in *Installing vRealize Automation with vRealize Easy Installer* and the *Port Requirements* topic in *vRealize Automation Reference Architecture Guide* in the vRealize Automation product documentation.

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 Enter your vCenter Server administrator credentials and click **Validate**.

   The data centers that are associated with the account are collected.

4 Select at least one of the available data centers on the specified vCenter Server to allow provisioning for this cloud account.

5 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 vRealize Automation Cloud Assembly cloud zones.

6 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 Cloud Assembly.

   For information about NSX-T cloud accounts, see Create an NSX-T cloud account in vRealize Automation Cloud Assembly.

7 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 vRealize Automation Cloud Assembly resources and deployments.

8 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 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 vRealize Automation Cloud Assembly resource infrastructure.

Create an NSX-V cloud account in vRealize Automation Cloud Assembly

For network and security purposes, you can create and associate an NSX-V cloud account with a 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.

- Verify that you have the cloud administrator user role. See What are the vRealize Automation Cloud Assembly user roles.
- 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 Assembly.
- Verify that you have properly configured your ports and protocols to support the cloud account. See the *Ports and Protocols for vRealize Automation* topic in *Installing vRealize Automation with vRealize Easy Installer* and the *Port Requirements* topic in *vRealize Automation Reference Architecture Guide* in the vRealize Automation product documentation.

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. 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, validation fails.
4. If available, select the vCenter endpoint that represents the vCenter cloud account that you are associating with this NSX-V account.
5. 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 vRealize Automation Cloud Assembly resources and deployments.
6. Click **Save**.

What to do next

You can create or edit a vCenter cloud account and associate it with this NSX cloud account. See Create a vCenter cloud account in vRealize Automation Cloud Assembly.

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 vRealize Automation Cloud Assembly cloud zones.

Configure infrastructure resources for this cloud account. See Chapter 4 Building your vRealize Automation Cloud Assembly resource infrastructure.

Create an NSX-T cloud account in vRealize Automation Cloud Assembly

For network and security purposes, you can create and associate an NSX-T cloud account with a vCenter cloud account.

To facilitate fault tolerance and high availability in deployments, each NSX-T data center endpoint represents a cluster of three NSX managers.

- vRealize Automation can point to one of the NSX managers. With this option, one NSX manager receives the API calls from vRealize Automation.

- vRealize Automation can point to the Virtual IP of the cluster. With this option, one NSX manager assumes control of the VIP. That manager receives the API calls from vRealize Automation. In case of failure, another node in the cluster assumes control of the VIP and receives the API calls from vRealize Automation.

  For more information about VIP configuration, 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 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.

  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 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.
- Verify that you have the cloud administrator user role. See What are the vRealize Automation Cloud Assembly user roles.
- 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 Assembly.
- Verify that you have properly configured your ports and protocols to support the cloud account. See the *Ports and Protocols for vRealize Automation* topic in *Installing vRealize Automation with vRealize Easy Installer* and the *Port Requirements* topic in *vRealize Automation Reference Architecture Guide* in the vRealize Automation product documentation.

Procedure

1. Select **Infrastructure > Connections > Cloud Accounts** and click **Add Cloud Account**.
2. Select the NSX-T account type and enter the host IP address for the NSX-T endpoint manager instance or VIP (see above).
3. 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, validation fails.
4. If available, select the vCenter endpoint that represents the vCenter cloud account that you are associating with this NSX-T cloud account.
5. 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 vRealize Automation Cloud Assembly resources and deployments.

6 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 Assembly.

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 vRealize Automation Cloud Assembly cloud zones.

Configure infrastructure resources for this cloud account. See Chapter 4 Building your vRealize Automation Cloud Assembly resource infrastructure.

Create a VMware Cloud on AWS cloud account in vRealize Automation

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 blueprints.

VMware Cloud on AWS requires some unique configuration procedures in vRealize Automation. To properly configure vRealize Automation 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 VMware Cloud on AWS use case 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.
- Verify that you have the cloud administrator user role. See What are the vRealize Automation Cloud Assembly user roles.
- If you do not have external Internet access, configure an Internet server proxy. See How do I configure an Internet proxy server for vRealize Automation.

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 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.

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 need 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 vRealize Automation Cloud Assembly resources and deployments.

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 for VMware Cloud on AWS, see VMware Cloud on AWS use case .

For related information about VMware Cloud on AWS outside of vRealize Automation, see VMware Cloud on AWS documentation.

Integrating vRealize Automation with other applications

Integrations enable you to add external systems to vRealize Automation.

Integrations include vRealize Orchestrator, configuration management and other external systems such as GitHub, Ansible, Puppet, and external IPAM providers such as Infoblox.

Note If you do not have external Internet access and your integration requires it, you can configure an Internet server proxy. See How do I configure an Internet proxy server for vRealize Automation.

How do I use GitLab and GitHub integration in vRealize Automation Cloud Assembly

vRealize Automation Cloud Assembly supports integration with GitLab and GitHub repositories so that you can manage blueprints and action scripts under source control. This functionality facilitates auditing and accountability of processes around deployment.

You must have an appropriate local Git repository configured with access for all designated users in order to set up Git integration with vRealize Automation Cloud Assembly. Also, you must save your blueprints 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 blueprints associated with selected projects become available to qualified users. You can use these blueprints 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 vRealize Automation Cloud Assembly. You can version action scripts either directly to Git, or you can create versions in vRealize Automation Cloud Assembly. If you create a version of an action in vRealize Automation Cloud Assembly, then it is automatically saved to Git as a version. Blueprints are a bit more complicated, because you cannot directly add them to a Git integration from vRealize Automation Cloud Assembly. You must save them directly to a Git instance, and then you can retrieve them from Git when working with the blueprint management page in vRealize Automation Cloud Assembly.

Before you Begin

You must create and save your blueprints in a specific structure in order for them to be detected by GitLab or GitHub.

- Configure and store Blueprints to be integrated with GitLab correctly. Only valid blueprint are imported into GitLab.
  - Create one or more designated folders for the blueprints.
  - All blueprints must be stored within `blueprint.yaml` files.
  - Ensure that the top of your blueprints 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 blueprints used with Git integration.

- Each blueprint must reside in a separate folder.
- All blueprints must be named `blueprint.yaml`.
- All blueprint YAML files must use `name` and `version` fields.
- Only valid blueprint are imported.

- If you update a draft blueprint 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 blueprint and also allow further sync's from Git, then you must create a new version after final changes.
- Configure GitLab blueprint integration in vRealize Automation Cloud Assembly

  This procedure demonstrates configuring GitLab integration in vRealize Automation Cloud Assembly so that you can work with blueprints in the repository and automatically download saved blueprints that are associated with designated projects. To use blueprints with GitLab, you must create a connection to an appropriate GitLab instance, and then save the desired blueprints to that instance.
- Configure GitHub integration in vRealize Automation Cloud Assembly

  You can integrate the GitHub cloud-based repository hosting service in vRealize Automation Cloud Assembly

Configure GitLab blueprint integration in vRealize Automation Cloud Assembly

This procedure demonstrates configuring GitLab integration in vRealize Automation Cloud Assembly so that you can work with blueprints in the repository and automatically download saved blueprints that are associated with designated projects. To use blueprints with GitLab, you must create a connection to an appropriate GitLab instance, and then save the desired blueprints to that instance.

When GitLab integration is configured with an existing repository, all blueprints associated with selected projects become available to qualified users. You can use these blueprints 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 blueprints to the Git repository from vRealize Automation Cloud Assembly. Also, you cannot push new blueprints to the repository from vRealize Automation Cloud Assembly. To add blueprints to a repository, developers must use the Git interface.

If you update a draft blueprint 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 blueprint and also allow further sync's from Git, then you must create a new version after final changes.

After you set up your blueprints for use with GitLab and collect required information, you must set up integration with your GitLab instance. Then, you can import the designated blueprints into GitLab. You can view a video demonstration of this procedure at 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 vRealize Automation Cloud Assembly. Also, you must create and save your blueprints in a specific structure in order for them to be detected by GitLab.

- Configure and store Blueprints to be integrated with GitLab correctly. Only valid blueprint are imported into GitLab. See How do I use GitLab and GitHub integration in vRealize Automation Cloud Assembly.

Procedure

1. Set up integration with your GitLab environment in vRealize Automation 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 vRealize Automation Cloud Assembly for more information.
   g. Click **Add**.
2. Configure the GitLab connection to accept blueprints in an appropriate repository.
   a. Select **InfrastructureIntegrations** 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 blueprints into the platform.

When the synchronization tasks are complete, a message indicates that the blueprints have been imported.

Results

You can now retrieve blueprints from GitLab.

Configure GitHub integration in vRealize Automation Cloud Assembly

You can integrate the GitHub cloud-based repository hosting service in vRealize Automation Cloud Assembly

You need a valid GitHub token to configure GitHub integration in vRealize Automation Cloud Assembly See the GitHub documentation for information about creating and locating your token.

Prerequisites

- You must have access to GitHub.
- Configure and store Blueprints to be integrated with GitHub correctly. Only valid blueprint are imported into GitHub. See How do I use GitLab and GitHub integration in vRealize Automation 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 vRealize Automation Cloud Assembly resources and deployments and Creating a Tagging Strategy.
6. Click **Add**.

Results

GitHub is available for use in vRealize Automation Cloud Assembly blueprints.

What to do next

You can now retrieve blueprints from GitHub.

Configure an external IPAM integration point in vRealize Automation

You can create a provider-specific external IPAM integration point to manage the IP addresses used in your blueprint 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.

You can create a provider-specific IPAM integration point to manage IP addresses and DNS settings for blueprint deployments and VMs in vRealize Automation.

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 point in vRealize Automation .

For information about how to create the needed assets to enable external IPAM partners and vendors to integrate their IPAM solution with vRealize Automation, see How do I use the IPAM SDK to create a provider-specific external IPAM integration package for vRealize Automation.

Prerequisites

- Verify that you have cloud administrator credentials. See Credentials required for working with cloud accounts in vRealize Automation.
- Verify that you have the cloud administrator user role. See What are the vRealize Automation Cloud Assembly 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 Marketplace and then deployed to vRealize Automation.
- Verify that you have access to a configured running environment for the IPAM provider.
- If you are using an actions-based extensibility (ABX) On-Prem Embedded running environment, verify that you have an HTTP proxy server in the vRealize Automation network that is able to pass outgoing traffic to external sites such as gcr.io and storage.googleapis.com. For details, see Pulling Docker images behind proxy in vRealize Automation 8.x (75180).

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 **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 and the IPAM provider.

   The IPAM framework only supports an actions-based extensibility (ABX) On-Prem Embedded running environment.

   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 do I upgrade to a newer IPAM integration package in vRealize Automation

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 existingvRealize Automation 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 point in vRealize Automation .

- Verify that you have cloud administrator credentials. See Credentials required for working with cloud accounts in vRealize Automation.
- Verify that you have the cloud administrator user role. See What are the vRealize Automation Cloud Assembly 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 Marketplace and then deployed to vRealize Automation.

  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 Assembly.
- Verify that you have access to a configured running environment for the IPAM provider. The running environment is typically an actions-based extensibility (ABX) On-Prem Embedded integration point.

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

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**.

Configure MyVMware Integration in vRealize Automation Cloud Assembly

You can integrate MyVMware with vRealize Automation Cloud Assembly to support VMware related actions and capabilities, such as accessing the VMware Marketplace for blueprints.

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 vRealize Automation Cloud Assembly resources and deployments and Creating a Tagging Strategy.
5. Click **Add**.

Results

My VMware is available for use with blueprints.

What to do next

Add a My VMware component to the desired blueprints.

Configure 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.

vRealize Automation includes a preconfigured vRealize Orchestrator instance that can be used for extensibility subscriptions. You can also access the client of the embedded vRealize Orchestrator from the vRealize Automation Cloud Services Console.

With the vRealize Orchestrator integration to vRealize Automation Cloud Assembly, you can add an external vRealize Orchestrator instance and use the included workflow library in extensibility subscriptions. For more information, see Extensibility workflow subscriptions.

Prerequisites

- Verify that you have cloud administrator credentials. For more information, see What are the vRealize Automation Cloud Assembly user roles.
- Migrate your vRealize Orchestrator 7.x instance to version 8.0. See, *Migrating Standalone vRealize Orchestrator to External vRealize Orchestrator 8.0* in *Installing, Configuring, and Migrating VMware vRealize Orchestrator*.

Procedure

1. Select **Infrastructure > Connections > Integrations**.

2 Click **Add integration**.

3 Select vRealize Orchestrator.

4 In vRealize Automation Cloud Assembly, enter the URL of the vRealize Orchestrator instance.

5 To validate the integration, click **Validate**.

6 Enter a name for the vRealize Orchestrator integration.

7 (Optional) Enter a description for the vRealize Orchestrator integration.

8 (Optional) Add capability tags. For more information on capability tags, see Using capability tags in vRealize Automation Cloud Assembly.

Note Capability tags can be used to manage multiple vRealize Orchestrator integrations. See Managing multiple vRealize Orchestrator integrations with project constraints.

9 Click **Add**.

The vRealize Orchestrator integration is saved.

What to do next

To verify that the integration is configured and that the workflows are added, select **Extensibility > Library > Workflows**.

Managing multiple vRealize Orchestrator integrations with project constraints

You can use project constraints to manage what vRealize Orchestrator integrations are used in workflow subscriptions.

vRealize Automation 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 blueprints provisioned by your project with soft or hard project constraints. For more information on project constraints, see Using vRealize Automation Cloud Assembly project tags and custom properties .

Prerequisites

- Verify that you have cloud administrator credentials. For more information, see What are the vRealize Automation Cloud Assembly user roles.
- Configure two or more vRealize Orchestrator integrations in vRealize Automation Cloud Assembly. For more information, see Configure vRealize Orchestrator integration in Cloud Assembly.
- Add capability tags to your vRealize Orchestrator integrations. For more information on capability tags, see Using capability tags in vRealize Automation Cloud Assembly.

Procedure

1 Navigate to **Infrastructure > Configure > 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 blueprint, vRealize Automation 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.

vRealize Automation 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. For more information, see What are the vRealize Automation Cloud Assembly user roles.
- Configure two or more vRealize Orchestrator integrations in vRealize Automation Cloud Assembly. For more information, see Configure vRealize Orchestrator integration in Cloud Assembly.
- Add capability tags to your vRealize Orchestrator integrations. For more information on capability tags, see Using capability tags in vRealize Automation 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 blueprint, vRealize Automation Cloud Assembly uses the tagging in the associated cloud account to manage what vRealize Orchestrator integrations are used in workflow subscriptions.

How do I work with Kubernetes in vRealize Automation Cloud Assembly

You can integrate Pivotal Container Service (PKS) or Red Hat OpenShift with vRealize Automation Cloud Assembly to manage and deploy Kubernetes resources. You can also integrate external Kubernetes resources in vRealize Automation Cloud Assembly.

After you create a PKS or OpenShift integration, applicable Kubernetes clusters become available in vRealize Automation Cloud Assembly and you can add and create Kubernetes components to vRealize Automation 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.

- Configure PKS Integration in vRealize Automation Cloud Assembly

  You can configure a PKS resource connection on premises and in the cloud to support Kubernetes integration and management capabilities in vRealize Automation Cloud Assembly.

- Working with Kubernetes clusters and namespaces in vRealize Automation Cloud Assembly

  You can add, view, and manage the configuration of Kubernetes clusters and namespaces which serve as the basis of Kubernetes deployments in vRealize Automation Cloud Assembly.

- Configure a Kubernetes Zone in vRealize Automation Cloud Assembly

  Kubernetes zones enable cloud administrators to define policy based placement of Kubernetes clusters and namespaces used in vRealize Automation Cloud Assembly deployments. An administrator can use this page to specify what clusters are available for provisioning of Kubernetes namespaces and, additionallly, what properties are acceptable for clusters.

- Adding Kubernetes components to blueprints in vRealize Automation Cloud Assembly

  When adding Kubernetes components to a vRealize Automation Cloud Assembly blueprint, 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 vRealize Automation Cloud Assembly Extensibility with Kubernetes

  vRealize Automation Cloud Assembly provides a standard set of event topics that correspond to typical actions related to Kubernetes cluster deployment. Users can subscribe to these topics as desired, and they receive notification when the event related to the subscribed topic occurs. You can also configure vRO workflows to run based on event notifications.

Configure PKS Integration in vRealize Automation Cloud Assembly

You can configure a PKS resource connection on premises and in the cloud to support Kubernetes integration and management capabilities in vRealize Automation Cloud Assembly.

PKS integrations enable you to manage PKS instances on premises and in the cloud and Kubernetes clusters provisioned on PKS 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 Pivotal Container Service (PKS) server set up with UAA authentication.
- Verify that you have cloud administrator credentials. For more information, see What are the vRealize Automation Cloud Assembly user roles.

Procedure

1. Select **Infrastructure > Connections > Integrations** and click **Add Integration**.
2. Select VMware Enterprise PKS.
3. Enter the IP address or FQDN, and PKS address for the PKS cloud account you are creating.
   - The IP address is the FQDN or IP address of the PKS user authentication server.
   - The PKS address is the FQDN or IP address for the main PKS server.
4. Select whether this PKS server is local or located in the public cloud or on a private cloud.
5. Enter an appropriate **Username** and **Password** for the PKS 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 vRealize Automation 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 blueprints to enable users to generate self-service deployments that use Kubernetes.

Working with Kubernetes clusters and namespaces in vRealize Automation Cloud Assembly

You can add, view, and manage the configuration of Kubernetes clusters and namespaces which serve as the basis of Kubernetes deployments in vRealize Automation Cloud Assembly.

You can view, add, and manage Kubernetes clusters and namespaces to which you are entitled access on the **Infrastructure > Resources > Kubernetes** page. 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 vRealize Automation 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.
- 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-baesed 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 Kubenetes Clusters page. When you deploy a namespace, it includes a link to a Kubconfig file that enables valid users, such as developers, to view and manage some aspects of that namespace.

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 Kubernetes Clusters in vRealize Automation Cloud Assembly

You can add new, existing, or external clusters to vRealize Automation 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 vRealize Automation 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.

Option	Description	Details
Deploy	Add new clusters to vRealize Automation Cloud Assembly	You must specify the PKS cloud account that to which this cluster will be deployed as well as the desired plan and the number of nodes.
Add Existing	Configure an existing cluster to work with your project.	You must specify the PKS 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 Kubernentes zones and namespaces appropriately.
Add External	Add a vanilla Kubernetes cluster, that might not be associated with PKS, to vRealize Automation Cloud Assembly.	You must designate a project to which the custer is associated, enter the IP address for the desired cluster and select a cloud proxy and certificate information required to connect to this cluster.

3 Click **Add** to make the cluster available within vRealize Automation Cloud Assembly.

Working with Kubernetes Namespaces in vRealize Automation 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 vRealize Automation 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 vRealize Automation Cloud Assembly.

Configure a Kubernetes Zone in vRealize Automation Cloud Assembly

Kubernetes zones enable cloud administrators to define policy based placement of Kubernetes clusters and namespaces used in vRealize Automation Cloud Assembly deployments. An administrator can use this page to specify what clusters are available for provisioning of Kubernetes namespaces and, additionallly, what properties are acceptable for clusters.

Cloud administrators can associate Kubernetes zones with PKS 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 PKS providers, these correspond to PKS 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.

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 Kubernaetes namespaces and clusters provisioned from the catalog.

Prerequisites

Configure integration with a suitable PKS deployment. See Configure PKS Integration in vRealize Automation Cloud Assembly

Procedure

1. Select **Infrastructure > Configure > Kubernetes Zone** and click **New Kubernetes Zone**.
2. Enter the PKS integration **Account** name to which you want this zone to apply.
3. Add a **Name** and **Description** for the Kubernetes Zone.
4. Add capability tags if appropriate. See Using capability tags in vRealize Automation Cloud Assembly for more information.
5. Click **Save**.
6. Click the On-demand tab and add PKS 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** button to add Kubernetes clusters to the zone. If you are working with an external cluster, it is automatically onboarded to vRealize Automation Cloud Assembly when you select it.

   You can add Kubenetes namespaces to the cluster on the Kubernetes Clusters page in vRealize Automation Cloud Assembly.

Results

Kubernetes zones are configured for use with vRealize Automation Cloud Assembly deployments.

What to do next

Assign the Kubernetes zone to a project.

1 Select **Infrastructure > Configure > 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 appliacable, and you also set the priority on the zones.

4 Click **Save.**

After you assign a zone to a project, you can use the Blueprints page to provision a deployment based on the Kubernetes zone and project configuration.

Adding Kubernetes components to blueprints in vRealize Automation Cloud Assembly

When adding Kubernetes components to a vRealize Automation Cloud Assembly blueprint, 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 blueprint in vRealize Automation Cloud Assembly, click Blueprints, select **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 blueprint is the most straightforward method of making Kubernetes resources available to valid users. You can use tags on clusters to control where they are deplyed just as you do with other Cloud Assembly resources. You can use tags to select a zone and a PKS plan during the allocation phase of cluster deployment.

Once you add a cluster in this way, it is automatically available to all valid users.

Blueprint Examples

The first blueprint example shows a blueprint 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 blueprint. If there were more than one zone configured with the tag, the one with the lowest priority number would be selected.

```
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 blueprint examples shows how to set up a blueprint 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 PKS plan durring the resource allocation phase of cluster deployment.

```
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 mange cluster usage, you can set up a variable in the blueprint 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 blueprint something like the following example:

```
1 formatVersion: 1
2 inputs:
3 name:
4    type: string
5    title: "Namespace name"
6 resources:
```

```
7    Cloud_KBS_Namespace_1:
8        type: Cloud.KBS.Namespace
9        properties:
10           name: ${input.name}
```

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**.

Using vRealize Automation Cloud Assembly Extensibility with Kubernetes

vRealize Automation Cloud Assembly provides a standard set of event topics that correspond to typical actions related to Kubernetes cluster deployment. Users can subscribe to these topics as desired, and they 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 > Libaray > Event Topics** page in vRealize Automation 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

Click one of the topics to view the schema for that topic which shows all the information that is collected and transmitted. 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 vRealize Automation Cloud Assembly

What Is configuration management in vRealize Automation Cloud Assembly

vRealize Automation Cloud Assembly supports integration with Puppet Enterprise and Ansible Open Source 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 vRealize Automation Cloud Assembly instance. Then you can make Puppet configuration management available to vRealize Automation Cloud Assembly by adding it to appropriate blueprints.

The vRealize Automation 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 appear. 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 vRealize Automation 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`

vRealize Automation Cloud Assembly manages the Ansible inventory file, so you must ensure that the vRealize Automation 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 vRealize Automation Cloud Assembly open-source integration, the users require a set of permissions to run the commands used by the vRealize Automation Cloud Assembly open-source provider. The following commands must be set in the user's sudoers file.

```
Defaults:myuser !requiretty
```

If the user is not part of an admin group that has no askpass application specified, set the following command in the user's sudoers file.

```
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.

Configure Puppet Enterprise integration in vRealize Automation Cloud Assembly

vRealize Automation 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.

Description	Log Location
Log for create and install related events	Logs are on the deployed machine at `~/var/tmp/vmware/provider/user_defined_script/$(ls -t ~/var/tmp/vmware/provider/user_defined_script/ | head -1)/`. Refer to the **log.txt** file for full logs. For detailed Puppet agent logs, refer to https://puppet.com/docs/puppet/4.8/services_agent_unix.html#logging
Log for Puppet delete and run related tasks	Logs are on the PE at `~/var/tmp/vmware/provider/user_defined_script/$(ls -t ~/var/tmp/vmware/provider/user_defined_script/ | head -1)/`. Refer to the **log.txt** file for full logs.

Procedure

1. Select **Infrastructure > Connections > Integrations** and click **Add Integration**.
2. Select Puppet.
3. Enter the required information on the Puppet configuration page.
4. Click **Validate** to check the integration.
5. Click **Add**.

Results

Puppet is available for use with blueprints.

What to do next

Add Puppet components to the desired blueprints.

1. Select Puppet under the Content Management heading on the blueprint menu and drag the Puppet component to the canvas.
2. Enter Puppet Properties on the pane to the right.

Property	Description
Master	Enter the name of the Puppet primary machine to be used with this blueprint.
Environment	Select the environment for the Puppet primary machine.
Role	Select the Puppet role to be used with this blueprint.
Agent Run Interval	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 blueprint.

3 Click the Code tab on the right pane to view the YAML code for the Puppet configuration properties.

Configure Ansible Open Source integration in vRealize Automation Cloud Assembly

vRealize Automation 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 vRealize Automation 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 blueprint 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 blueprint with the osType Windows property, the connection_type variable is automatically set to winm.

Ansible integration supports physical machines that do no 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 vRealize Automation Cloud Assembly for more details on configuring Ansible Open Source for integation in vRealize Automation Cloud Assembly.

Prerequisites

- The Ansible control machine must use Ansible version 2.6.0 or later.
- 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) NOPASSD: 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 blueprints 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 Click **Validate** to check the integration.

5 Click **Add**.

Results

Ansible is available for use with blueprints.

What to do next

Add Ansible components to the desired blueprints.

1 On the blueprint canvas page, select Ansible under the Configuration Management heading on the blueprint 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 blueprints. 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 blueprint, but the Ansible Control Machine will throw an when you run a playbook in the case of incorrect format or syntax.

The following blueprint YAML snippet shows an example useage 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}
```

How do I create an Active Directory integration in vRealize Automation Cloud Assembly

vRealize Automation 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 integration supports only an LDAP connection to the Active Directory server.

Prerequisites

- If you are configuring an Active Directory integration with vCenter on-premises, you must configure an extensibility cloud proxy for the Active Directory integration. 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 Active Directory server must use an LDAP server connection.
- You must have a project configured with appropriate cloud zones, and image and flavor mappings to use with the Active Directory integration.
- 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.
4. Enter the name and password for the LDAP server.
5. Enter the appropriate Base DN for the desired users and groups in your Active Directory.

   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. Click **Save**.

Results

You can now associate the project with Active Directory integration to a blueprint. When a machine is previsioned using this blueprint, it will be pre-staged in the specified Active Directory and Organizational Unit.

What are onboarding plans in vRealize Automation 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 vRealize Automation Cloud Assembly project.

When you add a cloud account that contains machines that were deployed outside of vRealize Automation Cloud Assembly, the machines are not managed by Cloud Assembly until you onboard them. Use an onboarding plan to bring unmanaged machines into vRealize Automation 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 blueprint and can also create one or many deployments.

You can onboard one or many unmanaged machines in a single plan. You can select machines manually or by using a filtering rule. Filtering rules select machines for onboarding based on criteria such as machine name, status, IP address, and tags.

- You can onboard up to 3,500 unmanaged machines within a single onboarding plan per hour.
- You can onboard up to 17,000 unmanaged machines concurrently within multiple onboarding plans per hour.

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 examples

For examples of onboarding techniques, see Example: Onboard selected machines as a single deployment in vRealize Automation Cloud Assembly and Example: Onboard rule-filtered machines as separate deployments in vRealize Automation 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.

Example: Onboard selected machines as a single deployment in vRealize Automation Cloud Assembly

In this example, you onboard two unmanaged machines as a single vRealize Automation Cloud Assembly deployment and create a single blueprint 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 vRealize Automation Cloud Assembly, you can use an onboarding plan to allow vRealize Automation Cloud Assembly to manage the machine deployments.

Prerequisites

- Verify that you have the required user role. See What are the vRealize Automation Cloud Assembly user roles.
- Review What are onboarding plans in vRealize Automation Cloud Assembly.
- Create and prepare a vRealize Automation Cloud Assembly project.

  This procedure involves some of the steps from the basic Wordpress use case. See The WordPress use case.

  - Create a project, add users, and assign user roles in the project. See WordPress use case: create a project.
  - Create an Amazon Web Services cloud account for the project. See WordPress use case: add cloud accounts .

    The Amazon Web Services cloud account in this procedure contains machines that were deployed before the cloud account was added to vRealize Automation Cloud Assembly and by an application other than vRealize Automation Cloud Assembly.
- Verify that the **Machines** page contains machines to onboard. See Machine resources.

Procedure

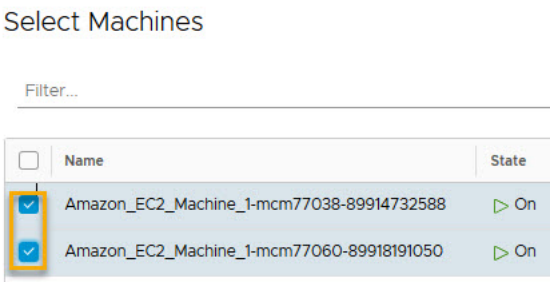
1 Go to **Infrastructure > Onboarding**.

2 Click **New Onboarding Plan** and enter sample values.

Setting	Sample Value
Plan name	VC-sqa-deployments
Description	Sample onboarding plan for AWS machine for OurCo-AWS cloud account
Cloud account	OurCo-AWS
Default project	WordPress

3 Click **Create**.

4 On the plan's **Deployments** tab, click **Select Machines**, choose one or more machines, and click **OK**.

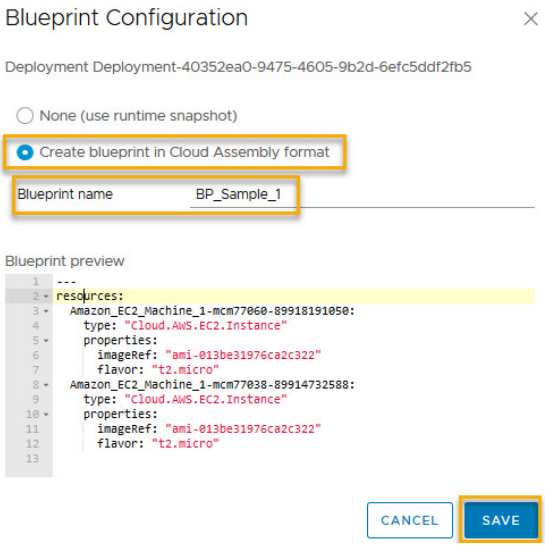


5 Select **Create one deployment that contains all the machines** and click **Create**.

6 Click the check box next to the new deployment name and click **Blueprint...**.

7 Click **Create blueprint in Cloud Assembly format**.

8 Enter a blueprint name and click **Save**.



Note When your onboarding plan uses a vSphere machine, you must edit the blueprint after the onboarding process is complete. The onboarding process cannot link the source vSphere machine and its machine template, and the resultant blueprint will contain the `imageRef: "no image available"` entry in the blueprint code. The blueprint cannot be deployed until you specify the correct template name in the `imageRef:` field. To make it easier to locate and update the blueprint after the onboarding process is complete, use the **Blueprint name** option on the deployment's **Blueprint Configuration** page. Record the auto-generated blueprint name or enter and record a blueprint name of your choice. When onboarding is complete, locate and open the blueprint 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 blueprint.

10 Open and examine the blueprint by clicking the **Blueprints** tab and then clicking the blueprint name.

11 Open and examine the deployment by clicking the **Deployments** tab and then clicking the deployment name.

Example: Onboard rule-filtered machines as separate deployments in vRealize Automation Cloud Assembly

In this example, you use a filtering rule to onboard machines whose state is On and whose name begins with the letters BG. You also create a separate vRealize Automation Cloud Assembly blueprint and deployment for each machine 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 vRealize Automation Cloud Assembly, you can use an onboarding plan to allow vRealize Automation Cloud Assembly to manage the machine deployments.

Prerequisites

- Verify that you have the required user role. See What are the vRealize Automation Cloud Assembly user roles.
- Review What are onboarding plans in vRealize Automation Cloud Assembly.
- Create and prepare a vRealize Automation Cloud Assembly project and populate it with one or more cloud accounts.

  This involves some of the basic steps in your guided setup procedure.

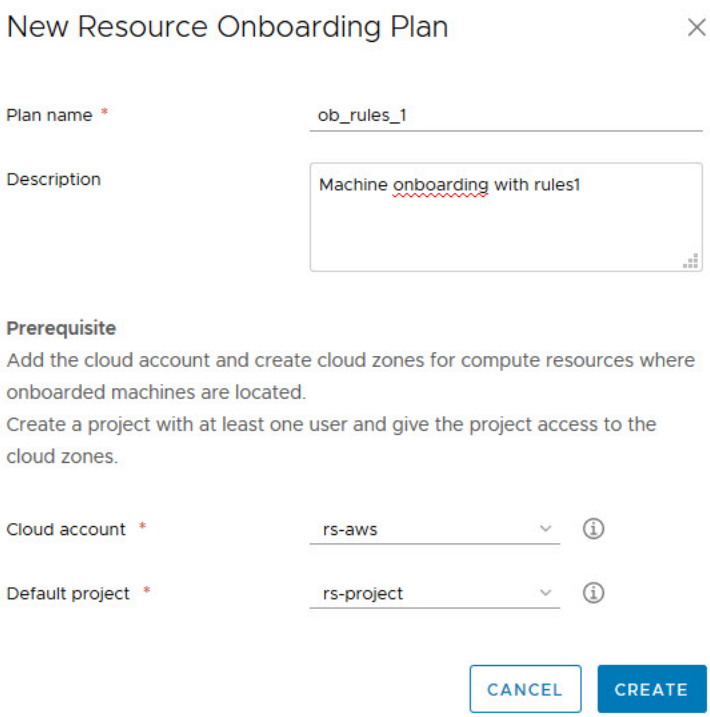
  - Create a project, add users, and assign user roles in the project. See WordPress use case: create a project.
  - Create one or more cloud accounts in designated regions for the project. See WordPress use case: add cloud accounts .
- Verify that the **Machines** page contains machines to onboard. See Machine resources.

Procedure

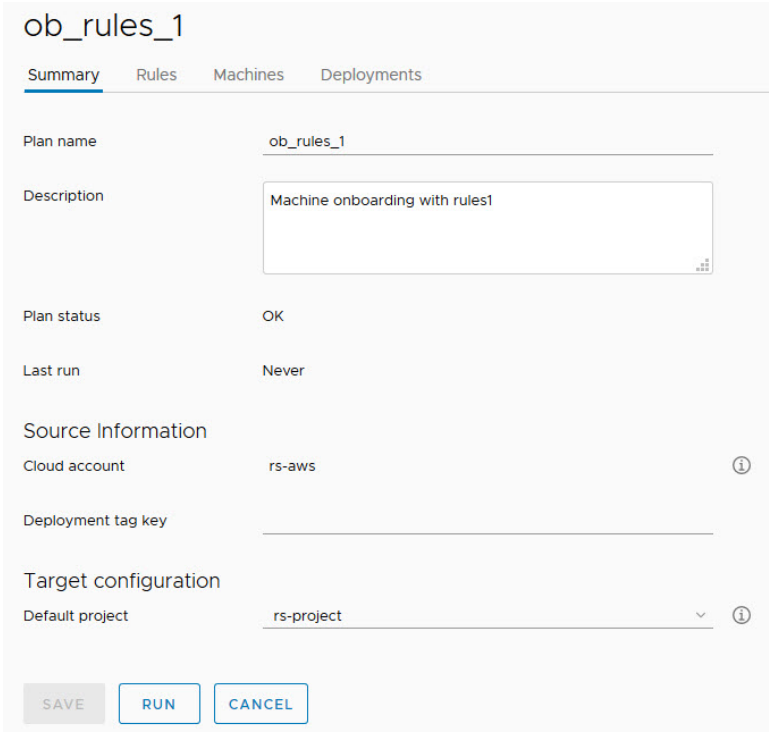
1 Go to **Infrastructure > Onboarding**.

2 Click **New Onboarding Plan** and enter values.

Setting	Sample Value
Plan name	ob_rules_1
Description	Machine onboarding with rules1
Cloud account	rs-aws
Default project	rs-project

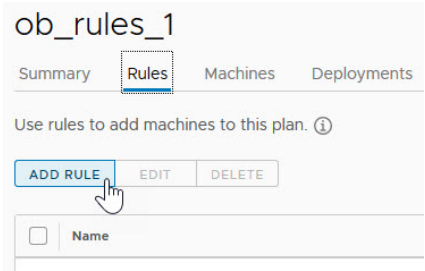


3 Click **Create**.

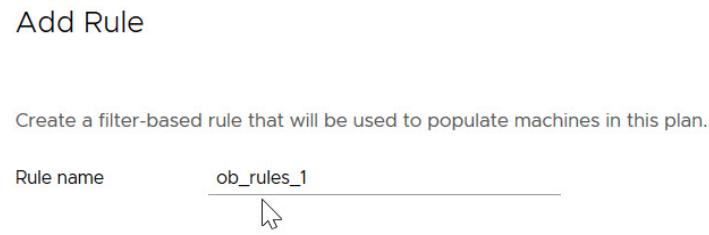


4 Click the **Rules** tab and then click **Add Rule**.

You can create one or more rules to select a group of machines for onboarding based on specific machine characteristics.

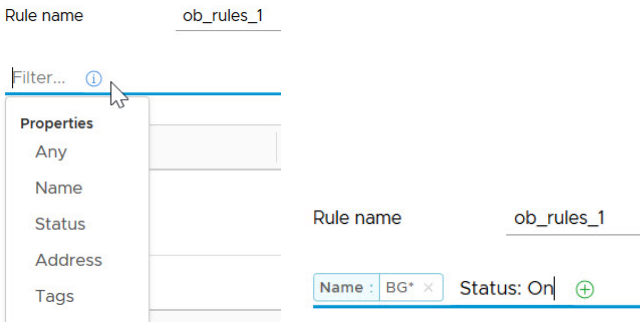


5 Enter a rule name, such as **`ob_rules_1`**.



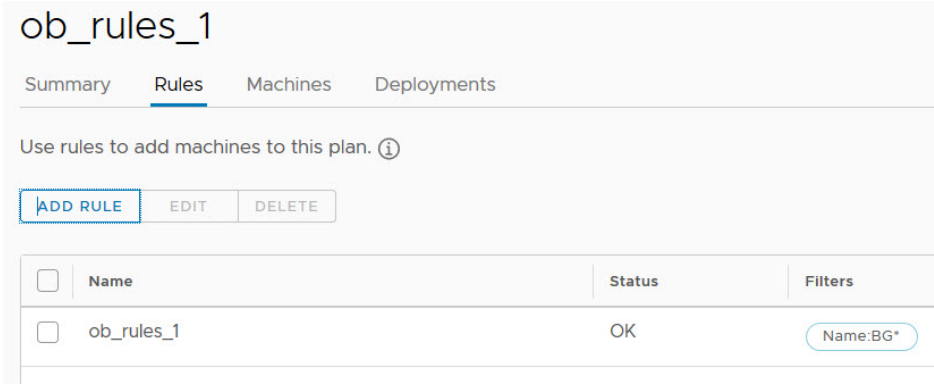
6 Build the rule by adding filters.

For this example, use the **Status** and **Name** filters on the **Filter** drop-down menu to specify all the machine whose name contains `BG*` and whose status is `On`.

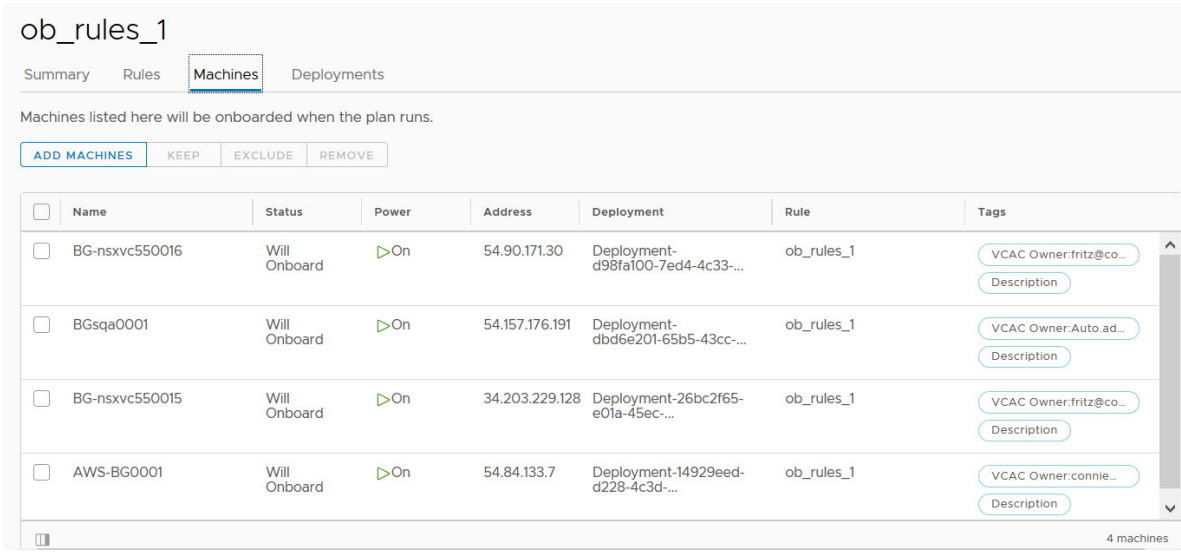


7 Click **Save**.

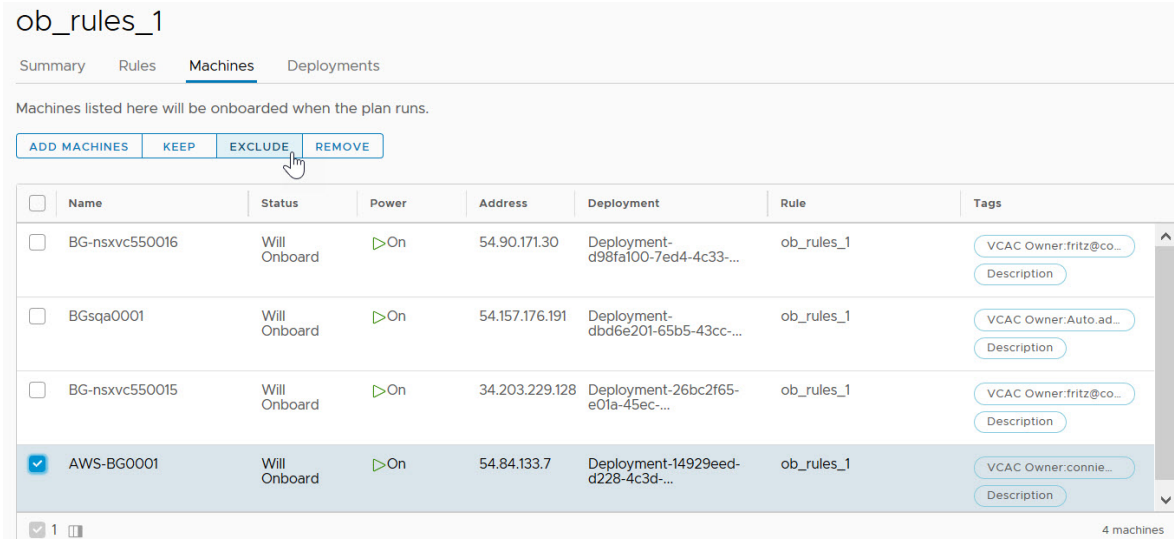
Although you can make additional rules, this example uses a single rule.



8 Click the **Machines** tab. In this example, 4 machines are selected, 3 that begin with the letters `BG` and one that contains the letters `BG`.

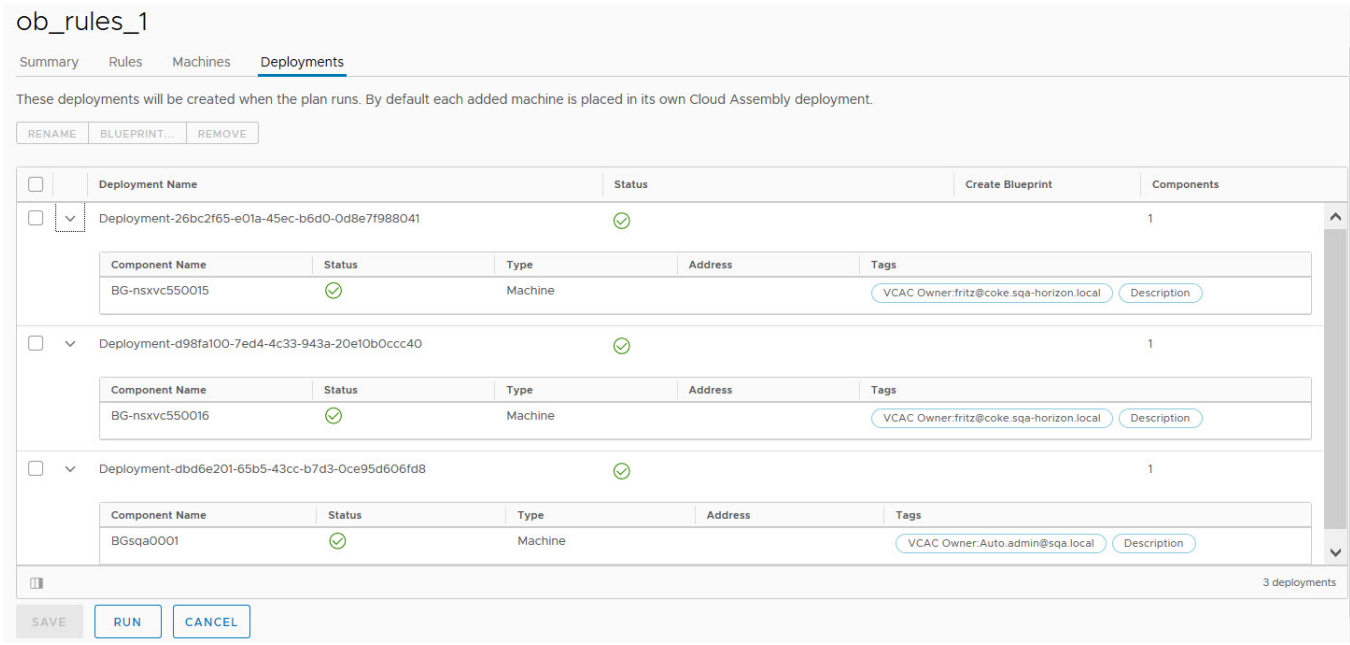


9 Remove the machine whose name does not begin with `BG` by selecting its check box and then clicking **Exclude**.

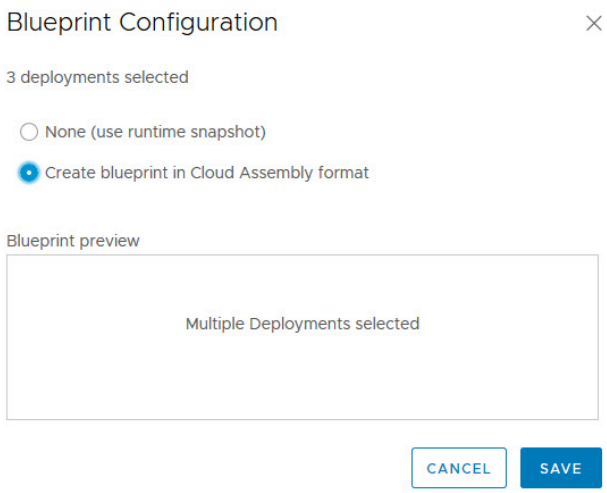


10 Click the **Deployments** tab.

The 3 machines that begin with the letters `BG` and that are powered `On` are ready to be deployed. By default, a separate blueprint and deployment is created for each machine.

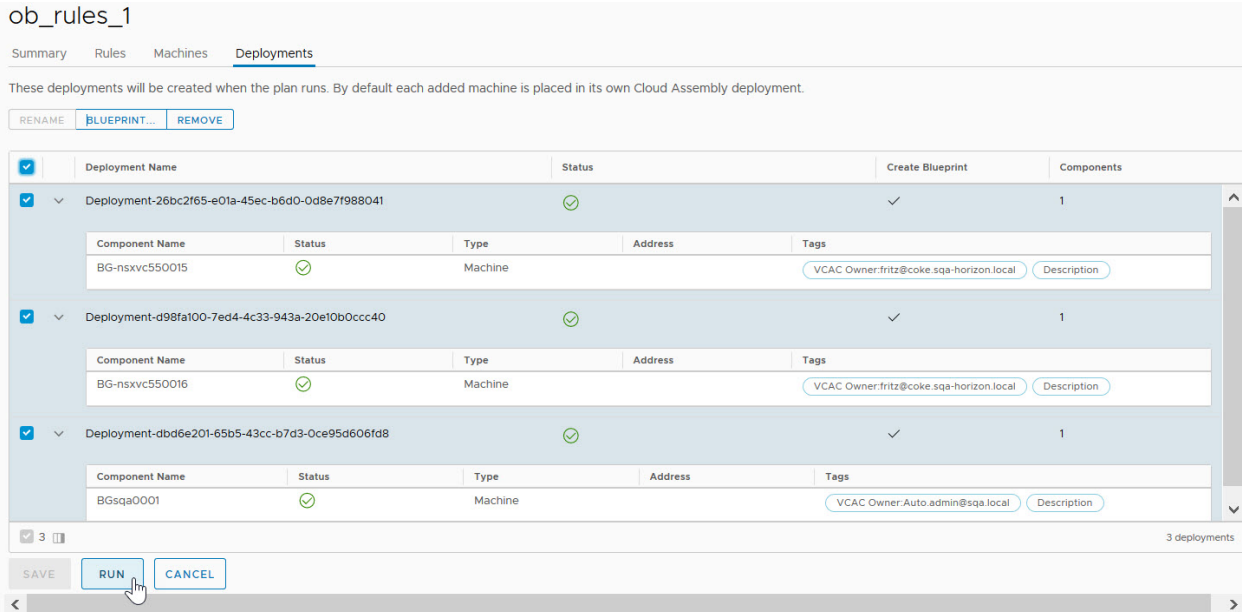


11 Click the check box next to the three deployment names, click **Blueprints**, click **Create blueprint in Cloud Assembly format**, and click **Save**.

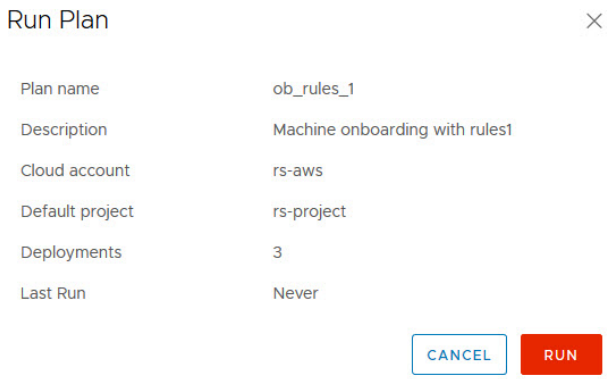


Note When your onboarding plan uses a vSphere machine, you must edit the blueprint after the onboarding process is complete. The onboarding process cannot link the source vSphere machine and its machine template, and the resultant blueprint will contain the `imageRef: "no image available"` entry in the blueprint code. The blueprint cannot be deployed until you specify the correct template name in the `imageRef:` field. To make it easier to locate and update the blueprint after the onboarding process is complete, use the **Blueprint name** option on the deployment's **Blueprint Configuration** page. Record the auto-generated blueprint name or enter and record a blueprint name of your choice. When onboarding is complete, locate and open the blueprint and replace the `"no image available"` entry in the `imageRef:` field with the correct template name.

12 On the **Deployments** page, click the check box next to the three deployment names, and click **Run**.



13 When prompted to confirm, click **Run** to onboard the machines.



The plan is run and the machines are brought into vRealize Automation Cloud Assembly management. A separate blueprint and deployment is created for each machine.

Advanced configuration for vRealize Automation Cloud Assembly environment

You can configure your vRealize Automation Cloud Assembly environment to further support project configuration, integration, and deployment.

For related and additional information about administration methods, such as using working with users and logs, and joining or leaving the Customer Experience program, see the Administering vRealize Automation help.

How do I configure an Internet proxy server for vRealize Automation

For vRealize Automation 8.0.1 forward installations on isolated networks with no direct Internet access, you can use an Internet proxy server to allow Internet by proxy functionality. The Internet proxy server supports HTTP and HTTPS.

To configure and use public cloud providers such as Amazon Web Services (AWS), Microsoft Azure, and Google Cloud Platform (GCP) as well as external integration points such as IPAM, Ansible, and Puppet, with vRealize Automation, you must configure an Internet proxy server to access the internal vRealize Automation Internet proxy server.

vRealize Automation contains an internal proxy server that communicates with your Internet proxy server. This server communicates with your proxy server if it has been configured with the `vracli proxy set ...` command. If you have not configured an Internet proxy server for your organization, then the vRealize Automation internal proxy server attempts to connect directly to the Internet.

You can set up vRealize Automation to use an Internet proxy server by using the supplied `vracli` command line utility. Information about how to use the `vracli` API is available by using the `--help` argument in the `vracli` command line, for example`vracli proxy --help`.

Access to the Internet proxy server requires use of the actions-based extensibility (ABX) On-Prem Embedded controls that are built into vRealize Automation.

Note Access to Workspace ONE Access (previously named VMware Identify Manager) is not supported by way of the Internet proxy. You cannot use the `vracli set vidm` command to access Workspace ONE Access through the Internet proxy server.

The internal proxy server requires IPv4 as its default IP format. It doesn't require Internet protocol restrictions, authentication or man-in-the-middle actions on TLS (HTTPS) certificate traffic.

Prerequisites

- Verify that you have an existing HTTP or HTTPS server, that you can use as the Internet proxy server, in the vRealize Automation network that is able to pass outgoing traffic to external sites. The connection must be configured for IPv4.
- Verify that the target Internet proxy server is configured to support IPv4 as its default IP format and not IPv6.
- If the Internet proxy server uses TLS and requires an HTTPS connection with its clients, you must import the server certificate by using one of the following commands, prior to setting the proxy configuration.
  - `vracli certificate proxy --set path_to_proxy_certificate.pem`
  - `vracli certificate proxy --set stdin`

    Use the `stdin` parameter for interactive input.

Procedure

1 Create a proxy configuration for the pods or containers that are used by Kubernetes. In this example, the proxy server is accessed by using the HTTP scheme.

```
vracli proxy set --host http://proxy.vmware.com:3128
```

2 Show the proxy configuration.

```
vracli proxy show
```

The result will be similar to:


```
{
    "enabled": true,
    "host": "10.244.4.51",
    "java-proxy-exclude": "*.local|*.localdomain|localhost|10.244.*|
192.168.*|172.16.*|kubernetes|sc2-rdops-vm06-dhcp-198-120.eng.vmware.com|10.192.204.9|
*.eng.vmware.com|sc2-rdops-vm06-dhcp-204-9.eng.vmware.com|10.192.213.146|sc2-rdops-vm06-
dhcp-213-146.eng.vmware.com|10.192.213.151|sc2-rdops-vm06-dhcp-213-151.eng.vmware.com",
    "java-user": null,
    "password": null,
    "port": 3128,
    "proxy-
exclude": ".local,.localdomain,localhost,10.244.,192.168.,172.16.,kubernetes,sc2-
rdops-vm06-dhcp-198-120.eng.vmware.com,10.192.204.9,.eng.vmware.com,sc2-
rdops-vm06-dhcp-204-9.eng.vmware.com,10.192.213.146,sc2-rdops-vm06-
dhcp-213-146.eng.vmware.com,10.192.213.151,sc2-rdops-vm06-dhcp-213-151.eng.vmware.com",
    "scheme": "http",
    "upstream_proxy_host": null,
    "upstream_proxy_password_encoded": "",
    "upstream_proxy_port": null,
    "upstream_proxy_user_encoded": "",
    "user": null,
    "internal.proxy.config": "dns_v4_first on \nhttp_port
0.0.0.0:3128\nlogformat squid %ts.%03tu %6tr %>a %Ss/%03>Hs
%<st %rm %ru %[un %Sh/%<a %mt\naccess_log stdio:/tmp/logger squid\ncoredump_dir /\ncache
deny all \nappend_domain .prelude.svc.cluster.local\nacl mylan src 10.0.0.0/8\nacl mylan
src 127.0.0.0/8\nacl mylan src 192.168.3.0/24\nacl proxy-exclude dstdomain .local\nacl
proxy-exclude dstdomain .localdomain\nacl proxy-exclude dstdomain localhost\nacl
proxy-exclude dstdomain 10.244.\nacl proxy-exclude dstdomain 192.168.\nacl proxy-exclude
dstdomain 172.16.\nacl proxy-exclude dstdomain kubernetes\nacl proxy-exclude dstdomain
10.192.204.9\nacl proxy-exclude dstdomain .eng.vmware.com\nacl proxy-exclude dstdomain
```

```
10.192.213.146\nacl proxy-exclude dstdomain 10.192.213.151\nalways_direct allow proxy-
exclude\nhttp_access allow mylan\nhttp_access deny all\n# End autogen configuration\n",
    "internal.proxy.config.type": "default"
}
```

Note If you have configured an Internet proxy server for your organization, then `"internal.proxy.config.type": "non-default"` appears in the above example instead of `'default'`. For security, the password is not shown.

Note If you use the `-proxy-exclude` parameter, you must edit the default values. For example, if you want to add `acme.com` as a domain that cannot be accessed by using the Internet proxy server, use the following steps:

a Enter `vracli proxy default-no-proxy` to obtain the default proxy-exclude settings. This is a list of automatically generated domains and networks.

b Edit the value to add `.acme.com`.

c Enter `vracli proxy set .... --proxy-exclude ...` to update the configuration settings.

d Run the `/opt/scripts/deploy.sh` command to redeploy the environment.

3 (Optional) Exclude DNS domains, FQDNs, and IP addresses from being accessed by the Internet proxy server.

Always modify the default values of the `proxy-exclude` variable using `parameter --proxy-exclude`. To add the domain `exclude.vmware.com`, first use the `vrali proxy show` command, then copy the `proxy-exclude` variable, and add the domain value using the `vracli proxy set ...` command as below:

```
vracli proxy set --host http://
proxy.vmware.com:3128 --proxy-exclude "exclude.vmware.com,docker-
registry.prelude.svc.cluster.local,localhost,.local,.cluster.local,10.244.,192.,172.16.,sc-
rdops-vm11-dhcp-75-38.eng.vmware.com,10.161.75.38,.eng.vmware.com"
```

Note Add elements to `proxy-exclude` instead of replacing values. If you delete `proxy-exclude` default values, vRealize Automation does not function properly. If this happens, delete the proxy configuration and start over.

4 After you set the Internet proxy server with `vracli proxy set ...` command, you can use the `vracli proxy apply` command to update the Internet proxy server configuration and make the latest proxy settings active.

5 If you have not already done so, activate the script changes by running the following command:

```
/opt/scripts/deploy.sh
```

6 (Optional) If needed, configure the proxy server to support external access on port 22.

To support integrations such as Puppet and Ansible, the proxy server must allow port 22 to access the relevant hosts.

Example: Sample Squid configuration

Relative to step 1, if you are setting up a Squid proxy, you can tune your configuration in `/etc/squid/squid.conf` by adapting it to the following sample:

```
acl localnet src 192.168.11.0/24

acl SSL_ports port 443

acl Safe_ports port 80 # http
acl Safe_ports port 21 # ftp
acl Safe_ports port 443 # https
acl Safe_ports port 70 # gopher
acl Safe_ports port 210 # wais
acl Safe_ports port 1025-65535 # unregistered ports
acl Safe_ports port 280 # http-mgmt
acl Safe_ports port 488 # gss-http
acl Safe_ports port 591 # filemaker
acl Safe_ports port 777 # multiling http
acl CONNECT method CONNECT

http_access allow !Safe_ports
http_access allow CONNECT !SSL_ports
http_access allow localnet

http_port 0.0.0.0:3128

maximum_object_size 5 GB
cache_dir ufs /var/spool/squid 20000 16 256
coredump_dir /var/spool/squid
refresh_pattern ^ftp: 1440 20% 10080
refresh_pattern ^gopher: 1440 0% 1440
refresh_pattern -i (/cgi-bin/|\?) 0 0% 0
refresh_pattern (Release|Packages(.gz)*)$ 0 20% 2880
refresh_pattern . 0 20% 4320

client_persistent_connections on
server_persistent_connections on
```

How do I set up a Windows template with cloud-init or cloudbase-init in vRealize Automation

You can configure your vRealize Automation environment to support cloud-init and cloudbase-init scripting for creating and using custom images in Windows deployments.

Cloud configuration scripts are supported in image mapping and blueprint code. Cloud configuration scripting adheres to cloud-init and cloudbase-init rules and formats. For related information about cloud-init and cloudbase-init, see https://cloudbase.it/cloudbase-init.

Note For information about configuring cloud-init or cloudbase-init for Windows, see the following VMware blogs:

- Windows Cloud-Init solution blog article
- Windows guest initialization with Cloudbase-Init in vCenter blog article

For related information about configuring cloud-init for Linux, see blog post Building a vRealize Automation Cloud Ready Ubuntu Template for vSphere.

For information about using cloud-init and cloud configuration scripts in vRealize Automation, see:

- Learn more about image mappings in vRealize Automation
- How to automatically initialize a machine in a vRealize Automation Cloud Assembly blueprint

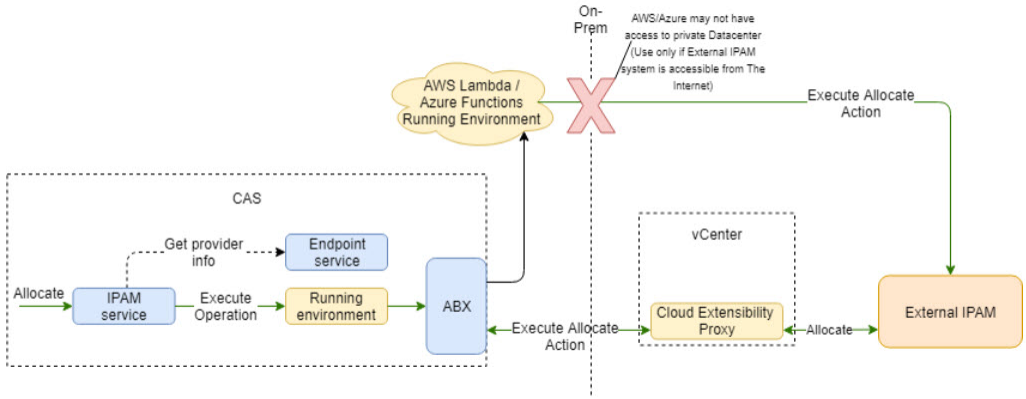
Also see these cloud-init and cloudbase-init vendor pages:

- https://cloud-init.io
- https://cloudinit.readthedocs.io/en/latest/index.html
- https://cloudinit.readthedocs.io/en/latest/topics/examples.html#yaml-examples
- https://cloudbase.it/cloudbase-init
- https://cloudbase-init.readthedocs.io/en/latest/services.html
- https://cloudbase-init.readthedocs.io/en/latest/userdata.html#userdata

How do I use the IPAM SDK to create a provider-specific external IPAM integration package for vRealize Automation

External IPAM vendors and partners can download and use the IPAM SDK to create an IPAM integration package that enables vRealize Automation to support their provider-specific IPAM solution.

The process for building and deploying a custom IPAM integration package for vRealize Automation 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 *VMware vRealize Automation Third-Party IPAM SDK* from the VMware solutions exchange marketplace.



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. You can check for a provider-specific IPAM integration package on the IPAM provider's website, in the VMware solutions exchange marketplace and from the vRealize Automation **Marketplace** tab.

While the Provider-specific external IPAM integration use case example is vendor-specific, it also contains helpful reference information.

vRealize Automation Cloud Assembly use cases

3

These use cases show examples that create resource infrastructure in vRealize Automation Cloud Assembly and then design and deploy applications onto that infrastructure.

The uses cases present example values only. Your own environment structure and naming conventions will vary.

This chapter includes the following topics:

The WordPress use case

This end-to-end vRealize Automation Cloud Assembly use case shows an example of creating infrastructure and deploying a WordPress site onto that infrastructure.

Look at the sequential setup to understand the process that brings a WordPress site to completion.

Remember that 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 needs.

Procedure

1 WordPress use case: create the infrastructure

As a cloud administrator, you first need to configure the resources where engineering can later develop, test, and put a WordPress site into production.

2 WordPress use case: create a project

A project enables the users who can provision, and configures how much provisioning is possible.

3 WordPress use case: create and expand a blueprint

As a developer, you define the WordPress site in the form of a generic vRealize Automation Cloud Assembly blueprint that can be deployed to any cloud vendor.

WordPress use case: create the infrastructure

As a cloud administrator, you first need to configure the resources where engineering can later develop, test, and put a WordPress site into production.

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

Procedure

1 WordPress use case: add cloud accounts

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

2 WordPress use case: add cloud zones

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

3 WordPress use case: add flavor mappings

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

4 WordPress use case: add image mappings

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

5 WordPress use case: add network profiles

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

6 WordPress use case: add storage profiles

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

WordPress use case: add cloud accounts

In this step, the cloud administrator adds two cloud accounts. The 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.

Setting	Sample Value
Access key ID	R5SDR3PXVV2ZW8B7YNSM
Secret access key	SZXAINXU4UHNAQ1E156S
Name	OurCo-AWS
Description	WordPress
Capabilities	cloud:aws

Remember that all values are only use case samples. 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.

Setting	Sample Value
Subscription ID	ef2avpf-dfdv-zxlugui1i-g4h0-i8ep2jwp4c9arbfe
Tenant ID	dso9wv3-4zgc-5nrcy5h3m-4skf-nnovp40wfxsro22r
Client application ID	bg224oq-3ptp-mbhi6aa05-q511-uf1yjr2sttyik6bs
Client application secret key	7uqxi57-0wtn-kymgf9wcj-t2l7-e52e4nu5fig4pmdd
Name	OurCo-Azure
Description	WordPress
Capabilities	cloud:az

7 To verify credentials, click **Validate**.

8 Click **Add**.

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

What to do next

Add cloud zones where the project will deploy the WordPress site. See WordPress use case: add cloud zones.

WordPress use case: add cloud zones

In this 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.

Prerequisites

Add cloud accounts. See WordPress use case: add cloud accounts .

Procedure

1 Go to **Infrastructure > Configure > Cloud Zones**.

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

Cloud Zone Setting	Sample Value
Account / region	OurCo-AWS/us-east-1
Name	OurCo-AWS-US-East
Description	WordPress
Placement policy	Default
Capability tags	env:dev

Remember that all values are only use case samples. 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.

Cloud Zone Setting	Sample Value
Account / region	OurCo-AWS/us-west-2
Name	OurCo-AWS-US-West
Description	WordPress
Placement policy	Default
Capability tags	env:test

Cloud Zone Setting	Sample Value
Account / region	OurCo-Azure/East US
Name	OurCo-Azure-East-US
Description	WordPress
Placement policy	Default
Capability tags	env:prod

What to do next

Account for different size machine deployments by adding flavor mappings. See WordPress use case: add flavor mappings.

WordPress use case: add flavor mappings

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

Flavor mapping is informally referred to as T-shirt sizing.

Prerequisites

Add cloud zones. See WordPress use case: add cloud zones.

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.

Setting	Sample Value
Flavor name	small
Account/region Value	OurCo-AWS/us-east-1 t2.micro
Account/region Value	OurCo-AWS/us-west-2 t2.micro
Account/region Value	OurCo-Azure/East US Standard_A0

   Remember that all values are only use case samples. Your flavors will vary.

3. Click **Create**.
4. Repeat the process twice, with values for medium and large flavors.

Setting	Sample Value
Flavor name	medium
Account/region Value	OurCo-AWS/us-east-1 t2.medium
Account/region Value	OurCo-AWS/us-west-2 t2.medium
Account/region Value	OurCo-Azure/East US Standard_A3

Setting	Sample Value
Flavor name	large
Account/region Value	OurCo-AWS/us-east-1 t2.large
Account/region Value	OurCo-AWS/us-west-2 t2.large
Account/region Value	OurCo-Azure/East US Standard_A7

What to do next

Plan for the operating system by adding image mappings. See WordPress use case: add image mappings.

WordPress use case: add image mappings

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

Each cloud zone needs a Ubuntu image mapping.

Prerequisites

Add cloud zones. See WordPress use case: add cloud zones.

Procedure

1 Go to **Infrastructure > Configure > Image Mappings**.

2 Click **New Image Mapping**, and enter values for Ubuntu servers.

Setting	Sample Value
Image name	ubuntu-16
Account/region Value	OurCo-AWS/us-east-1 ubuntu-16.04-server-cloudimg-amd64
Account/region Value	OurCo-AWS/us-west-2 ubuntu-16.04-server-cloudimg-amd64
Account/region Value	OurCo-Azure/East US azul-zulu-ubuntu-1604-923eng

Remember that all values are only use case samples. Your images will vary.

3 Click **Create**.

What to do next

Add networks. See WordPress use case: add network profiles.

WordPress use case: add network profiles

In this 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 over.

Prerequisites

Add cloud zones. See WordPress use case: add cloud zones.

Procedure

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

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

Network Profile Setting	Sample Value
Account / region	OurCo-AWS/us-east-1
Name	devnets
Description	WordPress
Capability tags	env:dev

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

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

Remember that all values are only use case samples. 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.

Network Profile Setting	Sample Value
Account / region	OurCo-AWS/us-west-2
Name	testnets
Description	WordPress
Capability tags	env:test

Network Profile Setting	Value
Account / region	OurCo-Azure/East US
Name	prodnets
Description	WordPress
Capability tags	env:prod

What to do next

Add storage. See WordPress use case: add storage profiles.

WordPress use case: add storage profiles

In this 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.

Prerequisites

Add cloud zones. See WordPress use case: add cloud zones.

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.

Storage Profile Setting	Sample Value
Account / region	OurCo-AWS/us-east-1
Name	OurCo-AWS-US-East-Disk
Description	WordPress
Device type	EBS
Volume type	General Purpose SSD
Capability tags	usage:general

Remember that all values are only use case samples.

3 Click **Create**.

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

Storage Profile Setting	Sample Value
Account / region	OurCo-AWS/us-west-2
Name	OurCo-AWS-US-West-Disk
Description	WordPress
Device type	EBS
Volume type	General Purpose SSD
Capability tags	usage:general

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

Storage Profile Setting	Sample Value
Account / region	OurCo-Azure/East US
Name	OurCo-Azure-East-US-Disk
Description	WordPress
Storage type	Managed disks
Disk type	Premium LRS
OS disk caching	Read only
Data disk caching	Read only
Capability tags	usage:fast

What to do next

Create a project to identify users, and to define provisioning settings. See WordPress use case: create a project.

WordPress use case: create a project

A 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

Prerequisites

Add cloud zones. See WordPress use case: add cloud zones.

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 vRealize Automation Cloud Assembly for the user.

Remember that addresses shown here are only use case samples.

- 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.

Project Cloud Zone Setting	Sample Value
Cloud zone	OurCo-AWS-US-East
Provisioning priority	1
Instances limit	5
Cloud zone	OurCo-AWS-US-West
Provisioning priority	1
Instances limit	5
Cloud zone	OurCo-Azure-East-US
Provisioning priority	0
Instances limit	1

7 Click **Create**.

8 Go to **Infrastructure > Configure > Cloud Zones**, and open a zone that was created in WordPress use case: add cloud zones.

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

10 Check the other zones created in WordPress use case: add cloud zones.

What to do next

Create a basic blueprint.

WordPress use case: create and expand a blueprint

As a developer, you define the WordPress site in the form of a generic vRealize Automation Cloud Assembly blueprint that can be deployed to any cloud vendor.

The use case blueprint consists of a WordPress application server, MySQL database server, and supporting components that are deployable to AWS, Azure, or vSphere-based clouds. The blueprint starts with a few components, and then grows as you modify existing components and add more components.

The examples from WordPress use case: create the infrastructure included 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 16 configured in each zone.
- Network profiles with internal and external subnets for each zone: devnets, testnets, prodnets.
- Storage to support an archive disk, general storage for development and test, with fast storage for production.
- The WordPress project includes all three cloud zone environments plus users who can try the use case.

Prerequisites

Be familiar with your infrastructure values. For example, the use case example uses AWS for development and test, and Azure for production. When creating your own blueprint, substitute your own values, typically set by your cloud administrator.

Procedure

1 WordPress use case: create a basic blueprint

  As a developer, you start with a vRealize Automation Cloud Assembly blueprint that contains only minimal WordPress components, such as having only one application server.

2 WordPress use case: test a basic blueprint

  During development, you are typically building a vRealize Automation Cloud Assembly blueprint by starting with the essentials, then deploying and testing as the blueprint grows.

WordPress use case: create a basic blueprint

As a developer, you start with a vRealize Automation Cloud Assembly blueprint that contains only minimal WordPress components, such as having only one application server.

vRealize Automation Cloud Assembly is an infrastructure-as-code tool. You drag components 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.

Prerequisites

Be familiar with your infrastructure. The examples shown here use the infrastructure values from WordPress use case: create the infrastructure, but you would substitute your own.

Procedure

1. Go to **Blueprints**, and click **New**.
2. Name the blueprint **`Wordpress-BP`**.
3. Select the **WordPress** project, and click **Create**.
4. From the components on the left of the blueprint 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:

```
resources:
  DBTier:
    type: Cloud.Machine
    properties:
        name: mysql
        image: 'ubuntu-16'
        flavor: 'small'
        constraints:
            -   tag: env:dev
  WebTier:
    type: Cloud.Machine
    properties:
        name: wordpress
```

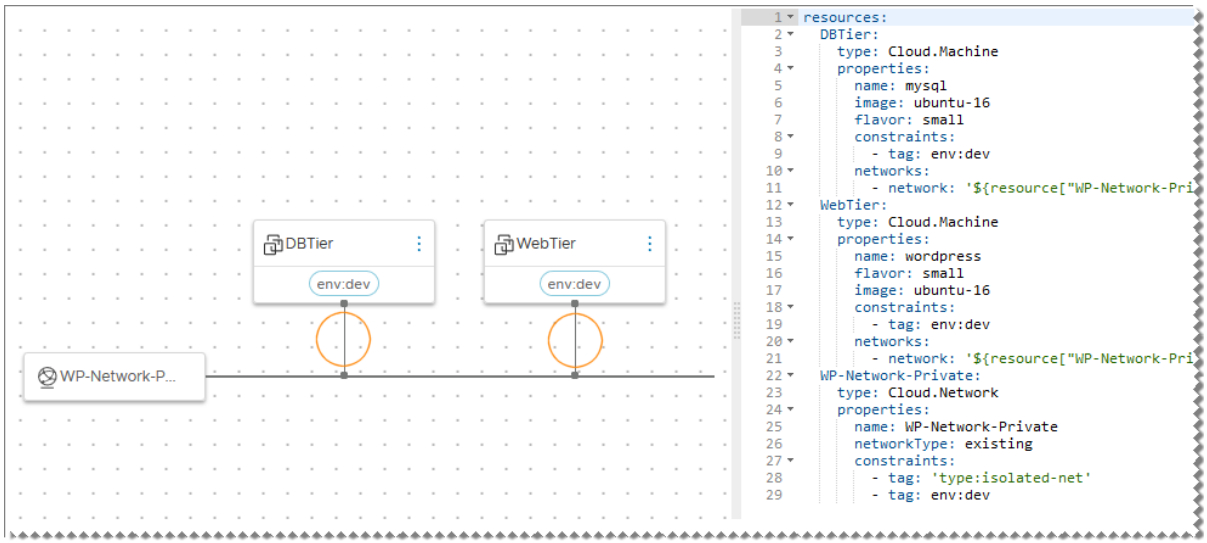
```
        image: 'ubuntu-16'
        flavor: 'small'
        constraints:
            - tag: env:dev
```

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
       constraints:
           - tag: 'type:isolated-net'
           - tag: 'env:dev'
```

7 Connect the machines to the network:

Click and hold where the line touches the network block, drag to a machine block, and release.



In the editor, notice that the network code gets added to the two machines:

```
resources:
  DBTier:
    type: Cloud.Machine
    properties:
        name: mysql
        image: 'ubuntu-16'
        flavor: 'small'
        constraints:
            -   tag: env:dev
        networks:
            - network: '${resource["WP-Network-Private"].id}'
  WebTier:
    type: Cloud.Machine
```

```
    properties:
        name: wordpress
        image: 'ubuntu-16'
        flavor: 'small'
        constraints:
            - tag: env:dev
        networks:
            - network: '${resource["WP-Network-Private"].id}'
```

8 Add user input prompting.

In some places, the use case 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
- Storage disk speeds for general and fast usage

You might set a specific option directly in the blueprint, but a better approach is to let the user select the option at blueprint deployment time. Prompting for user input lets you create one blueprint that can be deployed many ways, instead of having many hard-coded blueprints.

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:

```
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:

```
resources:
  DBTier:
    type: Cloud.Machine
    properties:
        name: mysql
        image: 'ubuntu-16'
        flavor: '${input.size}'
        constraints:
            -   tag: '${input.env}'
        networks:
            - network: '${resource["WP-Network-Private"].id}'
  WebTier:
    type: Cloud.Machine
    properties:
        name: wordpress
        image: 'ubuntu-16'
        flavor: '${input.size}'
        constraints:
            - tag: '${input.env}'
        networks:
            - network: '${resource["WP-Network-Private"].id}'
  WP-Network-Private:
    type: Cloud.Network
```

```
    properties:
        name: WP-Network-Private
        networkType: existing
        constraints:
            - tag: 'type:isolated-net'
            - tag: '${input.env}'
```

9 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, a database disk, and deployment-time cloudConfig initialization scripts.

Component	Example

Additional DBTier Inputs	<pre>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
	databaseDiskSize:
	  type: number
	  default: 4
	  maximum: 10
	  title: MySQL Data Disk Size
	  description: Database Disk Size</pre>

DBTier Resource	<pre>DBTier:
	  type: Cloud.Machine
	  properties:
	  name: mysql
	  image: ubuntu-16
	  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: []</pre>

WebTier Resource	<pre>WebTier:
	  type: Cloud.Machine
	  properties:
	  name: wordpress
	  flavor: '${input.size}'</pre>

Component	Example
	image: ubuntu-16 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: - mkdir -p /var/www/html/mywordpresssite && cd /var/www/html && wget https://wordpress.org/latest.tar.gz && tar -xzf /var/www/html/latest.tar.gz -C /var/www/html/mywordpresssite --strip-components 1 - 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 ${DBTier.networks[0].address} -e "create database wordpress_blog;" - mv /var/www/html/mywordpresssite/wp-config-sample.php /var/www/html/mywordpresssite/wp-config.php - sed -i -e s/"define('DB_NAME', 'database_name_here');"/"define('DB_NAME', 'wordpress_blog');"/ /var/www/html/mywordpresssite/wp-config.php && sed -i -e s/"define('DB_USER', 'username_here');"/"define('DB_USER', 'root');"/ /var/www/html/mywordpresssite/wp-config.php && sed -i -e s/"define('DB_PASSWORD', 'password_here');"/"define('DB_PASSWORD', 'mysqlpassword');"/ /var/www/html/mywordpresssite/wp-config.php && sed -i -e s/"define('DB_HOST', 'localhost');"/"define('DB_HOST', '${DBTier.networks[0].address}');"/ /var/www/html/mywordpresssite/wp-config.php - service apache2 reload

Example: Completed Basic Blueprint Code Example

```
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@#$]+'
    encrypted: true
    title: Database Password
    description: Database Password
  databaseDiskSize:
    type: number
    default: 4
    maximum: 10
    title: MySQL Data Disk Size
    description: Database Disk Size
resources:
  DBTier:
    type: Cloud.Machine
    properties:
      name: mysql
      image: ubuntu-16
      flavor: '${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: []
  WebTier:
```

```
    type: Cloud.Machine
    properties:
      name: wordpress
      flavor: '${input.size}'
      image: ubuntu-16
      constraints:
        - tag: '${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:
         - mkdir -p /var/www/html/mywordpresssite && cd /var/www/html && wget
https://wordpress.org/latest.tar.gz && tar -xzf /var/www/html/latest.tar.gz -C /var/www/html/
mywordpresssite --strip-components 1
         - 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 ${DBTier.networks[0].address} -e "create database
wordpress_blog;"
         - mv /var/www/html/mywordpresssite/wp-config-sample.php /var/www/html/
mywordpresssite/wp-config.php
         - sed -i -e s/"define( 'DB_NAME', 'database_name_here' );"/"define( 'DB_NAME',
'wordpress_blog' );"/ /var/www/html/mywordpresssite/wp-config.php && sed
-i -e s/"define( 'DB_USER', 'username_here' );"/"define( 'DB_USER',
'root' );"/ /var/www/html/mywordpresssite/wp-config.php && sed -i
-e s/"define( 'DB_PASSWORD', 'password_here' );"/"define( 'DB_PASSWORD',
'mysqlpassword' );"/ /var/www/html/mywordpresssite/wp-config.php && sed
-i -e s/"define( 'DB_HOST', 'localhost' );"/"define( 'DB_HOST', '$
{DBTier.networks[0].address}' );"/ /var/www/html/mywordpresssite/wp-config.php
         - service apache2 reload
  WP-Network-Private:
    type: Cloud.Network
    properties:
      name: WP-Network-Private
      networkType: existing
      constraints:
        - tag: 'type:isolated-net'
        - tag: '${input.env}'
```

What to do next

Test the blueprint by checking the syntax and deploying it.

WordPress use case: test a basic blueprint

During development, you are typically building a vRealize Automation Cloud Assembly blueprint by starting with the essentials, then deploying and testing as the blueprint grows.

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

Prerequisites

Create the basic blueprint. See WordPress use case: create a basic blueprint.

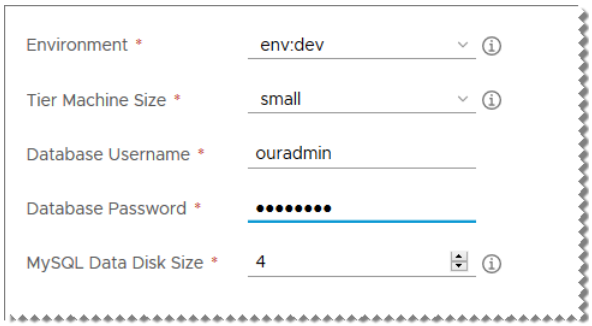
Procedure

1 Click **Blueprints**, and open the WordPress-BP blueprint.

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

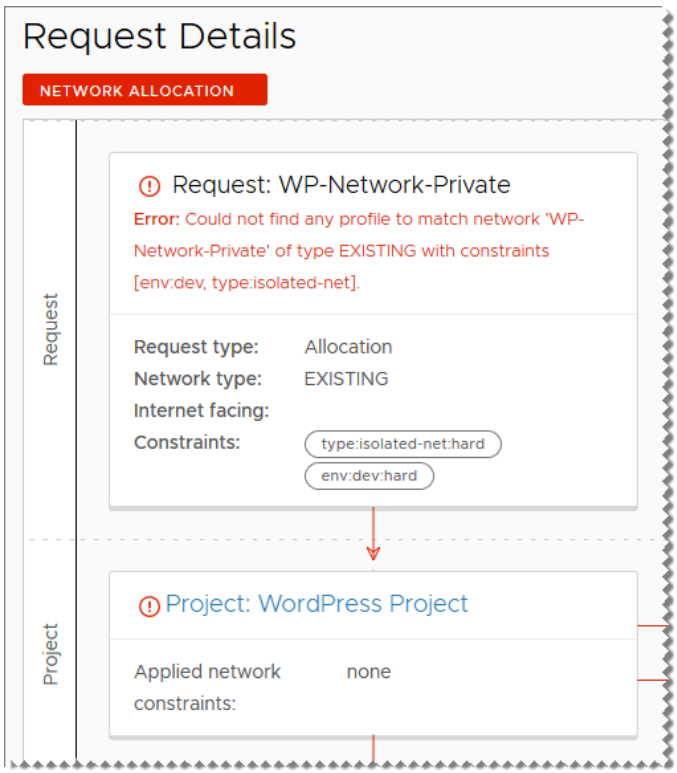
2 To check blueprint syntax, placement, and basic validity, click **Test** at the lower left.

3 Enter input values, and click **Test**.



The test is only a simulation and does not actually deploy virtual machines or other resources. The simulation exposes potential issues, such as not having any resource capabilities defined that match hard constraints in the blueprint.

The test includes a link to a **Provisioning Diagram**, where you can inspect the simulated deployment flow and see any errors that occurred.



A successful simulation doesn't guarantee that you can deploy the blueprint without errors.

4 After the blueprint 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 blueprint successfully deployed, look under **Deployments**.

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

Timestamp	Status	Resource Type	Resource Name	Details
August 13, 2019, 9:12:56 AM	REQUEST_FINISHED			
August 13, 2019, 9:12:55 AM	CREATE_FINISHED	Cloud.Machine	WebTier	
August 13, 2019, 9:12:30 AM	CREATE_IN_PROGRESS	Cloud.Machine	WebTier	
August 13, 2019, 9:12:30 AM	CREATE_FINISHED	Cloud.Machine	DBTier	
August 13, 2019, 9:12:05 AM	CREATE_IN_PROGRESS	Cloud.Machine	DBTier	
August 13, 2019, 9:12:05 AM	CREATE_FINISHED	Cloud.Network	WP-Network-Private	

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 vRealize Automation 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 > 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 use case, 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 blueprint changes and redeploy using the **Update an existing deployment** option.

11 Consider versioning the blueprint. You can revert to a working version if a change causes deployment to fail.

a On the blueprint 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 blueprint, select **`medium`** at deployment time, redeploy, and verify the application again.

What to do next

Expand the blueprint into a production-worthy application by adding more components.

WordPress use case: expand a blueprint

After you create and test a basic vRealize Automation Cloud Assembly blueprint, you expand it into a multiple tier application that is deployable to development, test, and eventually production.

To expand the blueprint, 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 blueprint and test it. See WordPress use case: create a basic blueprint and WordPress use case: test a basic blueprint.

Procedure

1 Click **Blueprints**, and open the WordPress-BP blueprint.

The basic blueprint 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.

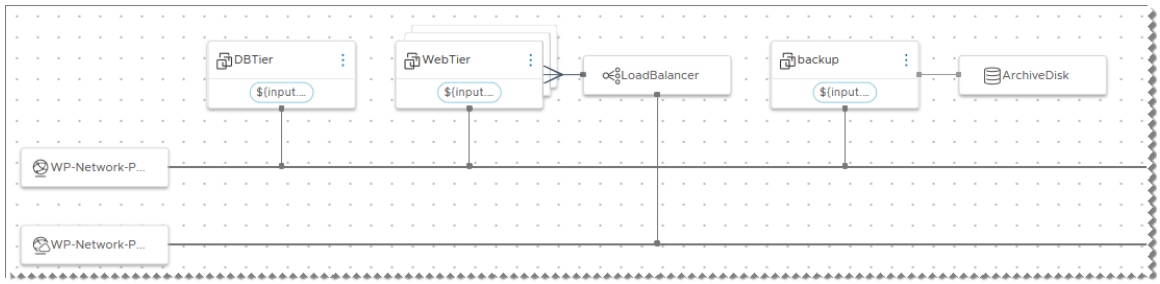
g Connect the load balander to the public network.

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 `archiveusage` input prompt for the storage disk speed.

k Add an `archiveDiskSize` input prompt for the storage disk size.



3 Deploy, test, and make changes in the same way that you did for the basic blueprint.

You can update existing deployments, or even deploy new instances so that you can compare deployments.

The goal is to reach a solid, repeatable blueprint that can be used for production deployments.

Example: Completed Expanded Blueprint Code Example

```
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
  databaseDiskSize:
    type: number
    default: 4
    maximum: 10
    title: MySQL Data Disk Size
    description: Database Disk Size
  count:
    type: integer
    default: 2
    maximum: 5
    minimum: 2
    title: WordPress Cluster Size
    description: WordPress Cluster Size (Number of Nodes)
  archiveDiskSize:
    type: number
    default: 4
    maximum: 10
    title: WordPress Archive Disk Size
    description: Archive Storage Disk Speed
  archiveusage:
    type: string
```

```
    enum:
      - 'usage:general'
      - 'usage:fast'
    description: Archive Storage Disk Speed
    title: Archive Disk Speed
resources:
  DBTier:
    type: Cloud.Machine
    properties:
      name: mysql
      image: ubuntu-16
      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: []
  WebTier:
    type: Cloud.Machine
    properties:
      name: wordpress
      flavor: '${input.size}'
      image: 'ubuntu-16'
      count: '${input.count}'
      constraints:
        - tag: '${input.env}'
      networks:
        - network: '${resource["WP-Network-Private"].id}'
          assignPublicIpAddress: true
      storage:
        disks:
          - capacityGb: '${input.archiveDiskSize}'
            name: ArchiveDisk
      cloudConfig: |
        #cloud-config
        repo_update: true
        repo_upgrade: all
```

```
        packages:
         - apache2
         - php
         - php-mysql
         - libapache2-mod-php
         - php-mcrypt
         - mysql-client

        runcmd:
         - mkdir -p /var/www/html/mywordpresssite && cd /var/www/html && wget
https://wordpress.org/latest.tar.gz && tar -xzf /var/www/html/latest.tar.gz -C /var/www/html/
mywordpresssite --strip-components 1
         - 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 ${DBTier.networks[0].address} -e "create database
wordpress_blog;"
         - mv /var/www/html/mywordpresssite/wp-config-sample.php /var/www/html/
mywordpresssite/wp-config.php
         - sed -i -e s/"define( 'DB_NAME', 'database_name_here' );"/"define( 'DB_NAME',
'wordpress_blog' );"/ /var/www/html/mywordpresssite/wp-config.php && sed
-i -e s/"define( 'DB_USER', 'username_here' );"/"define( 'DB_USER',
'root' );"/ /var/www/html/mywordpresssite/wp-config.php && sed -i
-e s/"define( 'DB_PASSWORD', 'password_here' );"/"define( 'DB_PASSWORD',
'mysqlpassword' );"/ /var/www/html/mywordpresssite/wp-config.php && sed
-i -e s/"define( 'DB_HOST', 'localhost' );"/"define( 'DB_HOST', '$
{DBTier.networks[0].address}' );"/ /var/www/html/mywordpresssite/wp-config.php &&
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
  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
      constraints:
        - tag: 'type:isolated-net'
        - tag: '${input.env}'
  WP-Network-Public:
    type: Cloud.Network
    properties:
      name: WP-Network-Public
      networkType: public
      constraints:
        - tag: 'type:public-net'
        - tag: '${input.env}'
  backup:
    type: Cloud.Machine
    properties:
      name: backup
      flavor: '${input.size}'
      image: 'ubuntu-16'
      networks:
        - network: '${resource["WP-Network-Private"].id}'
      constraints:
        - tag: '${input.env}'
      attachedDisks:
        - source: '${ArchiveDisk.id}'
  ArchiveDisk:
    type: Cloud.Volume
    properties:
      name: ArchiveDisk
      capacityGb: 5
      constraints:
        - tag: '${input.archiveusage}'
        - tag: '${input.env}'
```

What to do next

Define your own infrastructure and create your own blueprints.

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

VMware Cloud on AWS use case

This vRealize Automation Cloud Assembly use case shows the process of defining resource infrastructure and blueprint 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.

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 Assembly

This vRealize Automation Cloud Assembly use case shows the process of defining resource infrastructure and a corresponding blueprint for deployment to a VMware Cloud on AWS environment.

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

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

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

This vRealize Automation Cloud Assembly use case shows the process of defining resource infrastructure and a corresponding blueprint for deployment to a VMware Cloud on AWS environment.

In this procedure, you configure infrastructure that supports blueprint deployment to resources in your existing VMware Cloud on AWS environment.

Prerequisites

- Before you can create and configure a VMware Cloud on AWS cloud account in vRealize Automation 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.

- To facilitate the needed connection between your existing VMware Cloud on AWS host SDDC in vCenter and a VMware Cloud on AWS cloud account in vRealize Automation Cloud Assembly, you must provide a network connection, and add firewall rules, by using a VPN or similar networking means. See Prepare your VMware Cloud on AWS SDDC to connect with VMware Cloud on AWS cloud accounts in vRealize Automation.

Procedure

1 Prepare your VMware Cloud on AWS SDDC to connect with VMware Cloud on AWS cloud accounts in vRealize Automation

   When using VMware Cloud on AWS cloud accounts in your vRealize Automation Cloud Assembly on-premises environment, you must create a network connection to support communication between your SDDC in vCenter and any VMware Cloud on AWS cloud accounts in vRealize Automation.

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

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

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

   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 Assembly.

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

   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 Assembly.

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

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

6 Define a vCenter machine resource in a blueprint design to support VMware Cloud on AWS deployment in vRealize Automation Cloud Assembly

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

Prepare your VMware Cloud on AWS SDDC to connect with VMware Cloud on AWS cloud accounts in vRealize Automation

When using VMware Cloud on AWS cloud accounts in your vRealize Automation Cloud Assembly on-premises environment, you must create a network connection to support communication

between your SDDC in vCenter and any VMware Cloud on AWS cloud accounts in vRealize Automation.

To facilitate the needed connection between your existing VMware Cloud on AWS host SDDC in vCenter and a VMware Cloud on AWS cloud account in vRealize Automation, you must provide a network connection between the two elements by using a VPN or similar networking means.

Procedure

1 Configure a VPN connection over the public Internet or AWS Direct connect.

   See *VMware Cloud on AWS Networking and Security* at VMware Cloud on AWS Documentation.

2 Verify that the vCenter Server FQDN is resolvable at a private IP address on the management network.

   See *VMware Cloud on AWS Networking and Security* at VMware Cloud on AWS Documentation.

3 Configure needed firewall rules.

   You must configure management gateway firewall rules in the SDDC's VMware Cloud on AWS console to support 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 vRealize Automation appliance/server or vRealize Automation load balancer VIP.
   - Limit network traffic to vCenter for ICMP (All ICMP), SSO (TCP 7444), and HTTPS (TCP 443) services to the discovered IP address of the vRealize Automation appliance/server or vRealize Automation load balancer VIP.
   - Limit network traffic to the NSX-T Manager for HTTPS (TCP 443) services to the discovered IP address of the vRealize Automation appliance/server or vRealize Automation load balancer VIP.

   The required firewall rules are summarized in the following table.

Table 3-1. Required Management Gateway Firewall Rules Summary

Name	Source	Destination	Service
vCenter	CIDR block of on-premises data center	vCenter	Any (All Traffic)
vCenter ping	Any	vCenter	ICMP (All ICMP)
NSX Manager	CIDR block of on-premises data center	NSX Manager	Any (All Traffic)
On pemises to ESXi ping	CIDR block of on-premises data center	ESXi Management Only	ICMP (All ICMP)

Table 3-1. Required Management Gateway Firewall Rules Summary (continued)

Name	Source	Destination	Service
On Premises to ESXi remote console and provisioning	CIDR block of on-premises data center	ESXi Management Only	TCP 902
On-premises to SDDC VM	CIDR block of on-premises data center	CIDR block of SDDC logical network	Any (All Traffic)
SDDC VM to on premises	CIDR block of SDDC logical network	CIDR block of on-premises data center	Any (All Traffic)

For related information, see *VMware Cloud on AWS Networking and Security* and *VMware Cloud on AWS Operations Guide* at VMware Cloud on AWS Documentation.

Results

After you have configured required gateway access and firewall rules, you can continue with the process of creating a VMware Cloud on AWS cloud account.

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

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

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

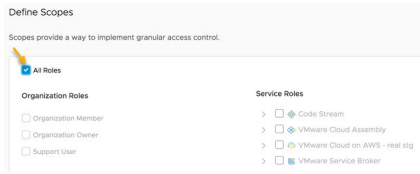
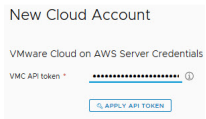
- This procedure assumes that you have the required administrator credentials, including VMware Cloud on AWS CloudAdmin credentials for the target SDDC in vCenter and that you have enabled HTTPS access on port 443. See Credentials required for working with cloud accounts in vRealize Automation.
- This procedure assumes that you have the cloud administrator user role. See What are the vRealize Automation Cloud Assembly user roles.
- To facilitate the needed connection between your existing VMware Cloud on AWS host SDDC in vCenter and a VMware Cloud on AWS cloud account in vRealize Automation, you must provide a network connection, and firewall rules, by using a VPN or similar networking means. See Prepare your VMware Cloud on AWS SDDC to connect with VMware Cloud on AWS cloud accounts in vRealize Automation. If you are using an external HTTP Internet proxy, it must be configured for IPv4.
- If you do not have external Internet access, configure an Internet server proxy. See How do I configure an Internet proxy server for vRealize Automation.

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.

Setting	Sample Value and Instruction	Description
VMC API Token	1 Click the *i* 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*_mytoken**. 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 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 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 and do not appear in the list of available SDDCs.

Setting	Sample Value and Instruction	Description
vCenter IP address/ FQDN	The address auto-populates based on your SDDC selection.	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.
NSX Manager IP address/FQDN	The address auto-populates based on your SDDC selection.	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.
vCenter user name and password	The user name auto-populates as cloudadmin@vmc.local.	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.
Validate	Click **Validate**.	Validate confirms your access rights to the specified vCenter and checks that the vCenter is running.
Name and Description	Enter **`OurCo-VMC`** for the cloud account name. Enter **`Sample deployment for VMC`** for the cloud account description.	
Allow provisioning to these data centers	This information is read-only.	Lists available data centers in your specified VMware Cloud on AWS SDDC environment.
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 vRealize Automation 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 vRealize Automation Cloud Assembly resources and deployments and Creating a Tagging Strategy.

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 **Infrastructure** tab.

What to do next

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

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

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 Assembly.

In VMware Cloud on AWS, the two primary administrator credentials are CloudGlobalAdmin and CloudAdmin. vRealize Automation 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 blueprint deploys machines, networks, and storage. See Learn more about vRealize Automation 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 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.
- This procedure assumes that you have the cloud administrator user role. See What are the vRealize Automation Cloud Assembly user roles.

Procedure

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

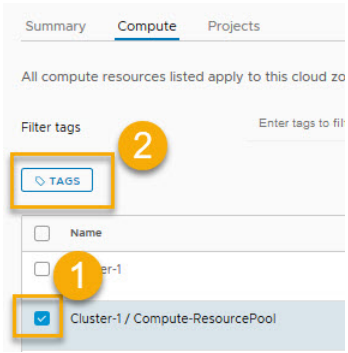
2 Click **New Cloud Zone**, and enter values for the VMware Cloud on AWS environment.

Setting	Sample Value
Account / region	OurCo-VMC / Datacenter:Datacenter-abz This is the cloud account and associated region that you defined in the previous step, Create a VMware Cloud on AWS cloud account in vRealize Automation within a sample workflow.
Name	VMC_cloud_zone-1
Description	VMware Cloud on AWS resources only
Placement policy	Default
Capability tags	Leave this empty. This workflow does not use capability tags.

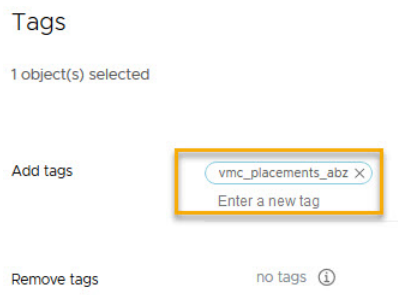
3 Click the **Compute** tab.

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.

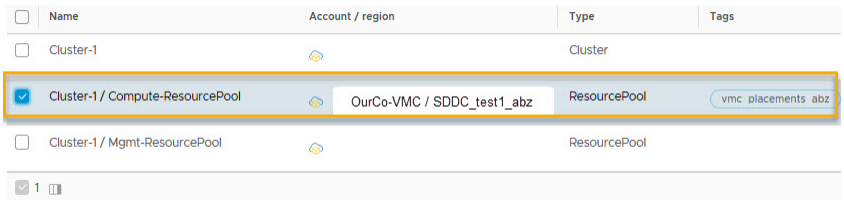


5 As shown in area 2 above, add the tag name `vmc_placements_abz`.



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

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 Assembly.

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

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 Assembly.

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 blueprint.

For general information about mappings and profiles, see Chapter 4 Building your vRealize Automation 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 Assembly.
- 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.
- This procedure assumes that you have the cloud administrator user role. See What are the vRealize Automation Cloud Assembly user roles.

Procedure

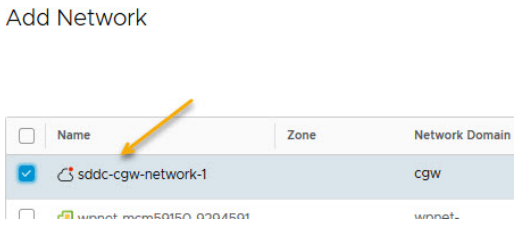
1 Define a network profile for VMware Cloud on AWS deployments.

a Select **Infrastructure > Configure > Network Profiles** and click **New Network Profile**.

Setting	Sample value
Account / region	OurCo-VMC / Datacenter:Datacenter-abz **Note** 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 within a sample workflow.
Name	vmc-network1
Description	Contains networks that can be accessed by blueprint administrators who have VMware Cloud on AWS CloudAdmin credentials.

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.

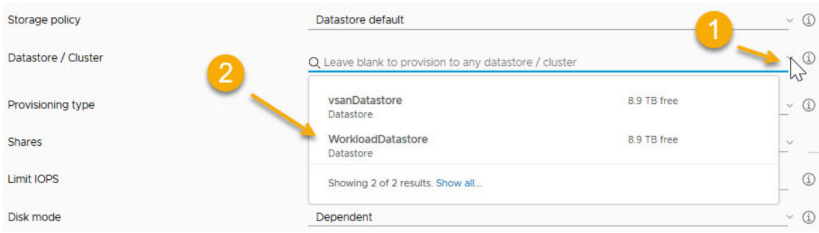
3 Define a storage profile for VMware Cloud on AWS deployments.

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**.

Setting	Sample Value
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 within a sample workflow.
Name	vmc-storage1
Description	Contains the datastore cluster that can be deployed to by blueprint 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 vRealize Automation 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 Assembly.

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

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

For information about projects, see How do vRealize Automation 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 Assembly 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.
- This procedure assumes that you have the cloud administrator user role. See What are the vRealize Automation Cloud Assembly user roles.

Procedure

1. Select **Infrastructure > Configure > 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.

Setting	Sample Value
Cloud zone	VMC_cloud_zone-1 You created this cloud zone in the earlier step, Create a cloud zone for VMware Cloud on AWS deployments in vRealize Automation Cloud Assembly.
Provisioning priority	1
Instances limit	3

6. For this example, ignore the other options.

What to do next

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

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

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

Create a blueprint 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 blueprint designer credentials. See What are the vRealize Automation Cloud Assembly 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.
- Configure the resource infrastructure and project as described in the preceding sections.

Procedure

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

Setting	Sample Value
Name	vmc-bp_abz
Description	1
Project	VMC_proj-1_abz 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) blueprint resource code in the machine resource.

```
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 blueprint 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 blueprint design 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:

- Project setting overrides the blueprint design setting and the cloud zone setting.
- Blueprint design setting overrides the cloud zone setting.

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

```
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 blueprint design, 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 Assembly.

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

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

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

vRealize Automation Cloud Assembly 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 vRealize Automation Cloud Assembly blueprint:

- 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 VMware Cloud on AWS use case workflow.

Procedure

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

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 blueprint to support network isolation for VMware Cloud on AWS in vRealize Automation Cloud Assembly

In this step, you drag a network machine component onto a vRealize Automation Cloud Assembly blueprint 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 Assembly

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

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

You can configure network isolation for your VMware Cloud on AWS deployment needs by specifying and using on-demand network settings in a vRealize Automation Cloud Assembly 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 blueprint and use the blueprint 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 Assembly workflow.
- Review Configure an isolated network in VMware Cloud on AWS workflow in vRealize Automation Cloud Assembly.

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

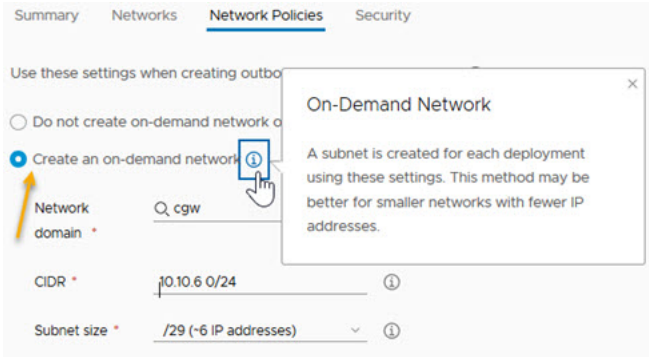
Procedure

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 Assembly.

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 blueprint. See Define a network component in a blueprint to support network isolation for VMware Cloud on AWS in vRealize Automation Cloud Assembly

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

You can configure network isolation for your VMware Cloud on AWS deployment needs by specifying and using an on-demand security group in a vRealize Automation Cloud Assembly 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 blueprint and use the blueprint in a VMware Cloud on AWS deployment.

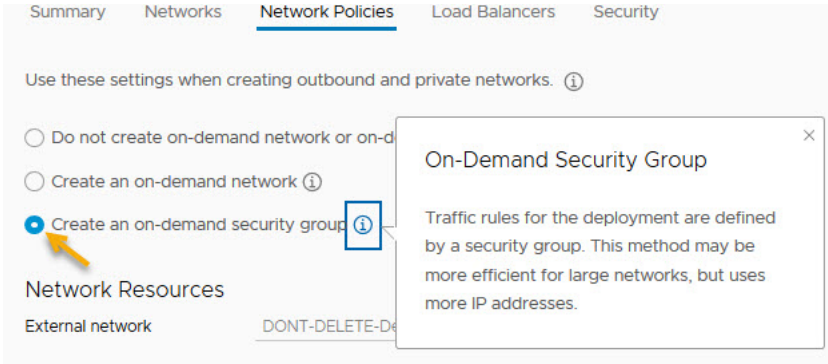
Unless otherwise indicated, the step values that you enter in this procedure are for this example workflow only.

Prerequisites

- Complete the Configure a basic VMware Cloud on AWS workflow in vRealize Automation Cloud Assembly workflow.
- Review Configure an isolated network in VMware Cloud on AWS workflow in vRealize Automation Cloud Assembly.
- 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.
- This procedure assumes that you have the cloud administrator user role. See What are the vRealize Automation Cloud Assembly 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 Assembly .
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 Assembly.
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 blueprint. See Define a network component in a blueprint to support network isolation for VMware Cloud on AWS in vRealize Automation Cloud Assembly

Define a network component in a blueprint to support network isolation for VMware Cloud on AWS in vRealize Automation Cloud Assembly

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

Add network isolation to the blueprint that you created earlier. The blueprint 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 Assembly or Configure on-demand network-based isolation in vRealize Automation Cloud Assembly procedure.
- This procedure assumes that you have blueprint designer credentials. See What are the vRealize Automation Cloud Assembly 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.

Procedure

1. Open the blueprint that you created in the previous workflow. See Define a vCenter machine resource in a blueprint design to support VMware Cloud on AWS deployment in vRealize Automation Cloud Assembly.
2. From the components on the left of the blueprint 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 blueprint.

Provider-specific external IPAM integration use case

You can use an external IPAM provider to manage IP address assignments for your blueprint deployments. This example procedure describes how to configure external IPAM integration in vRealize Automation Cloud Assembly and deploy blueprints that obtain their IP address allocation from that external IPAM provider. This example describes how to configure an IPAM integration 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 blueprint 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.

Procedure

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

  Before you can download and deploy the Infoblox provider package (`infoblox.zip`) for integration with vRealize Automation 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 Assembly

Before you can define an external IPAM integration point in vRealize Automation Cloud Assembly, you need a configured IPAM provider package. You can download a provider package from your IPAM provider's website or from the vRealize Automation Cloud Assembly Marketplace.

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

Before you can define a external IPAM integration point in vRealize Automation, you need to create or access an existing running environment to serve as an intermediary between the IPAM provider and vRealize Automation. 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 point in vRealize Automation

vRealize Automation supports integration with an external IPAM providers. You can use a provider-specific IPAM integration point to obtain and manage IP addresses and related network characteristics for blueprint deployments.

5 Configure a network and network profile in vRealize Automation Cloud Assembly to use IPAM provider values

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

6 Define and deploy a vRealize Automation Cloud Assembly blueprint that uses IPAM provider range assignment

You can define a blueprint to obtain and manage IP addresses assignments from your external IPAM provider.

7 Using Infloblox-specific properties and extensible attributes for IPAM integrations in vRealize Automation

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

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

Before you can download and deploy the Infoblox provider package (`infoblox.zip`) for integration with vRealize Automation 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 vRealize Automation 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`
- `Tenant ID`
- `CMP Type`
- `VM ID`
- `VM Name`

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.
- Review Using Infloblox-specific properties and extensible attributes for IPAM integrations in vRealize Automation.

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 vRealize Automation 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
  - `Tenant ID` - type String
  - `CMP Type` - type String
  - `VM ID` - type String
  - `VM Name` - 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 Infloblox package as described in Download and deploy an external IPAM provider package for use in vRealize Automation Cloud Assembly.

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

Before you can define an external IPAM integration point in vRealize Automation Cloud Assembly, you need a configured IPAM provider package. You can download a provider package from your IPAM provider's website or from the vRealize Automation Cloud Assembly Marketplace.

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 vRealize Automation Cloud Assembly **Marketplace** tab.

Note 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 vRealize Automation 8.1
- vRA Cloud Infoblox plugin version 1.0 - supports vRealize Automation 8.0.1
- vRA Cloud Infoblox plugin version 0.1 - supports vRealize Automation 8.0

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.

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

Procedure

1. Navigate to the vRA Cloud Infoblox plugin version 0.1 (vRealize Automation 8.0) or vRA Cloud Infoblox plugin version 1.0 (vRealize Automation 8.0.1) package page at the VMware solutions exchange 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.

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 point in vRealize Automation .

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

Before you can define a external IPAM integration point in vRealize Automation, you need to create or access an existing running environment to serve as an intermediary between the IPAM provider and vRealize Automation. 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 vRealize Automation 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 actions-based extensibility (ABX) On-Prem Embedded integration point. The running environment is used to connect to the external IPAM provider, for example Infoblox.

Note An Infoblox IPAM integration point requires an actions-based extensibility (ABX) On-Prem Embedded integration point.

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
- 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
- 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

Prerequisites

- Verify that you have cloud administrator credentials. See Credentials required for working with cloud accounts in vRealize Automation.
- Verify that you have the cloud administrator user role. See What are the vRealize Automation Cloud Assembly 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 vRealize Automation Cloud Assembly Marketplace and then deployed in vRealize Automation 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 Assembly.

Procedure

1. To create an On-Prem FaaS-based extensibility action to use as an IPAM integration running environment, select **Extensibility > Library > Actions.**
2. Click **New Action**, enter an action name and description, and specify a project.
3. In the **FaaS provider** drop-down menu, select **On Prem**.

4 Complete the form to define the extensibility action.

 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 point in vRealize Automation

vRealize Automation supports integration with an external IPAM providers. You can use a provider-specific IPAM integration point to obtain and manage IP addresses and related network characteristics for blueprint 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 vRealize Automation 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 vRealize Automation 8.1
- vRA Cloud Infoblox plugin version 1.0 - supports vRealize Automation 8.0.1
- vRA Cloud Infoblox plugin version 0.1 - supports vRealize Automation 8.0

Prerequisites

- Verify that you have cloud administrator credentials. See Credentials required for working with cloud accounts in vRealize Automation.
- Verify that you have the cloud administrator user role. See What are the vRealize Automation Cloud Assembly 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.

  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 Assembly.
- Verify that you have access to a configured running environment for the IPAM provider. The running environment is typically an actions-based extensibility (ABX) On-Prem Embedded integration point.

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

- Enable required extensible attributes in your Infoblox application. See Add required extensible attributes in the Infoblox application for integration with vRealize Automation.
- If you do not have external Internet access, you can configure an Internet server proxy. See How do I configure an Internet proxy server for vRealize Automation.

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 vRealize Automation 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 deploy an external IPAM provider package for use in vRealize Automation Cloud Assembly.

   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 do I upgrade to a newer IPAM integration package in vRealize Automation .
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 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.

The IPAM framework only supports an actions-based extensibility (ABX) On-Prem Embedded running environment.

Note An Infoblox IPAM integration point requires an actions-based extensibility (ABX) On-Prem Embedded integration point.

The configured cloud account or integration point allows communication between vRealize Automation 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.

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 *Infloblox_ 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 in vRealize Automation Cloud Assembly to use IPAM provider values

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

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

Prerequisites

This sequence of steps is shown in the context of an IPAM provider integration workflow. See Provider-specific external IPAM integration use case.

- Verify that you have cloud administrator credentials. See Credentials required for working with cloud accounts in vRealize Automation.
- Verify that you have the cloud administrator user role. See What are the vRealize Automation Cloud Assembly 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 point in vRealize Automation .

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 Assembly 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 *Infloblox_ 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 *Infloblox_ 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 Assembly, 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 Assembly. 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.

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 blueprint constraint tag to make the provisioning association in the blueprint.
- 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 to support the external IPAM integration and are available for use within a blueprint.

Define and deploy a vRealize Automation Cloud Assembly blueprint that uses IPAM provider range assignment

You can define a blueprint to obtain and manage IP addresses assignments from your external IPAM provider.

In this final step in the external IPAM integration workflow, you define and deploy a blueprint 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 vRealize Automation Cloud Assembly.

The 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.

After you deploy the blueprint 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 Provider-specific external IPAM integration use case.

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

- Verify that you have the cloud administrator user role. See What are the vRealize Automation Cloud Assembly 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 point in vRealize Automation .
- Verify that you have configured a vRealize Automation Cloud Assembly network and network profile that support external IPAM integration for your intended IPAM integration point. See Configure a network and network profile in vRealize Automation Cloud Assembly to use IPAM provider values.
- 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 blueprint, see The WordPress use case. For more information about tagging, see How do I use tags to manage vRealize Automation Cloud Assembly resources and deployments.

  For information about custom naming VMs by using settings in your project, see How do I customize the names of deployed resources using vRealize Automation Cloud Assembly.

Procedure

1 Click **Blueprints > New**, enter the following information in the **New Blueprint** page, and click **Create**.
   - **Name** = ipam-bpa
   - **Description** = Blueprint 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 blueprint canvas and connect the two components.
3 Edit the blueprint 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 blueprint code to specify that the network assignment type is *static*.

   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 *static* assignment setting is required, the *address* value is not. If you want to start external IP address selection at a particular address you can, but it is not required. If you do not specify an *address* value, the external IPAM provider selects the next available address in the external IPAM network.

5 Verify the blueprint code against the following example.

```
formatVersion: 1
inputs: {}
resources:
  Cloud_Network_1:
    type: Cloud.Network
    properties:
      networkType: existing
      name: ipam
      constraints:
        - tag: infoblox_abx
  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}'
```

Note 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.

6 Click **Deploy** on the blueprint page, name the deployment *Infoblox-1*, and click **Deploy** on the **Deployment Type** page.

7 As the blueprint 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 *Infloblox_ 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.

10 To view the new network record in vRealize Automation Cloud Assembly, select **Infrastructure > Resources > Networks** and click to open the **IP Addresses** tab.

11 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 vRealize Automation Cloud Assembly is *Infoblox_ Deallocate*.

Using Infloblox-specific properties and extensible attributes for IPAM integrations in vRealize Automation

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

The following Infloblox properties are available for use with your Infoblox IPAM integrations. You can use them in vRealize Automation to further control IP address allocation during blueprint deployment. Use of these properties is optional.

While all of the following Infloblox properties are available for use with the vRA Cloud Infoblox plugin version 0.1 package for vRealize Automation 8.0, the `Infoblox.IPAM.Network.dnsView` property is only available for use with the vRA Cloud Infoblox plugin version 1.0 for vRealize Automation 8.0.1 and vRA Cloud Infoblox plugin version 1.1 for vRealize Automation 8.1 packages.

Note Use of these properties is not included in the Provider-specific external IPAM integration use case sample workflow.

- `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. Default value: False.

- `Infoblox.IPAM.Network.dnsView`

  This property enables you to use a DNS view when creating a host record inside Infoblox. Default value: default.

- `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. Default value: True.

- `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. Default value: None (empty string).

  `Infoblox.IPAM.Network.dnsSuffix` is only applicable if `Infoblox.IPAM.Network.enableDns` is set to True.

  You can specify an Infoblox property using one of the following methods in vRealize Automation Cloud Assembly:

  - You can specify properties in a project by using the **Custom Properties** section on your **Infrastructure > Configure > 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 blueprint. Sample blueprint 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
```

- You can 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.

Building your vRealize Automation Cloud Assembly resource infrastructure

4

vRealize Automation Cloud Assembly resource infrastructure is where you define cloud account regions as zones into which blueprints 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 vRealize Automation Cloud Assembly target placement regions or data centers
- How to add vRealize Automation Cloud Assembly flavor mappings to create common machine sizes in vRealize Automation Cloud Assembly
- How to add vRealize Automation Cloud Assembly image mapping to create common operating systems
- How to add vRealize Automation Cloud Assembly network profiles
- How to add vRealize Automation Cloud Assembly storage profiles that account for different requirements
- How do I use tags to manage vRealize Automation Cloud Assembly resources and deployments
- How to work with resources in vRealize Automation Cloud Assembly

How to add cloud zones that define vRealize Automation Cloud Assembly target placement regions or data centers

A vRealize Automation 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 blueprints deploy workloads. Each cloud zone is associated with a vRealize Automation Cloud Assembly project.

Select **Infrastructure > Configure > Cloud Zones** and click **Add New Zone**.

Learn more about vRealize Automation Cloud Assembly cloud zones

vRealize Automation Cloud Assembly cloud zones are sections within your cloud account type such as AWS or vSphere. Cloud zones are specific to a project.

Additional placement controls include placement policy options, capability tags, and compute tags.

- Placement policy

  Placement policy drives host selection within the specified region.

  - default - Place compute resources on hosts randomly.
  - binpack - Place compute resources on the most loaded host that has enough available resources to run the given compute.
  - spread - Place compute resources evenly across hosts.
- Capability tags

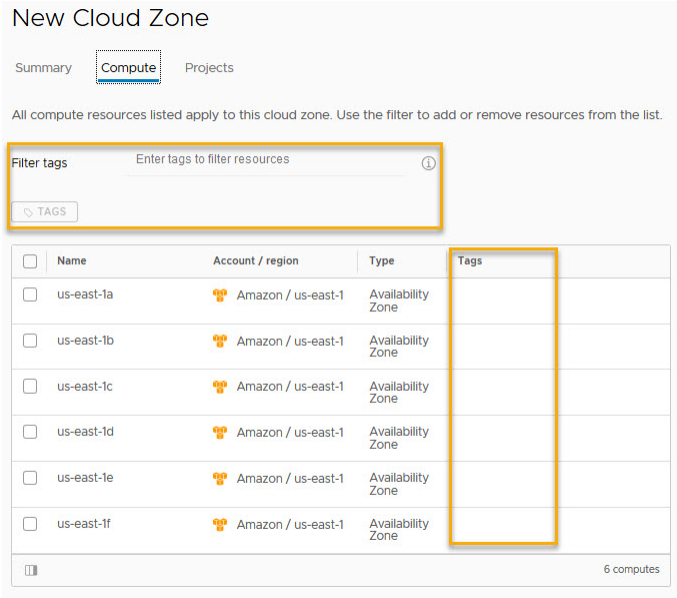
  Blueprints contain constraint tags to help determine deployment placement. During deployment, blueprint constraint tags are mapped to matching capability tags in cloud zones to determine which cloud zones are available for compute resource placement.
- Computes

  You can view and manage the computes that can be provisioned to this cloud zone.

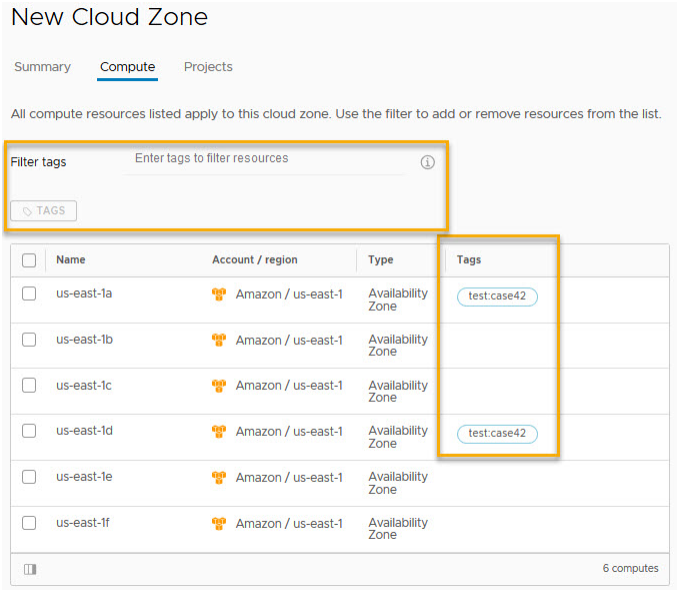
  If a vCenter compute cluster is DRS-enabled, the cloud zone only displays the cluster in the list of computes and it does not display the child hosts. If a vCenter compute cluster is not DRS-enabled, the cloud zone only displays standalone ESXi hosts, if present.

  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.

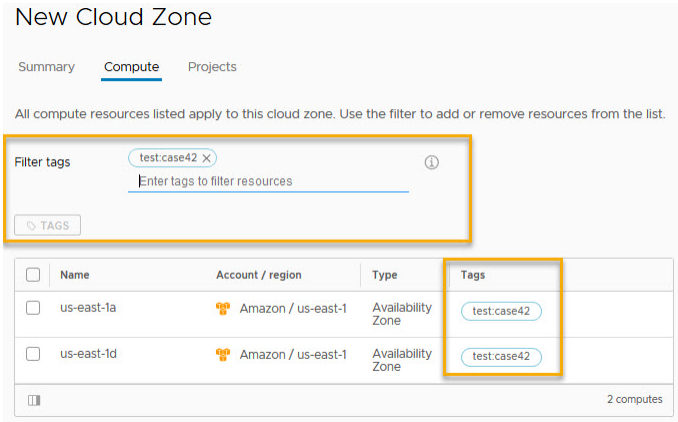
  - Computes contain no tags and no filtering is used.



- 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.

How to add vRealize Automation Cloud Assembly flavor mappings to create common machine sizes in vRealize Automation Cloud Assembly

A vRealize Automation Cloud Assembly 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 Assembly

A flavor mapping groups a set of target deployment sizings for a specific cloud account/region in vRealize Automation Cloud Assembly 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 blueprint, you pick an available flavor that fits your needs.

Organize flavor mappings for your project by deployment intent.

To simplify blueprint 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 blueprints 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 blueprint. If there are flavor mappings defined for a vSphere cloud zone, the flavor mapping serves as a limit for vSphere-specific configurations in the blueprint.

For a basic flavor mapping example, see WordPress use case: add flavor mappings.

How to add vRealize Automation Cloud Assembly image mapping to create common operating systems

A vRealize Automation Cloud Assembly 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

An image mapping groups a set of predefined target operating system specifications for a specific cloud account/region in vRealize Automation 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 vRealize Automation Cloud Assembly deployments.

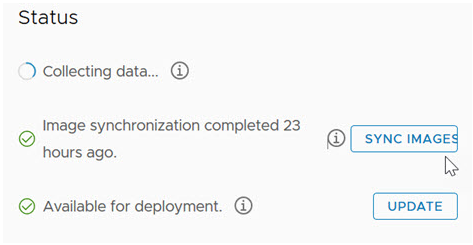
To assign a permission on a content library, an administrator must grant the permission to the user as a global permission. For related information, see *Hierarchical Inheritance of Permissions for Content Libraries* in *vSphere Virtual Machine Administration* at VMware vSphere Documentation.

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.



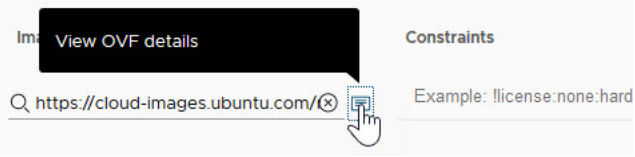
3 When the action is complete, click **Infrastructure > Configure > Image Mapping**. Define a new or edit an existing image mapping and select the cloud account/region from step 1.

4 Click the image synchronization icon on the **Image Mapping** page.

5 Configure image mappings settings for the specified cloud account/region on the **Image Mapping** page.

Viewing OVF details

You can include OVF specifications in vRealize Automation 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.



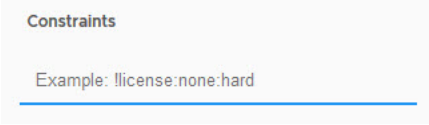
Using shared and latest images from a Microsoft Azure image gallery

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 vRealize Automation 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. vRealize Automation 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_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}
```

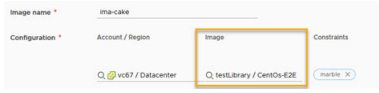
What happens when an image mapping and a cloud template contain a cloud configuration script

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.

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 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, 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, 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 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.

More information about configuring and using cloud configuration scripts

For more information about working with cloud configuration scripts in cloud templates, see How to automatically initialize a machine in a vRealize Automation Cloud Assembly blueprint .

How to add vRealize Automation Cloud Assembly network profiles

A vRealize Automation Cloud Assembly network profile describes the behavior of the network to be deployed.

For example, a network might need to be Internet facing versus internal only. Networks and their profiles are cloud-specific.

Select **Infrastructure > Configure > Network Profiles** and click **New Network Profile**.

Learn more about network profiles in vRealize Automation Cloud Assembly

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 Assembly.

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 Assembly, 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 blueprints 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 blueprint to help control network selection when the blueprint 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 blueprint is deployed.

When using static IP, the address range is managed by vRealize Automation. For DHCP, the IP start and end addresses are managed by the independent DHCP server, not by vRealize Automation. 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 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 blueprint 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.

- **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**

  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 blueprint, 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 blueprint, 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 blueprint that have a `network type: public` property are matched to networks that are flagged as public. Further matching occurs during blueprint deployment to determine network selection.

- **Default for zone**

  Select this option to flag the network as a default for the cloud zone. During blueprint 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 blueprint 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 blueprint, constraint tags in a blueprint'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 blueprint is deployed.

Network Policies

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.

For information about network types, see Using networks and network profiles in vRealize Automation Cloud Assembly.

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. Blueprints 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 blueprints 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 blueprint - 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 size of the on-demand network, using IPv4 notation, to create 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.

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.

- **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.

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 used in a load balancer component in the blueprint, the load balancer is considered during deployment. Load balancers in a matched network profile are used when a blueprint is deployed.

For more information, see Using load balancer settings in network profiles and blueprint designs in vRealize Automation Cloud Assembly and Network, security, and load balancer examples in vRealize Automation Cloud Assembly blueprints.

Security Groups

When a blueprint is deployed, the security groups in its network profile are applied to the machines 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.

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 blueprint, each matching a different network profile, you can use different security groups for different networks.

Adding a tag to an existing security group allows you to use the security group in a blueprint `Cloud.SecurityGroup` component. A security group must have at least one tag or it cannot be used in a blueprint. For more information, see Security resources and Network, security, and load balancer examples in vRealize Automation Cloud Assembly blueprints.

More information about network profiles, networks, blueprints, and tags

For more information about network profiles, see other topics in this section of the help as well as WordPress use case: add network profiles.

For more information about networks, see Network resources.

For examples of sample network component code in a blueprint, see Network, security, and load balancer examples in vRealize Automation Cloud Assembly blueprints.

For sample network automation workflows, see Network Automation with Cloud Assembly and NSX.

For more information about tags and tag strategy, see How do I use tags to manage vRealize Automation Cloud Assembly resources and deployments.

Working with IP addresses in networks and network profiles in vRealize Automation Cloud Assembly

Using network and network profile settings, you can control how network IP addresses are used in vRealize Automation Cloud Assembly blueprints and deployments.

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. For more information, see Learn more about network profiles in vRealize Automation Cloud Assembly.

IPv4 and IPv6 support in vRealize Automation Cloud Assembly networks

vRealize Automation Cloud Assembly 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 addresses for the networks in your blueprints 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 third-party 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 Configure an external IPAM integration point in vRealize Automation . For an example of how to create an IPAM integration point for a specific IPAM vendor, see Provider-specific external IPAM integration use case.

Using networks and network profiles in vRealize Automation Cloud Assembly

You use networks and network profiles in vRealize Automation Cloud Assembly to help define the behavior of network provisioning for your deployments.

In vRealize Automation Cloud Assembly, you can define cloud-specific network profiles. See Learn more about network profiles in vRealize Automation Cloud Assembly.

Network types

A network component in a blueprint is defined as one of the following `networkType` types.

Network type	Definition
`existing`	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.
`public`	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.
`private`	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.
`outbound`	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.
`routed`	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 blueprint specification in a `Cloud.NSX.Network` network component.

For examples of populated blueprints that contain network component data, see Network, security, and load balancer examples in vRealize Automation Cloud Assembly blueprints.

Networking scenarios

You can expect the following behavior when you deploy a blueprint that uses the following network profile configuration.

Table 4-1. Networking Scenarios

Network type or scenario	No network profiles available for cloud zone	Network profiles available for cloud zone
No network	If no network is specified in the blueprint, 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.	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.
Existing network	If the network component in the blueprint contains constraint tags, those constraints are used to filter the list of available networks. Constraint tags in the blueprint's 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.	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.
Public network	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.	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.
Private network	Provisioning fails because private networks require information from a network profile.	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.

Table 4-1. Networking Scenarios (continued)

Network type or scenario	No network profiles available for cloud zone	Network profiles available for cloud zone
Outbound network	Provisioning fails because outbound networks require information from a network profile.	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.
On-demand routed network	Provisioning fails because routed networks require information from a network profile.	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.
Example Wordpress use case with existing or public networks	Provisioning occurs as described for an existing network or public network.	See above descriptions for existing network and public network behavior. See The WordPress use case.
Example Wordpress use case with existing or public networks and private or outbound networks	Provisioning fails because the network requires information from a network profile.	See above descriptions for a private network and an outbound network. See The WordPress use case.
Example Wordpress use case with load balancer	Provisioning fails because a load balancer requires information from a network profile. Provisioning can occur when existing load balancers are present.	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 The WordPress use case.

Using load balancer settings in network profiles and blueprint designs in vRealize Automation Cloud Assembly

You can configure load balancers in the vRealize Automation Cloud Assembly blueprint based on your NSX-V or NSX-T network profile configuration.

Your load balancer options depend on settings in the blueprint's network profile and on settings in the NSX-T or NSX-V source application.

NSX-T networks and load balancer options

NSX-T supports one load balancer service for each Tier-1 logical router. Your load balancer options depend on the network that the load balancer component is associated to in your blueprint.

- On-demand outbound network

  If the load balancer computes are attached to an on-demand outbound network, a load balancer is created for the Tier-1 router of the on-demand network.

- On-demand private network

  The on-demand network VIP must be associated to the source external network's VIP.

  If the load balancer computes are attached to an on-demand private 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 external network.

- 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 small load balancer service is created only if there is no load balancer service attached to the Tier-1 router. If the load balancer service 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.

- On-demand network with new security group

  Because machines are attached to an existing network and isolation is created using a security group, 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.

A load balancer service created for an on-demand network is destroyed when it is no longer in use. Monitors, pools, application profiles and VIPs are destroyed when the load balancer is destroyed. A load balancer service that is created for an external network is not destroyed.

You can specify NSX-T load balancer settings in the blueprint by using the cloud-agnostic Load Balancer component that is available in the blueprint component panel.

To learn more about using a load balancer in an NSX-T network deployment, see VMware blog post vRA Cloud Assembly Load Balancer with NSX-T Deep Dive.

NSX-V networks and load balancer options

You can configure a load balancer as two-arm or as one-arm based on whether you create an isolated network by using a new network subnet or a security group.

You can specify NSX-V load balancer settings in the blueprint by using the Load Balancer component that is available in the NSX section of the blueprint component panel.

The workflow for creating a two-arm load balancer is:

1 Create a service edge.

2 Attach the uplink interface of the service edge to the public network.

3 Attach the downlink interface to the isolated (outbound) network.

4 Allocate a static IP address for the load balancer from the network profile static IP range.

5 Configure the load balancer.

6 Configure the firewall.

7 Configure the default gateway.

The workflow for creating a one-arm load balancer is:

1 Create a service edge.

2 Attach the network selected from reservation as uplink.

3 Allocate a static IP address for the load balancer from the network profile static IP range.

4 Configure the load balancer.

5 Configure the firewall

6 Configure the default gateway.

How to add vRealize Automation Cloud Assembly storage profiles that account for different requirements

A vRealize Automation Cloud Assembly storage profile describes the kind of storage to be deployed.

Storage is usually profiled according to characteristics such as service level or cost, performance, or purpose, such as backup.

Select **Infrastructure > Configure > Storage Profiles** and click **New Storage Profile**.

Learn more about vRealize Automation Cloud Assembly storage profiles

A cloud account region contains storage profiles that let the cloud administrator define storage for the region.

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, visualize 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 hard *fast* 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.

How do I use tags to manage vRealize Automation Cloud Assembly resources and deployments

Tags are a critical component of vRealize Automation 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 vRealize Automation Cloud Assembly.

Fundamentally, tags are labels that you add to vRealize Automation 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 vRealize Automation Cloud Assembly uses resources and infrastructure to build deployable services. Tags also support governance within Cloud Assembly.

Tag Structure

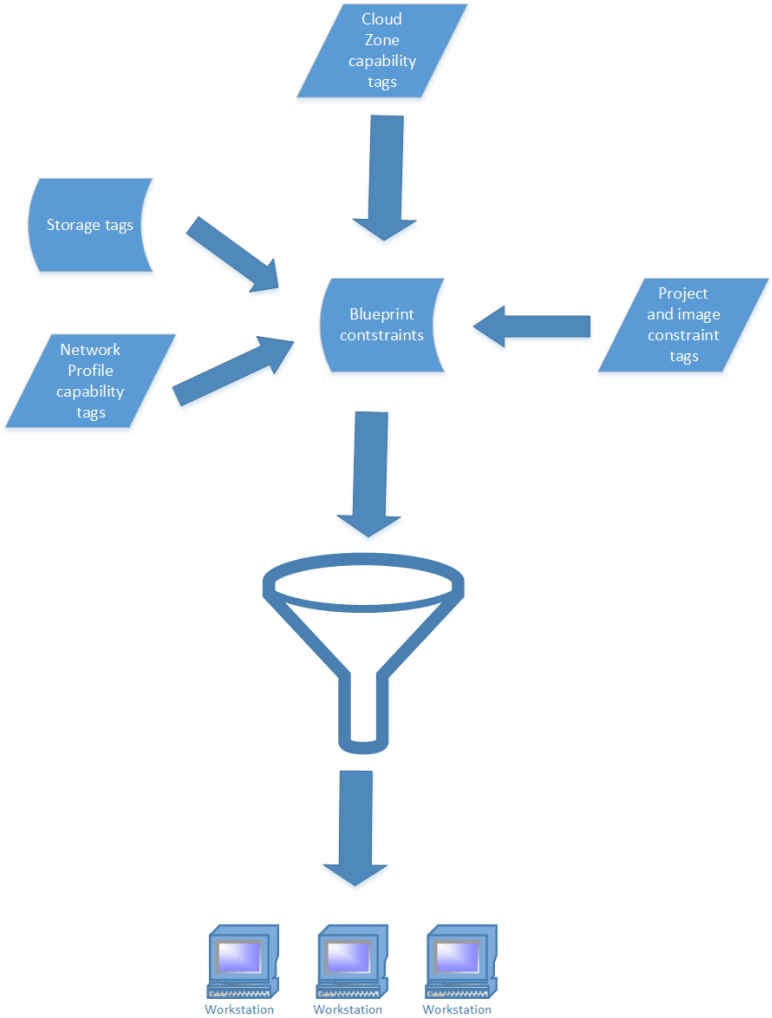
Structurally, tags must follow the `name:value` pair convention, but otherwise their construction is largely free form. Throughout vRealize Automation Cloud Assembly, all tags appear the same, and tag functionality is determined by context.

For example, tags on infrastructure resources function primarily as capability tags because vRealize Automation Cloud Assembly uses them to match resources with deployments. Secondarily, they also identify the resources.

Tag Function

The primary function of tags within vRealize Automation Cloud Assembly is to configure deployments using capabilities and constraints. Capability tags placed on cloud zones, network and storage profiles, and individual infrastructure resources define desired capabilities for deployments. Constraint tags that cloud administrators place on projects enable them to exercise a form of governance over those projects. These constraint tags are added to other constraints expressed in blueprints.

During provisioning, vRealize Automation Cloud Assembly matches these capabilities with constraints, also expressed as tags, in blueprints to define deployment configuration. This tag-based capability and constraint functionality serves as the foundation for deployment configuration in vRealize Automation 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.

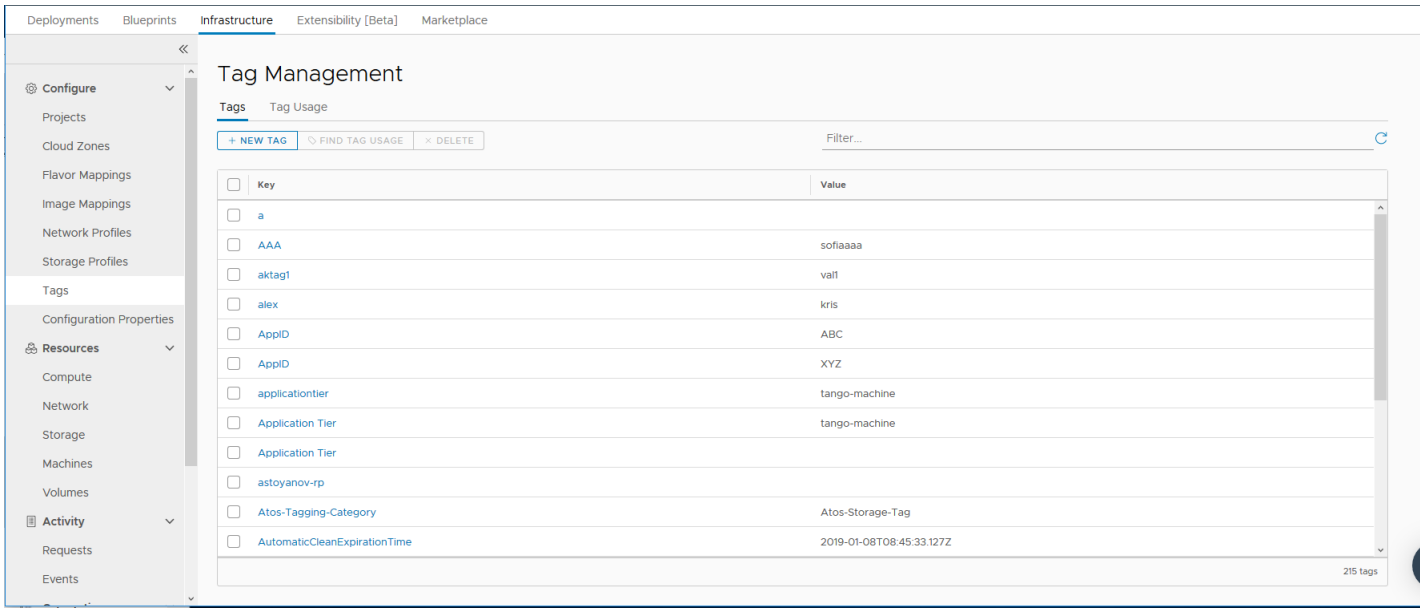
Also, the Manage Tags page and resource configuration pages contain search functions that enable you to locate items by tag names. Using logical and human readable tags for these items is key to facilitating this search and identification function.

External Tags

vRealize Automation Cloud Assembly might also contain external tags. These tags are imported automatically from cloud accounts that you associate with a vRealize Automation 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 Manage Tags page in vRealize Automation Cloud Assembly to monitor and manage your tags library. You can also create tags on this page. In addition, the Manage Tags page is the only page on which you can view and identify external tags.



Tag Strategy

To minimize confusion, before creating tags in vRealize Automation 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 organizations needs, structure, and goals.

Best Practices for Tagging

Some general characteristics of an effective tag strategy:

- Design and implement a coherent plan for tagging that relates to the structure of your business and communicate this plan to all applicable users. A plan 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.

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 blueprint 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 blueprint constraint tags. In general, your administrator will create capability tags for cloud zones and network profiles first, and then other users can design blueprints 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 vRealize Automation Cloud Assembly

In vRealize Automation Cloud Assembly, capability tags enable you to define placement logic for deployment of infrastructure components. They are a more powerful and succinct option to hard coding such placements.

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 Manage Tags page in vRealize Automation 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 things like 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.

vRealize Automation Cloud Assembly matches capability tags with constraints from cloud zones and on blueprints at deployment time. So, when creating and using capability tags, you must understand and plan to create appropriate blueprint constraints so that matching will occur as expected.

For example, in the Add Cloud Zones topic in the Wordpress example, you created dev and test tags for the OurCo-AWS-US-East and OurCo AWS-US-West zones. This indicates that the OurCo-AWS-US-East zone is a development environment, and the OurCo-AWS-US_West zone is a test environment. Paired with the appropriate constraint tags, these capability tags enable you to direct deployments to the desired environments.

Using constraint tags in vRealize Automation Cloud Assembly

You add constraint tags to blueprints and various other components within vRealize Automation Cloud Assembly to match capabilities defined on resources, cloud zones, and profiles to generate appropriate deployments.

There are two main areas in vRealize Automation Cloud Assembly where constraint tags are applicable. The first is on the configuration side in projects and images. The second is on the deployment side in blueprints. Constraints applied in both areas are merged in blueprints to form a set of deployment requirements.

How Constraint Tags Work on Projects

When configuring cloud assembly, cloud administrators can apply constraint tags on projects and image maps. In this way, cloud administrators can apply governance constraints directly at the project level. All constraints added at this level are applied to every blueprint requested for the applicable project.

If tags on the project conflict with tags on the blueprint, 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 blueprint, 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 Blueprints

In blueprints, 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 in the blueprint YAML code. Constraint tags from projects are added to the constraint tags created in blueprints.

vRealize Automation Cloud Assembly supports a simple string formatting to make using constraints easier in YAML files:

```
[!]tag_key[:tag_value][:hard|:soft]
```

By default vRealize Automation 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.

```
name: "wordPressWithMySql"
components:
  mysql:
    type: "Compute"
    data:
      name: "mysql"
      # ... skipped lines ...
  wordpress:
    type: "Compute"
    data:
      name: "wordpress"
      instanceType: small
      imageType: "ubuntu-server-1604"
      constraints:
        - tag: "!location:eu:hard"
        - tag: "location:us:soft"
```

```
        - tag: "!pci"
      # ... skipped lines ...
```

For more information about how to work with blueprints, see WordPress use case: create and expand a blueprint.

How hard and soft constraints work in projects and blueprints

Constraints in both projects and blueprints 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 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.

Note In projects and blueprints, the `failOnConstraintMergeConflict` flag modifies the behavior of constraints. When this flag is set to true, if there is a conflict between project constraints and blueprint constraints, the request will fail. If the flag is not present or set to false, the project constraints take precedence over the blueprint constraints.

Standard tags

vRealize Automation Cloud Assembly applies standard tags to some deployments to support analysis, monitoring, and grouping of deployed resources.

Standard tags are unique within vRealize Automation 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 4-2. Standard tags

Description	Tag
Organization	org:`orgID`
Project	project:`projectID`
Requester	requester:`username`
Deployment	deployment:`deploymentID`

Table 4-2. Standard tags (continued)

Description	Tag
Blueprint reference (if applicable)	blueprint:`blueprintID`
Component name in blueprint	blueprintResourceName:`CloudMachine_1`
Placement Constraints: applied in blueprint, request parameters, or via IT policy	constraints:`key:value:soft`
Cloud Account	cloudAccount:`accountID`
Zone or profile, if applicable	zone:`zoneID`, networkProfile:`profileID`, storageProfile:`profileID`

How vRealize Automation Cloud Assembly Processes Tags

In vRealize Automation Cloud Assembly, tags express capabilities and constraints that determine how and where resources are allocated to provisioned deployments during the provisioning process.

vRealize Automation Cloud Assembly uses a specific order and hierarchy 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 of capability and constraint tag processing:

- 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 blueprints.

How do I set up a simple tagging structure

This topic describes a basic approach and options for a logical vRealize Automation 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 vRealize Automation 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 The WordPress use case 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 blueprint 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 vRealize Automation 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 WordPress use case: add cloud zonesWordPress use case: add network profiles 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 vRealize Automation 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 WordPress use case: 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 Assembly

A cloud administrator can review vRealize Automation Cloud Assembly resources that are exposed through data collection. The cloud administrator can label resources with capability tags to affect where vRealize Automation Cloud Assembly blueprints are deployed.

Compute resources

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 Assembly provisioning.

Network resources

In vRealize Automation Cloud Assembly, cloud administrators can view and edit the network resources that have been data-collected from the project's cloud accounts and integrations.

After you add a cloud account, data collection discovers the cloud account's network and security information and makes that information available for use in network profiles and other options.

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.

The vRealize Automation 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-V, 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 Assembly.

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 define new, and manage existing, IP ranges within your available networks.

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 blueprints. 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 blueprint after the blueprint has been matched with one or more network profiles.

IP Ranges

Select an IP range to define or make changes to the start and end IP address for a particular network in your organization.

The default gateway cannot be included 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 add an IPAM IP range. This process is described within the context of an overall external IPAM integration workflow at Configure a network and network profile in vRealize Automation Cloud Assembly to use IPAM provider values.

IP Addresses

Display the status, for example available or allocated, of defined IP addresses that are used by your organization.

Load Balancers

Display 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.

Security resources

After you add a cloud account, 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

You can view on-demand security groups that have been created in vRealize Automation Cloud Assembly and existing security groups that were created in source applications, such as NSX-T and Amazon Web Services that are exposed through data collection.

You can view the available security groups and add or remove tags for selected security groups. You can also edit on-demand security groups that you created. Cloud administrators can open and edit security groups. A blueprint author can assign one or more security groups to a machine NIC to control network rules and other aspects of security for the blueprint deployment.

When using firewall rules in the blueprint code of a security group component,

Existing security groups from the underlying cloud account endpoint, such as NSX-V, NSX-T, or Amazon Web Services applications, are data-collected by vRealize Automation Cloud Assembly. Existing security groups are displayed and classified in the **Origin** column as `Discovered`. On-demand security groups that you create in vRealize Automation Cloud Assembly, either in a blueprint or in a network profile, are displayed and classified in the **Origin** column as `Managed by Cloud Assembly`. Only security groups that are available for use in your organization are visible.

If you edit an existing security group directly in the source application, such an in the source NSX application rather than in vRealize Automation Cloud Assembly, the updates are not visible in vRealize Automation Cloud Assembly until you run data collection on the associated cloud account or integration point from within vRealize Automation Cloud Assembly.

A cloud administrator can assign one or more tags to an existing security group to allow it to be used in a blueprint. A security group must have at least one tag or it cannot be used in a blueprint. A blueprint author can use a `Cloud.SecurityGroup` component in a blueprint to allocate an existing security group by using tag constraints. In the blueprint, a `Cloud.SecurityGroup` security group component must be applied to a machine NIC to control network rules and other aspects of security in the blueprint deployment.

For information about how an administrator can add tags to an existing security group, or how tags are data-collected from the security group in the source application, see Learn more about network profiles in vRealize Automation Cloud Assembly.

For examples of how a blueprint author can use tags in a blueprint security component, see Network, security, and load balancer examples in vRealize Automation Cloud Assembly blueprints.

For a blueprint code sample, see the *Existing security group with a constraint tag applied to a machine NIC* example at Network, security, and load balancer examples in vRealize Automation Cloud Assembly blueprints.

For more information about using security groups in network profiles, see Learn more about network profiles in vRealize Automation Cloud Assembly.

Ways to define security groups in network profiles and blueprints

You can leverage security group capabilities in any of the following ways:

- Existing security group specified in a network profile

  You can add an existing security group to a network profile. When a blueprint 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 component to a blueprint.

- Security group component associated to machine component in a blueprint

  You can drag and drop a security group component on to a blueprint and bind the security group component to a machine NIC by using constraint tags on the existing security group component in the blueprint and on the existing security group in the data-collected resource.

- NSX-T tags specified in your NSX-T application

- You can use an NSX-T tag, specified as a constraint on a network component in a blueprint, where the network component is connected to a machine NIC in the blueprint. 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.

Storage resources

A cloud administrator can work with storage resources and their capabilities, which are discovered through vRealize Automation Cloud Assembly 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 vRealize Automation Cloud Assembly. The additional tags might label a specific capability for matching purposes at provisioning time.

Capabilities on storage resources become visible as part of the definition of a vRealize Automation Cloud Assembly storage profile. See Learn more about vRealize Automation Cloud Assembly storage profiles .

Machine resources

In vRealize Automation, all users can review machine resources that are exposed through data collection.

All the machines in your projects appear in the Machines list.

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.

You can use a workload onboarding plan to bring unmanaged machines into vRealize Automation management.

For information about using onboarding plans to bring unmanaged machines into vRealize Automation management, see What are onboarding plans in vRealize Automation Cloud Assembly.

Volume resources

In vRealize Automation Cloud Assembly, all users can review volume resources.

vRealize Automation 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 vRealize Automation 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 vRealize Automation Cloud Assembly itself.

Learn more about resources in vRealize Automation Cloud Assembly

vRealize Automation Cloud Assembly can expose additional information around data-collected resources, such as cost.

How does data collection work in vRealize AutomationvRealize Automation Cloud Assembly

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.

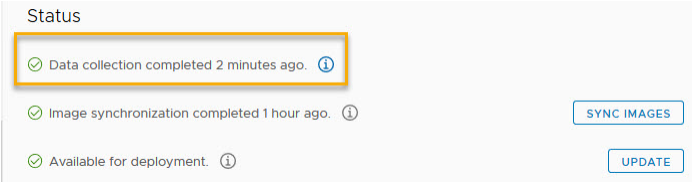
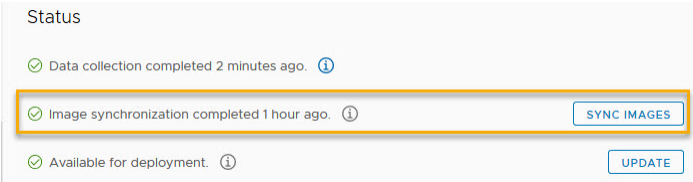


Image data collection

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.



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 Assembly.

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 vRealize Automation Cloud Assembly, you can use an onboarding plan to allow vRealize Automation Cloud Assembly to manage the machine deployments.

For information about adding cloud accounts, see Adding cloud accounts to vRealize Automation Cloud Assembly.

For information about onboarding unmanaged machines, see What are onboarding plans in vRealize Automation Cloud Assembly.

What is the cost of my vRealize Automation Cloud Assembly deployment

Project administrators and cloud administrators are often tasked with managing costs. vRealize Automation Cloud Assembly cost management helps you understand the monetary impact of your individual deployments to help you manage resources.

Before you can view costing, you must configure and enable Cost in vRealize Operations to work with vRealize Automation . When configuring vRealize Operations with vRealize Automation , ensure that both applications are set to the same timezone. To configure the timezone in vRealize Operations, enable SSH and log on to each vRealize Operations node, edit the `$ALIVE_Base/user/conf/analytics/advanced.properties` file, and add `timeZoneUseInMeteringCalculation = <time zone>`.

The cost of a deployment over time appears on the deployment card as the month to date cost, 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 might need to understand the aggregate costs for an entire project, for similar business reasons. Costs at the project level reflect the full cost of all the deployments for the project team. To view the full project cost, select **Infrastructure > Projects > Select Project > Cost**. The results show the aggregate cost and the cost breakdown by deployment.

Note Project costs only include the costs for private cloud workloads. If a project contains deployments that belong to public clouds, the costs for these deployments are not included in the project cost.

How are the costs calculated

The initial costs 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 cost rate is applied to hosts and the service calculates the CPU and memory rates. The server recalculates the cost every 24-hours.

You can also manually refresh the cost server at any time on the vROPs Endpoint page, **Infrastructure > Integrations > vROPs Endpoint >** . In the vCenter servers section, click **Sync**. When manually refreshing the cost server using the **Sync** option, the cost is recalculated for all projects in the organization. Depending on how many projects your organization has this process may be intensive and take time.

For a list of supported resources, see List of costed component types in vRealize Automation Cloud Assembly.

How can I customize the calculated rates

After working with the benchmark values for a period of time, a cloud administrator might find that your actual costs are discounted. You can adjust the rate that is used to calculate the cost so that it better reflects your business practices.

If you make adjustments, the calculation changes are reflected the next time the service runs the calculations. The server recalculates the values every 24-hours.

List of costed component types in vRealize Automation Cloud Assembly

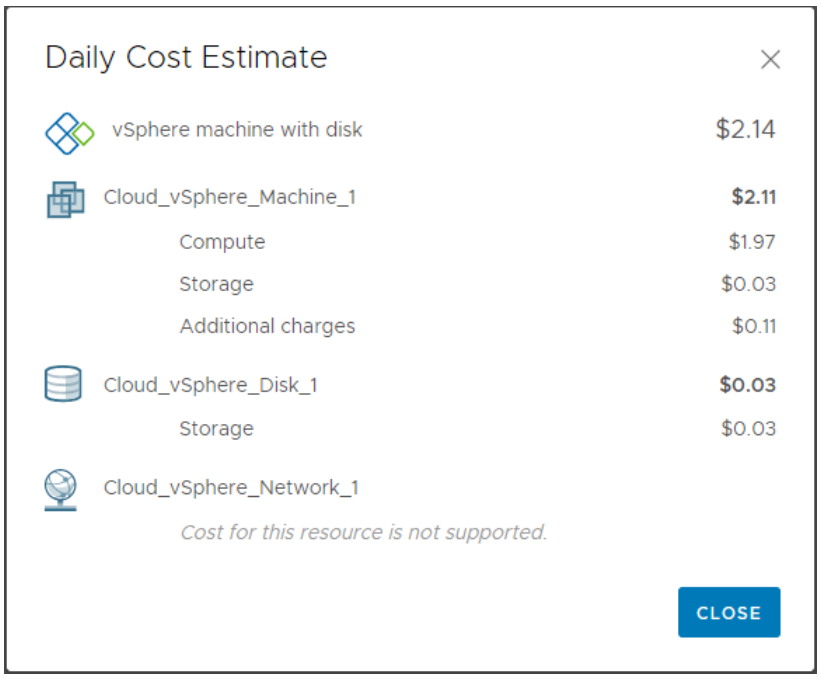
vRealize Automation Cloud Assembly provides benchmark cost information for the following blueprint component types.

Table 4-3. Costed Component Types

Blueprint Component Type	Service Name/Object Type	Blueprint Resource Type	Comments
Cloud Agnostic	Machine	Cloud.Machine	If an agnostic machine is configured with vSphere, you can view deployment cost.
	Disk	Cloud.Volume	If an agnostic disk is attached to a virtual machine that is configured with vSphere, you can view deployment cost.
vSphere	vSphere machine	Cloud.vSphere.Machine	Deployed using a cloud specific blueprint.
	vSphere disk	Cloud.vSphere.Disk	Deployed using a cloud specific blueprint attached to a virtual machine.

How do I estimate the cost of a deployment

Before deploying a catalog item, you can use the upfront cost as a cost estimate for your deployment.



The upfront cost of a deployment is a daily cost 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 cost as an aggregate of the upfront cost on the **Deployment** and **Infrastructure > Projects** tabs. Upfront costing 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.

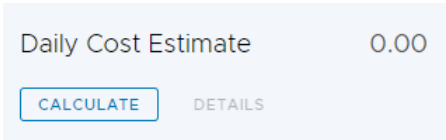
Note Upfront costing is not supported for public cloud resources, or non-vSphere Machine or Disk private cloud resources.

Prerequisites

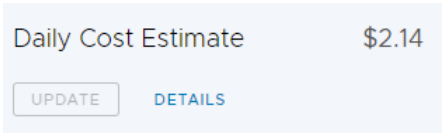
To view the upfront cost in vRealize Automation Cloud Assembly, you must have a vRealize Operations integration endpoint configured with costing enabled and currency preset.

Procedure

1 From the Catalog, select a catalog item and click **Request**.



2 Enter the details for your catalog item request and click **Calculate**.



3 (Optional) Click **Details** to view the cost breakdown in the Daily Cost Estimate window.

What to do next

If the daily cost estimate is acceptable, click **Submit** to continue the deployment request.

How do I determine why a deployment appears to be expensive

While managing costs, project administrators and cloud administrators might find that a deployment is expensive.

A deployment's cost is calculated by it's resources. Occasionally, the resources in a single deployment combined can increase the cost, making the deployment expensive for your business needs. Consequently, if your project contains one or more expensive deployments the aggregrate project cost is also increased.

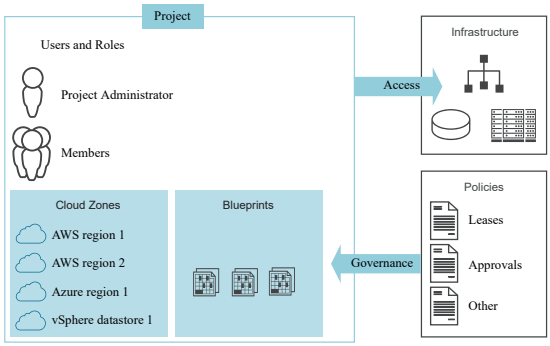
Determining if your deployment is expensive is unique to your own business needs. You can view the cost breakdown of your deployment by opening a deployment's details and clicking **Cost**. To modify the cost of your deployment, you must edit the resource makeup of it's blueprint and redeploy.

Adding and managing vRealize Automation Cloud Assembly projects

5

Projects control who has access to vRealize Automation Cloud Assembly blueprints and where the blueprints are deployed. You use projects to organize and govern what your users can do and to what cloud zones they can deploy blueprints in your cloud infrastructure.

Cloud administrators set up the projects, to which they can add users and cloud zones. Anyone who creates and deploys blueprints must be a member of at least one project.



This chapter includes the following topics:

- How do I add a project for my vRealize Automation Cloud Assembly development team
- Learn more about vRealize Automation Cloud Assembly projects

How do I add a project for my vRealize Automation Cloud Assembly development team

You create a project to which you add members and cloud zones so that the project members can deploy their blueprints to the associated zones. As the vRealize Automation 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 blueprint, you first select the project to associate it with. The project must exist before you can create the blueprint.

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 blueprint, see The WordPress use case.

This procedure is based on creating an initial project that includes only the basic configurations. As your development team creates and deploys their blueprints, 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 vRealize Automation Cloud Assembly projects.

Prerequisites

- Verify that you configured the cloud zones. See Chapter 4 Building your vRealize Automation 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 vRealize Automation Cloud Assembly resource infrastructure.
- Verify that you have the necessary permissions to perform this task. See What are the vRealize Automation Cloud Assembly user roles.
- Determine who you are designating as the project administrator. To understand what the project administrator can do in vRealize Automation Cloud Assembly, see What are the vRealize Automation Cloud Assembly user roles.
- If you are adding Active Directory groups to projects, verify that you configured Active Directory groups for your organization. See Edit group role assignments in vRealize Automation in *Administering vRealize Automation*. If the groups are not synchronized, they are not available when you try to add them to a project.

Procedure

1. Select **Infrastructure > Configure > Projects**, and click **New Project**.
2. Enter the project name.
3. Click the **User** tab.
   a. To make deployments by project members accessible only to the owner, turn off **Deployment sharing**.
   b. Add users with assigned roles.
4. Click the **Provisioning** tab and add one or more cloud zones.

   The cloud zones must contain the resources that support the blueprints deployed by the users.
5. Click **Create**.
6. 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 blueprints. See Chapter 6 Designing your vRealize Automation Cloud Assembly deployments.

Learn more about vRealize Automation Cloud Assembly projects

Projects are the connector between blueprints and resources. The more you understand about how they work and how you can make them work for you, the more effective your vRealize Automation Cloud Assembly development and deployment process will be.

Using vRealize Automation 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 vRealize Automation Cloud Assembly blueprints. 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, 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 blueprint must be provisioned on resources with the matching capability tag. The deployment process fails when no matching tag is found. For example, a blueprint 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 blueprints 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 blueprint 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 blueprint constraint tags and override them at deployment time. If you have a blueprint where this must never happen, you can use the `failOnConstraintMergeConflict:true` in the blueprint. For example, if your project has a network `loc:london` constraint, but the blueprint 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.

```
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 blueprint 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.

A blueprint 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.

How do vRealize Automation 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 blueprint 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 blueprint components are deployed. This understanding helps you create projects that support blueprint developers and to troubleshoot failed deployments.

When you create a blueprint, you first associate it with a project. At deployment time, the blueprint requirements are evaluated against the project cloud zones to find the best deployment location.

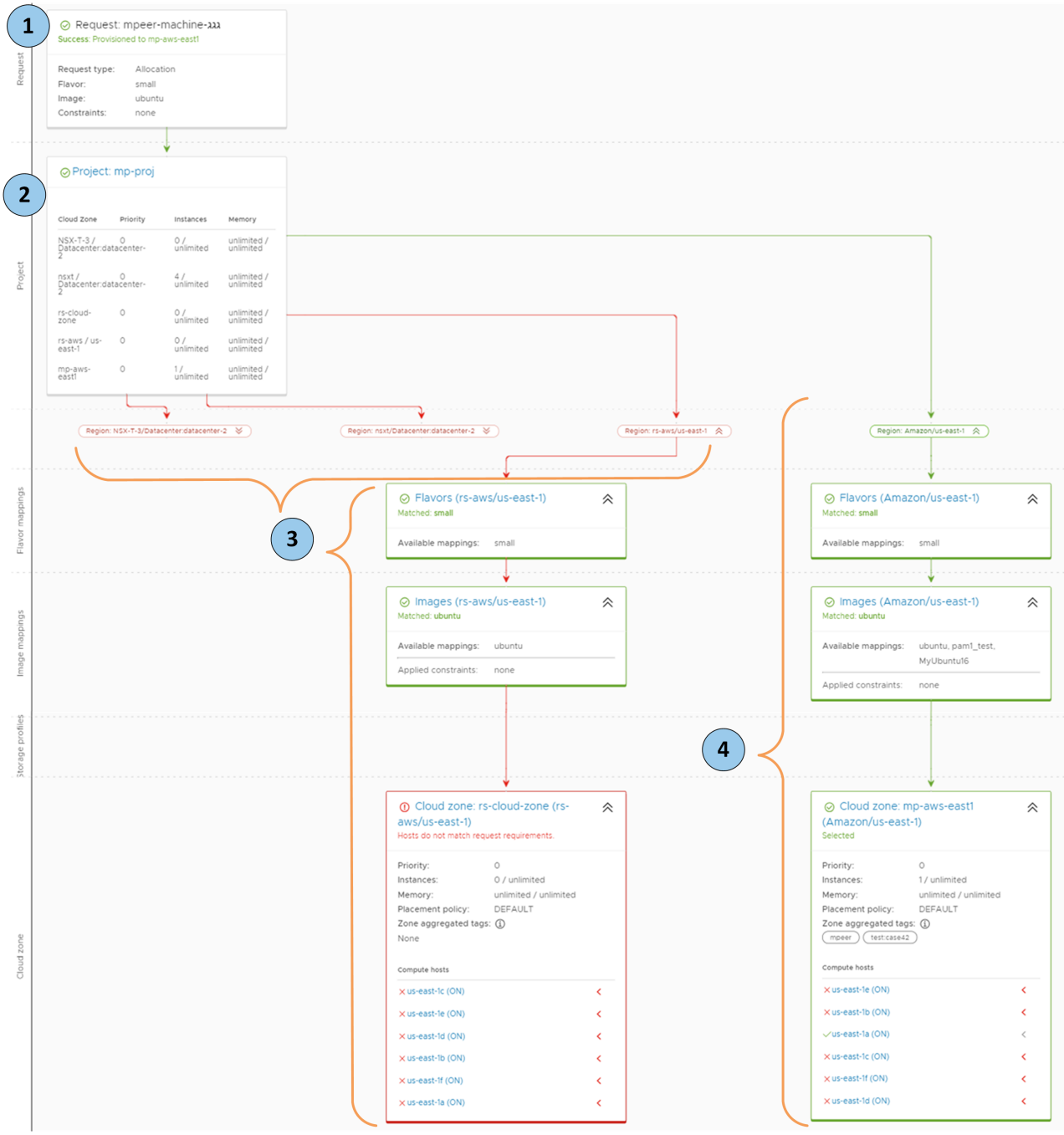
The following workflow illustrates the process.

1 You submit a blueprint deployment request.

2 The project evaluates the blueprint 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 blueprint is deployed to this cloud zone account region.



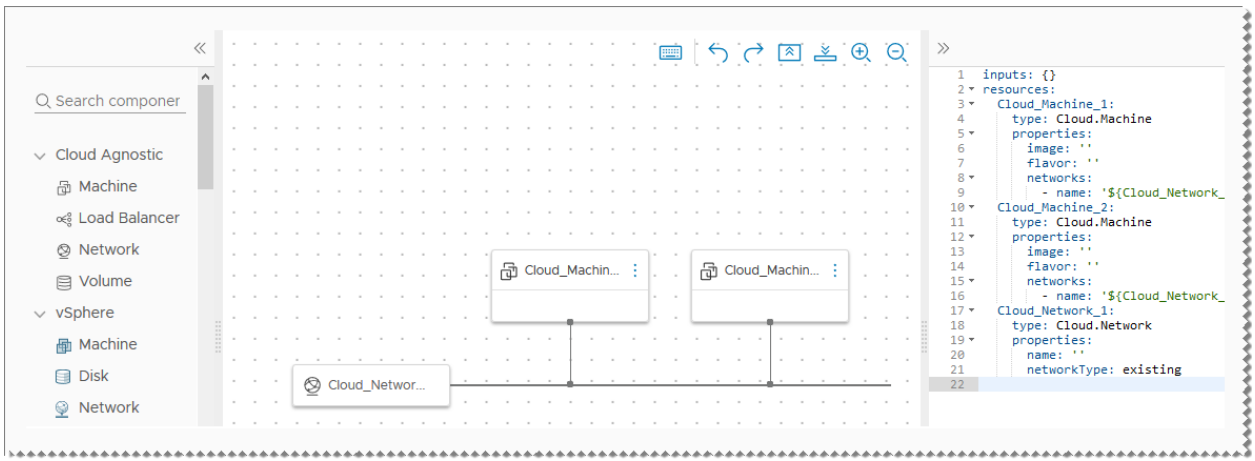
Designing your vRealize Automation Cloud Assembly deployments

6

Deployments begin with blueprints, the specifications that define the machines, applications, and services that you create on cloud resources by way of vRealize Automation Cloud Assembly.

As a blueprint developer, you can design blueprints that target specific cloud vendors, or make them cloud agnostic. The cloud zones that are assigned to your project determine which approach you might take. Check with your cloud administrator to make sure that you understand what kind of resources make up your cloud zones.

Be aware that vRealize Automation Cloud Assembly blueprint creation is an infrastructure-as-code process. You add and connect components in 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 code directly or enter property values into a form.



This chapter includes the following topics:

- Before you create a blueprint
- Ways to create blueprints
- How to create a simple vRealize Automation Cloud Assembly blueprint from scratch
- How to enhance a simple vRealize Automation Cloud Assembly blueprint
- How to save different versions of a vRealize Automation Cloud Assembly blueprint

- How do I customize the names of deployed resources using vRealize Automation Cloud Assembly
- What are the vRealize Automation resource properties
- What are some blueprint code examples
- How to use the vRealize Automation Cloud Assembly Marketplace
- How to extend and automate application life cycles with extensibility

Before you create a blueprint

You can create a vRealize Automation Cloud Assembly blueprint at any time, but to deploy it you first need to define your cloud resource infrastructure.

- Chapter 4 Building your vRealize Automation Cloud Assembly resource infrastructure

In addition, you must create a vRealize Automation Cloud Assembly project that includes those infrastructure resources as cloud zones.

- Using vRealize Automation Cloud Assembly project tags and custom properties

Ways to create blueprints

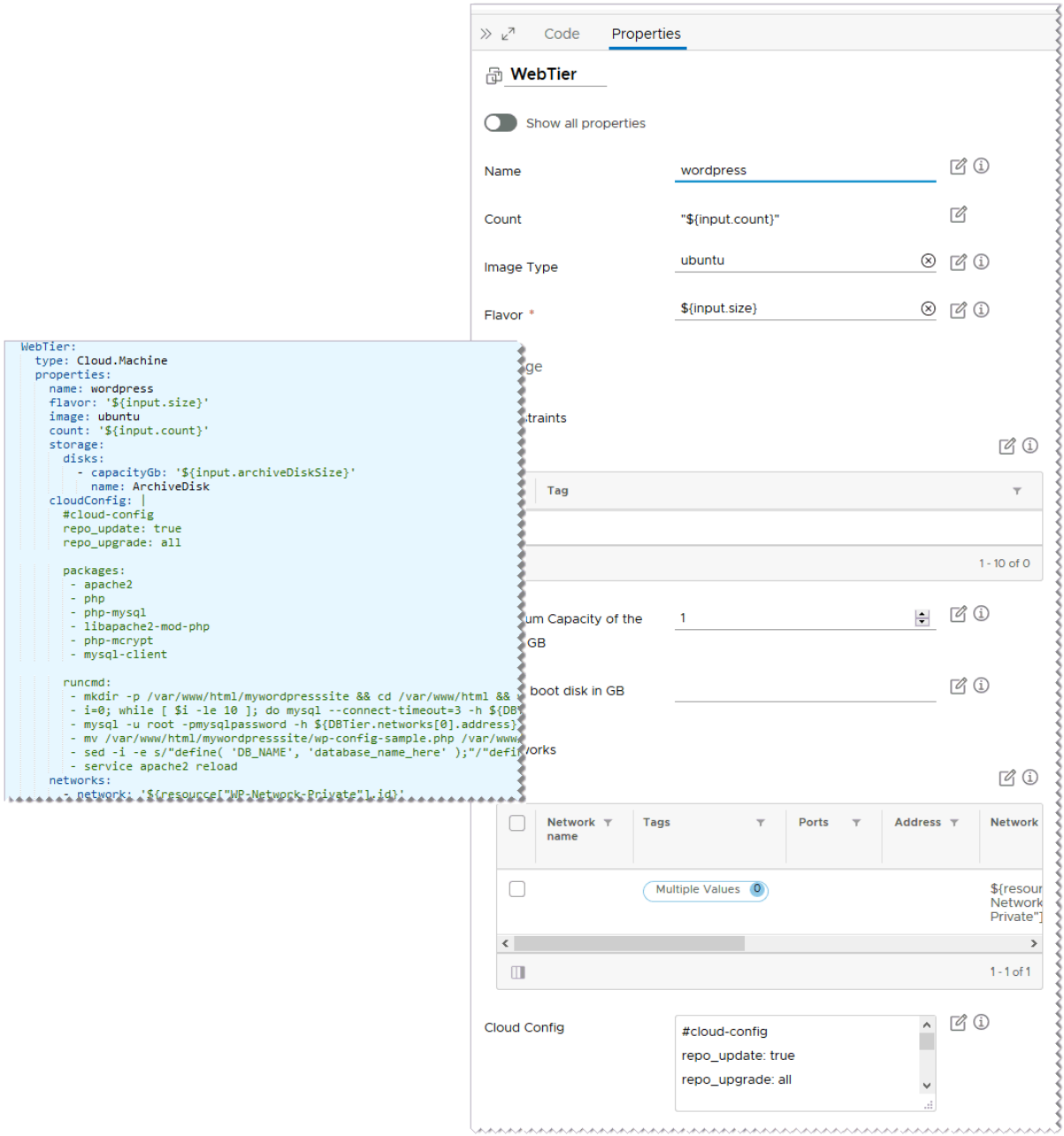
vRealize Automation Cloud Assembly creates and saves blueprints as code, which allows you to easily design and reuse blueprints.

You can build a blueprint from a blank canvas or take advantage of existing code.

The vRealize Automation Cloud Assembly blueprint design page

To create a blueprint from scratch, go to **Blueprints** and click **New**. Drag components to the canvas, connect them, and finish configuring them in the code editor.

The code editor allows you to type, cut, copy, and paste code directly. If you're uncomfortable editing code, you can select a resource 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 blueprint to another.

Blueprint cloning

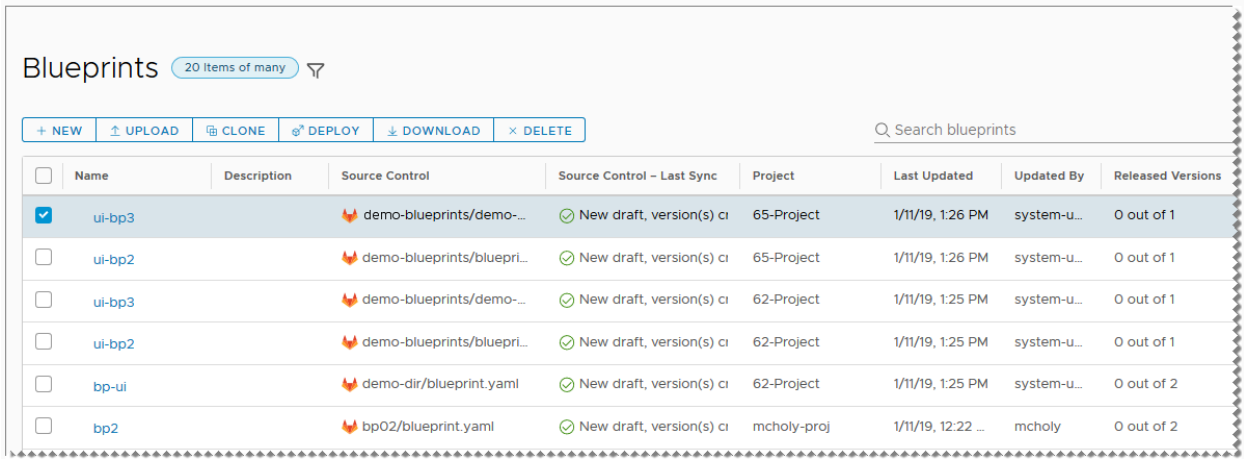
To clone a blueprint, go to **Blueprints**, select a source, and click **Clone**. You clone a blueprint to create a copy based on the source, then assign the clone to a new project or use it as starter code for a new application.

Uploading and downloading

The vRealize Automation Cloud Assembly Marketplace offers finished blueprints to jumpstart your effort. See How to use the vRealize Automation Cloud Assembly Marketplace .

In addition, you can upload, download, and share blueprint YAML code in any way that makes sense for your site. You can even modify blueprint code using external editors and development environments.

Note A good way to validate shared blueprint code is to inspect it in the vRealize Automation Cloud Assembly code editor on the blueprint design page.



How to create a simple vRealize Automation Cloud Assembly blueprint from scratch

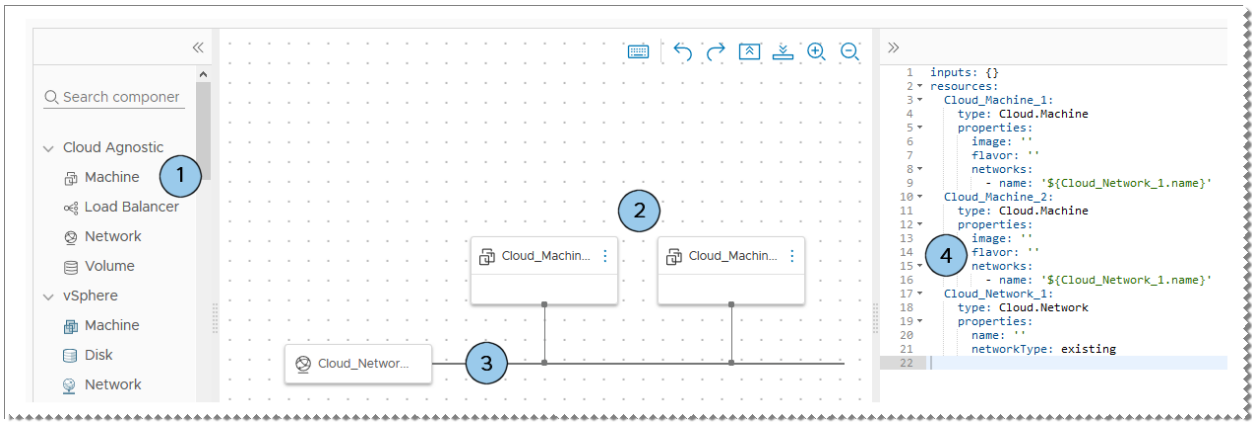
You use the design page to create vRealize Automation Cloud Assembly blueprint specifications for the machines or applications that you want to provision.

1 Locate components.

2 Drag components to the canvas.

3 Connect components.

4 Configure components by editing the blueprint code.



From the design page, you can also change the blueprint name, version or revert to versions, clone, or deploy a blueprint.

How to select and add vRealize Automation Cloud Assembly components to a blueprint

vRealize Automation Cloud Assembly components are your blueprint building blocks. The design page lets you use cloud agnostic components, or components specific to a cloud vendor.

Components appear for selection on the left side of the design page.

Cloud Agnostic Components

You can deploy cloud agnostic components to any cloud vendor. At provisioning time, the deployment uses cloud specific resources that match. For example, if you expect a blueprint to deploy to both AWS and vSphere cloud zones, use cloud agnostic components.

Cloud Vendor Components

Amazon Web Services, Microsoft Azure, and VMware vSphere components can only be deployed to matching AWS, Azure, or vSphere cloud zones.

You can add cloud agnostic components to a blueprint that contains cloud specific components for a particular vendor. Just be aware of what the project cloud zones support in terms of vendor.

Configuration Management Components

Configuration management components depend on your integrated applications. For example, a Puppet component can monitor and enforce the configuration of the other components.

How to connect blueprint resources in vRealize Automation Cloud Assembly

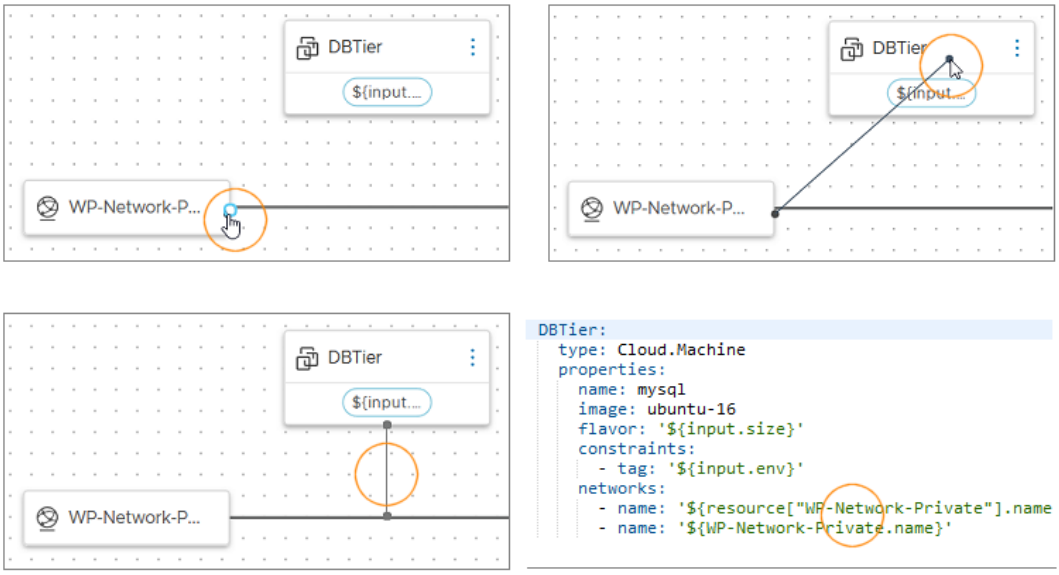
Use the graphical design canvas to connect vRealize Automation Cloud Assembly blueprint resources.

You can connect resources when they are 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.

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.

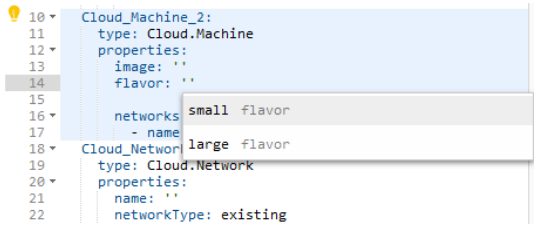
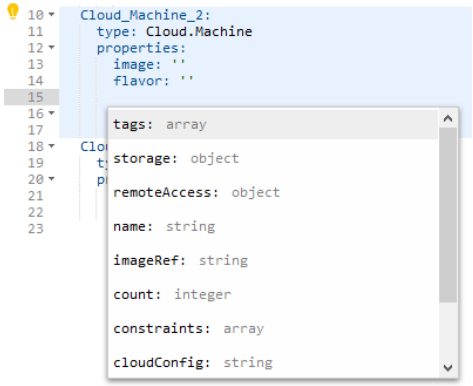
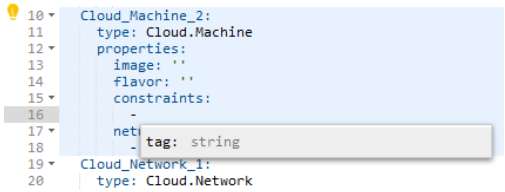
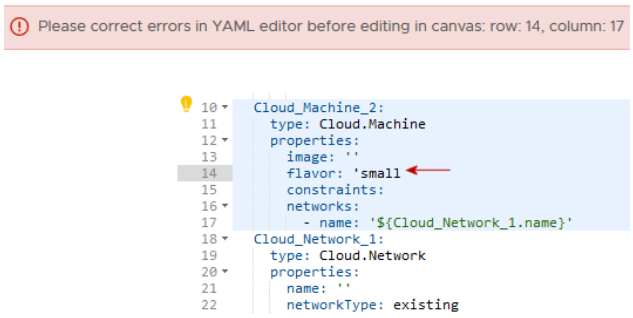
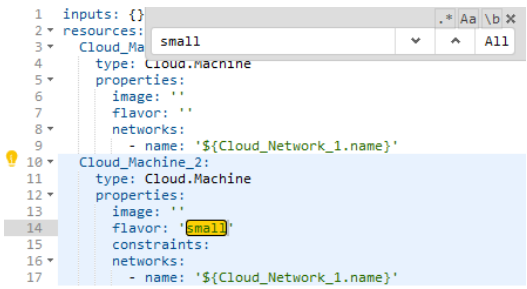


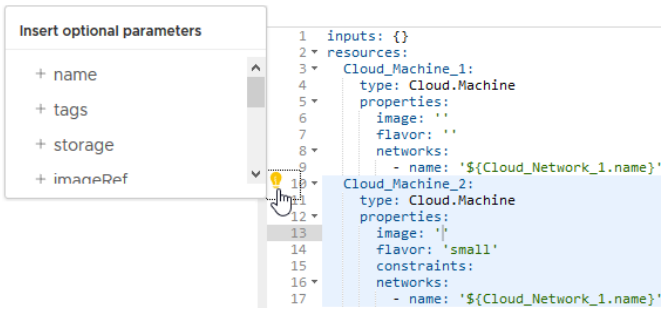
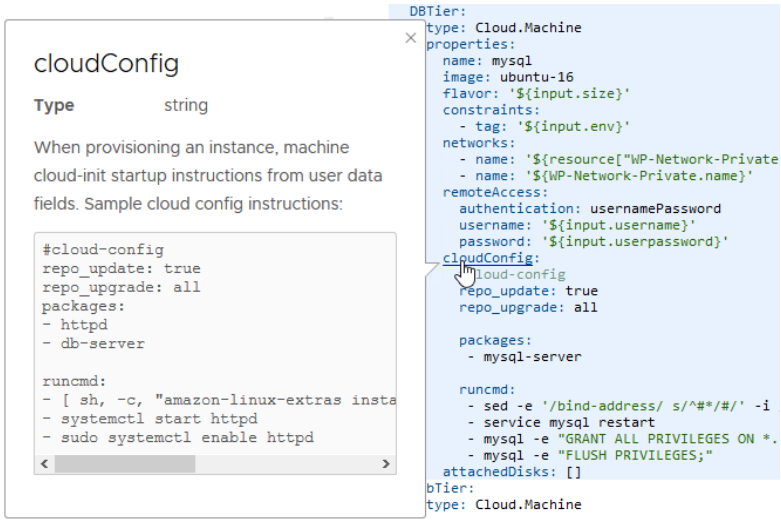
A solid line between resources indicates that the resources must end up in the same place. Even though you can add a connection on the canvas, deployment fails if conflicting placement constraint tags are present. For example, deployment fails if you connect resources where one is hard-constrained to a test us-west-1 cloud zone, and the other to a production us-east-1 cloud zone.

How to create valid vRealize Automation Cloud Assembly blueprint code

Adding vRealize Automation Cloud Assembly components 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 vRealize Automation Cloud Assembly editor includes syntax completion and error checking features.

Editor Hints	Example
Available values	
Allowed properties	
Child properties	
Syntax errors	
Ctrl+F to search	

Editor Hints	Example
Optional parameters	
Schema help	For all of the custom properties, you can also refer to the consolidated resource schema on the VMware {code} site. 

How to enhance a simple vRealize Automation Cloud Assembly blueprint

There are advanced vRealize Automation Cloud Assembly blueprint code possibilities that can take a simple blueprint to the next level.

The techniques described here require some comfort with infrastructure code. Fortunately, vRealize Automation Cloud Assembly code is human readable and fairly easy to follow.

How user input can customize a vRealize Automation Cloud Assembly blueprint

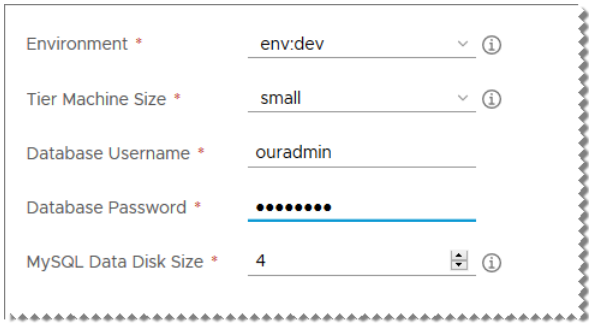
As a blueprint developer, you use input parameters so that users can make custom selections at request time.

When users supply inputs, you no longer need to save multiple copies of blueprints that are only slightly different. In addition, inputs can prepare a blueprint for Day 2 operations, because catalog users are prompted for fresh inputs whenever they update a deployment.

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, see the consolidated resource schema at vRealize Automation Blueprint Schema on VMware {code}.

The following inputs show how you might create one blueprint for a MySQL database server, where users can deploy that one blueprint to different cloud resource environments and apply different capacity and credentials each time.



How to define blueprint input parameters

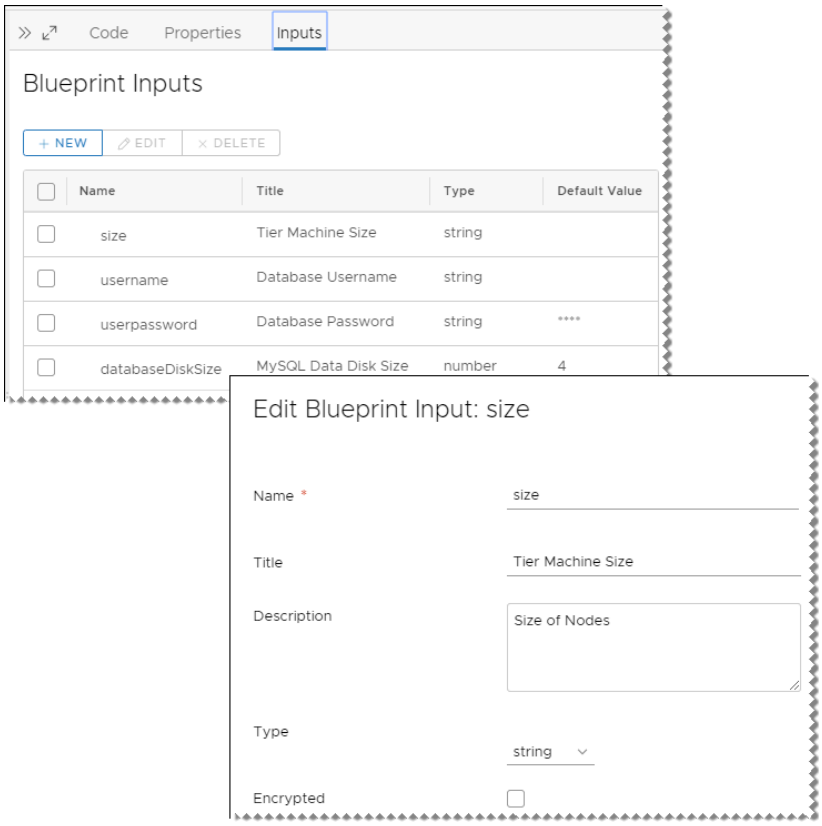
Add an `inputs` section to your blueprint code, where you set the selectable values.

In the following example, machine size, operating system, and number of clustered servers are selectable.

```
inputs:
  wp-size:
    type: string
    enum:
      - small
      - medium
    description: Size of Nodes
    title: Node Size
  wp-image:
    type: string
    enum:
      - coreos
      - ubuntu
    title: Select Image/OS
  wp-count:
    type: integer
    default: 2
```

```
    maximum: 5
    minimum: 2
    title: Wordpress Cluster Size
    description: Wordpress Cluster Size (Number of nodes)
```

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.



How to reference blueprint 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 blueprint code, you cannot use the word `input` except to indicate an input parameter.

```
resources:
  WebTier:
    type: Cloud.Machine
    properties:
```

```
      name: wordpress
      flavor: '${input.wp-size}'
      image: '${input.wp-image}'
      count: '${input.wp-count}'
```

List of input properties

Property	Description
const	Used with oneOf. The real value associated with the friendly title.
default	Prepopulated value for the input. The default must be of the correct type. Do not enter a word as the default for an integer.
description	User help text for the input.
encrypted	Whether to encrypt the input that the user enters, true or false. Passwords are usually encrypted.

enum	A drop-down menu of allowed values.
	Use the following example as a format guide.
	`enum:`
	`  - value 1`
	`  - value 2`

format	Sets the expected format for the input. For example, (25/04/19) supports date-time. Allows the use of the date picker in vRealize Automation Service Broker custom forms.
items	Declares items within an array. Supports number, integer, string, Boolean, or object.
maxItems	Maximum number of selectable items within an array.
maxLength	Maximum number of characters allowed for a string. For example, to limit a field to 25 characters, enter `maxLength: 25`.
maximum	Largest allowed value for a number or integer.
minItems	Minimum number of selectable items within an array.
minLength	Minimum number of characters allowed for a string.
minimum	Smallest allowed value for a number or integer.
oneOf	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.

Property	Description
pattern	Allowable characters for string inputs, in regular expression syntax. For example, `'[a-z]+'` or `'[a-z0-9A-Z@#$]+'`
properties	Declares the key:value properties block for objects.
readOnly	Used to provide a form label only.
title	Used with oneOf. The friendly name for a const value. The title appears on the user input form at deployment time.
type	Data type of number, integer, string, Boolean, or object.
writeOnly	Hides keystrokes behind asterisks in the form. Cannot be used with enum. Appears as a password field in vRealize Automation Service Broker custom forms.

Additional examples

String with enumeration

```
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

```
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

```
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.
  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
```

How to set the component deployment sequence in vRealize Automation Cloud Assembly

When you deploy a vRealize Automation Cloud Assembly blueprint, one component might need another component to be available first.

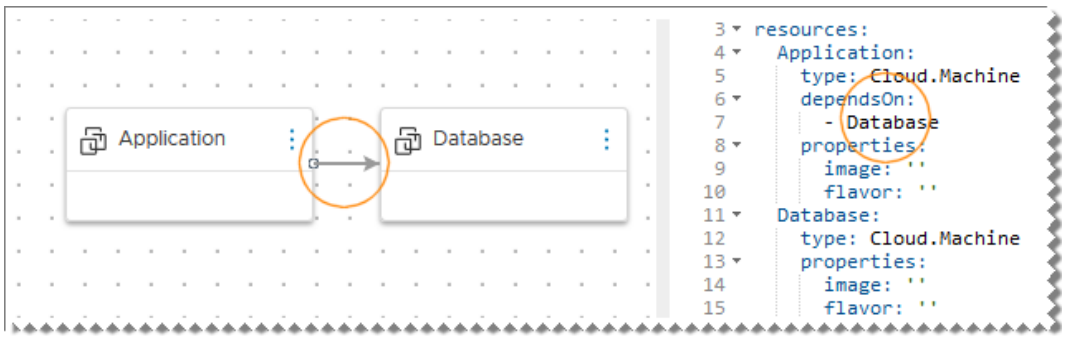
How to create an explicit dependency

Sometimes, a component 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 component and ending at the component to be deployed first.
- Code editor option—add a `dependsOn` property to the dependent component, and identify the component to be deployed first.

  An explicit dependency creates a solid arrow in the canvas.



How to create an implicit dependency or property binding

Sometimes, a component property needs a value found in a property of another component. 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 a property binding, an implicit dependency controls build order by waiting until the needed property is available before deploying the dependent component. You add an implicit dependency using the code editor.

- Edit the dependent component, adding a property that identifies the component and property that must exist first.

  An implicit dependency or property binding creates a dashed arrow in the canvas.

How to use expressions to make vRealize Automation Cloud Assembly blueprint code more versatile

For increased flexibility, you can add expressions to vRealize Automation Cloud Assembly blueprint code.

Expressions use the **`${expression}`** construct, as shown in the following examples.

The examples are pruned to show only the important lines. The entire, unedited blueprint appears at the end.

Examples

At deployment time, allow the user to paste in the encrypted key needed for remote access:

```
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*:

```
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:

```
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:

```
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:

```
resources:
  frontend:
    type: Cloud.Machine
    properties:
      networks:
        - name: '${resource.Cloud_Network_1.name}'
  apitier:
    type: Cloud.Machine
    properties:
      networks:
        - name: '${resource.Cloud_Network_1.name}'
  Cloud_Network_1:
    type: Cloud.Network
    properties:
      name: Default
      networkType: existing
```

Encrypt access credentials submitted to the API:

```
resources:
  apitier:
    type: Cloud.Machine
    properties:
      cloudConfig: |
        #cloud-config
        runcmd:
          - export apikey=${base64_encode(input.username:input.password)}
          - curl -i -H 'Accept:application/json' -H 'Authorization:Basic :$apikey' http://
example.com
```

Discover the address of the API machine:

```
resources:
  frontend:
    type: Cloud.Machine
    properties:
      cloudConfig: |
        runcmd:
          - echo ${resource.apitier.networks[0].address}
  apitier:
    type: Cloud.Machine
    properties:
      networks:
        - name: '${resource.Cloud_Network_1.name}'
```

Complete Blueprint

```
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:
        - name: '${resource.Cloud_Network_1.name}'
  apitier:
    type: Cloud.Machine
    properties:
      folderName: '${input.environment == "VMC" ? "Workload" : ""}'
      image: ubuntu
      flavor: small
      cloudConfig: |
        #cloud-config
        runcmd:
          - export apikey=${base64_encode(input.username:input.password)}
          - curl -i -H 'Accept:application/json' -H 'Authorization:Basic :$apikey' http://
example.com
      remoteAccess:
        authentication: publicPrivateKey
        sshKey: '${input.sshKey}'
```

```
      constraints:
        - tag: '${"env:" + to_lower(input.environment)}'
      networks:
        - name: '${resource.Cloud_Network_1.name}'
  Cloud_Network_1:
    type: Cloud.Network
    properties:
      name: Default
      networkType: existing
      constraints:
        - tag: '${"env:" + to_lower(input.environment)}'
```

vRealize Automation Cloud Assembly blueprint expression syntax

The vRealize Automation Cloud Assembly blueprint expression syntax exposes all of the capabilities of expressions.

The syntax is only partly represented in the examples shown in How to use expressions to make vRealize Automation Cloud Assembly blueprint code more versatile.

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 \"

  ' is escaped as \'

  \ 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.

Cluster count index

Syntax:

`count.index`

Example:

`${count.index == 0 ? "primary" : "secondary"}` (Return the node type for clustered resources.)

Limitations:

Use of `count.index` for resource allocation is not supported. For example, the following capacity expression fails when it references the position within an array of disks created at input time.

```
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
    properties:
      capacityGb: '${input.disks[count.index].size}'
      count: '${length(input.disks)}'
```

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"}`

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`*`(`*`arguments`*`...)}`

Example:

```
${to_lower(resource.app.name)}
```

Table 6-1. Functions

Function	Description
abs(number)	Absolute number value
floor(number)	Returns the largest (closest to positive infinity) value that is less than or equal to the argument and is equal to a mathematical integer
ceil(number)	Returns the smallest (closest to negative infinity) value that is greater than or equal to the argument and is equal to a mathematical integer
to_lower(str)	Convert string to lower case
to_upper(str)	Convert string to upper case
contains(array, value)	Check if array contains a value
contains(string, value)	Check if string contains a value
join(array, delim)	Join array of strings with a delimiter and return a string
split(string, delim)	Split string with a delimiter and return array of strings
slice(array, begin, end)	Return slice of array from begin index to end index
reverse(array)	Reverse entries of array
starts_with(subject, prefix)	Check if subject string starts with prefix string
ends_with(subject, suffix)	Check if subject string ends with suffix string
replace(string, target, replacement)	Replace string containing target string with target string
substring(string, begin, end)	Return substring of string from begin index until end index
format(format, values...)	Return a formatted string using Java Class Formatter format and values.
keys(map)	Return keys of map
values(map)	Return values of map
merge(map, map)	Return a merged map
length(string)	Return string length
length(array)	Return array length
max(array)	Return maximum value from array of numbers
min(array)	Return minimum value from array of numbers
sum(array)	Return sum of all values from array of numbers
avg(array)	Return average of all values from array of numbers

Table 6-1. Functions (continued)

Function	Description
digest(value, type)	Return digest of value using supported type (md5, sha1, sha256, sha384, sha512)
to_string(value)	Return string representation of the value
to_number(string)	Parse string as number
not_null(array)	Return the first entry which is not null
base64_encode(string)	Return base64 encoded value
base64_decode(string)	Return decoded base64 value
now()	Return current time in ISO-8601 format
uuid()	Return randomly generated UUID
from_json(string)	Parse json string
to_json(value)	Serialize value as json string
json_path(value, path)	Evaluate path against value using XPath for JSON.
matches(string, regex)	Check if string matches a regex expression
url_encode(string)	Encode string using url encoding specification
trim(string)	Remove leading and trailing spaces

How to automatically initialize a machine in a vRealize Automation Cloud Assembly blueprint

You can apply machine initialization in vRealize Automation Cloud Assembly through vSphere customization specifications, or by running commands directly.

A property in your blueprint code can reference a vSphere customization specification by name. Alternatively, you can add a cloudConfig section to the blueprint, in which you embed specific commands.

Use caution if you attempt to combine embedded commands and customization specification initialization. They aren't formally compatible and might produce inconsistent or unwanted results when used together.

For an example of how a customization specification interferes with commands in a cloudConfig section, see How to deploy a Linux machine with a static IP address.

vSphere customization specifications in vRealize Automation Cloud Assembly blueprints

Customization specifications let you apply guest operating system settings at deployment time, when deploying to vSphere based cloud zones.

The customization specification must exist in vSphere, at the target that you deploy to.

Edit the blueprint code directly. The following example points to a `cloud-assembly-linux` customization specification for a WordPress host on vSphere.

```
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'
```

Customization specifications versus initialization 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 to embed initialization commands in a cloudConfig section of a blueprint.

For more about cloudConfig sections in blueprints, see Configuration commands in vRealize Automation Cloud Assembly blueprints.

Configuration commands in vRealize Automation Cloud Assembly blueprints

You can add a cloudConfig section to vRealize Automation Cloud Assembly blueprint code, in which you add machine initialization commands that run at deployment time.

- Linux—initialization commands follow the open cloud-init standard.
- Windows—initialization commands use Cloudbase-init.

Note 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.

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

In vRealize Automation Cloud Assembly, you can also add initialization commands up front, to a machine image, when configuring infrastructure. All blueprints that reference the source image get the same initialization.

Important You might have an image map and a blueprint where both contain initialization commands. At deployment time, the commands merge, and vRealize Automation 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 for additional details.

The following example cloudConfig section is taken from WordPress use case: create a basic blueprint blueprint code for the Linux-based MySQL server.

To ensure correct interpretation of commands, always include the pipe character `cloudConfig: |` as shown.

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.

```
      cloudConfig: |
        #cloud-config
        repo_update: true
        repo_upgrade: all
        packages:
         - apache2
         - php
         - php-mysql
         - libapache2-mod-php
         - php-mcrypt
         - mysql-client
        runcmd:
         - mkdir -p /var/www/html/mywordpresssite && cd /var/www/html && wget
https://wordpress.org/latest.tar.gz && tar -xzf /var/www/html/latest.tar.gz -C /var/www/html/
mywordpresssite --strip-components 1
         - 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;"
         - mv /var/www/html/mywordpresssite/wp-config-sample.php /var/www/html/
mywordpresssite/wp-config.php
         - 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
```

How to deploy a Linux machine with a static IP address

When deploying to vSphere, a static IP address requires that vRealize Automation Cloud Assembly generate a vSphere customization specification, which can interfere with cloud-init commands.

Problem

- A vRealize Automation Cloud Assembly blueprint includes `assignment: static` to apply a static IP address to a vSphere virtual machine.
- The blueprint also contains a cloudConfig section, which includes initialization commands that are run using cloud-init.
- To give the virtual machine a static IP, vRealize Automation Cloud Assembly dynamically generates a vSphere customization specification to apply.
- Whenever a customization specification is applied, the last operation restarts the virtual machine.
- The customization specification doesn't know that cloud-init commands are running, so the restart interrupts them.
- The cloud-init commands only run upon first boot and don't automatically recover when interrupted.
- The resulting virtual machine remains only partially configured.

Workaround

Create a machine template that includes a timed disabling of cloud-init. Then, deploy machines based on the template so that the customization specification and restart can occur before cloud-init.

Example procedure—Ubuntu 18.04

The following steps apply to Ubuntu 18.04. You might need to make adjustments and adapt the ideas presented here for other Linux versions or offerings.

1 Create the virtual machine, and bring it up to date with version updates and packages that you want.

  Be aware that other Linux offerings might not have cloud-init pre-installed, but Ubuntu 18.04 does.

2 Reconfigure cloud-init, setting the datasource to OVF.

  ```
  sudo dpkg-reconfigure cloud-init
  ```

3 Edit the following file.

`/etc/cloud/cloud.cfg`

a Enable traditional guest operating system customization (GOSC) by adding the following line.

```
disable_vmware_customization: true
```

b Make sure that network configuration is enabled. Delete or comment out the disable setting, if it exists.

```
network:
  # config: disabled
```

Alternatively, inspect all configuration files in the following directory.

`/etc/cloud/cloud.cfg.d/*`

Delete any files that contain a `network: {config: disabled}` setting.

4 Edit the following file.

`/usr/lib/tmpfiles.d/tmp.conf`

- Prevent the temporary directory from clearing by commenting out the setting.

```
# D /tmp 1777 root root -
```

5 Edit the following file.

`/lib/systemd/system/open-vmtools.service`

- Configure open-vmtools to start after dbus.service by adding the following line under the `[Unit]` section.

```
After=dbus.service
```

6 Create the new, empty file that disables cloud-init.

```
sudo touch /etc/cloud/cloud-init.disabled
```

7 Create a re_init.sh script. After a cron job delay that pauses for the customization specification, the script re-enables and initializes cloud-init.

```
sudo rm -rf /etc/cloud/cloud-init.disabled
sudo cloud-init init
sleep 20
sudo cloud-init modules --mode config
sleep 20
sudo cloud-init modules --mode final
```

8 Add run permission for the script.

```
sudo chmod +x re_init.sh
```

9 Create the cron job that will run after 90 seconds of sleep at startup. Type `crontab -e` and enter the following:

```
@reboot ( sleep 90 ; sh /script_path/delay_init.sh )
```

You can apply more then 90 seconds if customization specifications and restarts are taking longer to finish.

10 Create a cleaner.sh script, which cleans the template. Replace `cloudadmin` with your own user that you set up during operating system installation.

The sample script is specific to Ubuntu. To create a script for other Linux offerings, make sure to include the highlighted, mandatory sections.

```
#!/bin/bash

# Add usernames to add to /etc/sudoers for passwordless sudo
users=("ubuntu" "cloudadmin")

for user in "${users[@]}"
do
cat /etc/sudoers | grep ^$user
RC=$?
if [ $RC != 0 ]; then
bash -c "echo \"$user ALL=(ALL) NOPASSWD:ALL\" >> /etc/sudoers"
fi
done

#grab Ubuntu Codename
codename="$(lsb_release -c | awk {'print $2}')"


#Stop services for cleanup
service rsyslog stop

#clear audit logs
if [ -f /var/log/audit/audit.log ]; then
cat /dev/null > /var/log/audit/audit.log
fi
if [ -f /var/log/wtmp ]; then
cat /dev/null > /var/log/wtmp
fi
if [ -f /var/log/lastlog ]; then
cat /dev/null > /var/log/lastlog
fi

#cleanup persistent udev rules
if [ -f /etc/udev/rules.d/70-persistent-net.rules ]; then
rm /etc/udev/rules.d/70-persistent-net.rules
fi

#cleanup /tmp directories
rm -rf /tmp/*
rm -rf /var/tmp/*

#cleanup current ssh keys
#rm -f /etc/ssh/ssh_host_*
```

```
#cat /dev/null > /etc/hostname

#cleanup apt
apt-get clean

#Clean Machine ID

truncate -s 0 /etc/machine-id
rm /var/lib/dbus/machine-id
ln -s /etc/machine-id /var/lib/dbus/machine-id

#Clean Cloud-init
cloud-init clean --logs --seed

#cleanup shell history
history -w
history -c
```

11 Add run permission for the template cleaning script.

```
sudo chmod +x cleaner.sh
```

12 In Ubuntu 18.04, the cleanup script needs root privileges. Edit the following file.

```
/etc/ssh/sshd_config
```

a Make sure that you can switch to root user.

```
PermitRootLogin yes
```

b Set a password for root.

```
sudo passwd root
```

13 Run the cleanup script.

```
sudo ./script_path/cleaner.sh
```

14 (Optional) For security, revert step 12 to prevent further root logins.

15 Shut down the virtual machine, and use vSphere to turn it into a template.

Template updates

The cron job runs whenever you update the template. If your update takes longer than the delay (such as 90 seconds) you need to re-add the `/etc/cloud/cloug-init.disabled` file and re-run the cleanup script before shutting down the template. Otherwise, cloud-init won't be disabled at first boot, and the customization specification restart goes back to interrupting the cloud-init commands.

Troubleshooting

If you suspect that the vSphere customization specification is preventing cloud-init completion, temporarily disable the customization specification, and determine whether cloud-init can finish as expected. To temporarily disable the customization specification, use the `customizeGuestOS: false` property.

```
    properties:
      image: ubuntu
      cpuCount: 1
      totalMemoryMB: 8192
      customizeGuestOS: false
```

How to make a vRealize Automation Cloud Assembly deployment wait for initialization

Sometimes, a virtual machine needs to be fully started before proceeding with vRealize Automation 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 uses 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 How to set the component deployment sequence in vRealize Automation Cloud Assembly.
- Phone home requires cloudConfig commands in the blueprint.
- Your creativity is a factor. A cloudConfig section might include embedded wait time between operations, which can be used in concert with phone home.
- Blueprint-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 vRealize Automation Cloud Assembly.

To wait for machine initialization by using phone home in vRealize Automation Cloud Assembly, add a `cloudConfigSettings` section to the blueprint:

```
cloudConfigSettings:
  phoneHomeShouldWait: true
  phoneHomeTimeoutSeconds: 600
  phoneHomeFailOnTimeout: true
```

Property	Description
phoneHomeShouldWait	Whether to wait for initialization, true or false.
phoneHomeTimeoutSeconds	When to decide whether to proceed with deployment even though initialization is still running. Default is 10 minutes.
phoneHomeFailOnTimeout	Whether to proceed with deployment after timing out, true or false. Note that even when proceeding, deployment might still fail for separate reasons.

How to enable remote access in vRealize Automation Cloud Assembly blueprints

To remotely access a machine that vRealize Automation Cloud Assembly has deployed, you add properties, before deployment, to the blueprint 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 blueprint, to automatically copy the keys upon provisioning. The specifics aren't documented here, but How to automatically initialize a machine in a vRealize Automation Cloud Assembly blueprint provides general information about cloudConfig.

Generate a key pair at vRealize Automation Cloud Assembly provisioning time

If you don't have your own public-private key pair for remote access authentication, you can have vRealize Automation Cloud Assembly generate a key pair.

Use the following code as a guideline.

1 In vRealize Automation Cloud Assembly, before provisioning, add `remoteAccess` properties to the blueprint as shown in the example.

The username is optional. If you omit it, the system generates a random ID as the username.

Example:

```
type: Cloud.Machine
properties:
  name: our-vm2
  image: Linux18
  flavor: small
  remoteAccess:
    authentication: generatedPublicPrivatekey
    username: testuser
```

2 In vRealize Automation Cloud Assembly, provision the machine from its blueprint, and bring it to a started-up state.

The provisioning process generates the keys.

3 Locate the key name in the **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 vRealize Automation 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 vRealize Automation 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.

For now, just generate and save the keys locally.

2 In vRealize Automation Cloud Assembly, before provisioning, add `remoteAccess` properties to the blueprint 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:

```
type: Cloud.Machine
properties:
  name: our-vm1
  image: Linux18
  flavor: small
  remoteAccess:
    authentication: publicPrivateKey
    sshKey: ssh-rsa Iq+5aQgBP3ZNT4o1baP5Ii+dstIcowRRkyobbfpA1mj9tslf
qGxvU66PX9IeZax5hZvNWFgjw6ag+ZlzndOLhVdVoW49f274/mIRild7UUW...
    username: testuser
```

3 In vRealize Automation Cloud Assembly, provision the machine from its blueprint, 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 vRealize Automation Cloud Assembly

By adding an AWS key pair name to the blueprint, you can remotely access a machine that vRealize Automation 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.

```
type: Cloud.Machine
properties:
  image: Ubuntu
  flavor: small
  remoteAccess:
    authentication: keyPairName
    keyPair: cas-test
constraints:
  - tag: 'cloud:aws'
```

Supply a username and password to vRealize Automation Cloud Assembly

By adding a username and password to the blueprint, you can have simple remote access to a machine that vRealize Automation 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 vRealize Automation Cloud Assembly, before provisioning, add `remoteAccess` properties to the blueprint as shown in the example.

   Set the username and password to the account that you expect to log in with.

   Example:

   ```
   type: Cloud.Machine
   properties:
     name: our-vm3
     image: Linux18
     flavor: small
   ```

```
remoteAccess:
  authentication: usernamePassword
  username: testuser
  password: admin123
```

2 In vRealize Automation Cloud Assembly, provision the machine from its blueprint, 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.

How to save different versions of a vRealize Automation Cloud Assembly blueprint

As a blueprint developer, you can safely capture a snapshot of a working blueprint before risking further changes.

At deployment time, you can select any of your versions to deploy.

How to capture a blueprint version

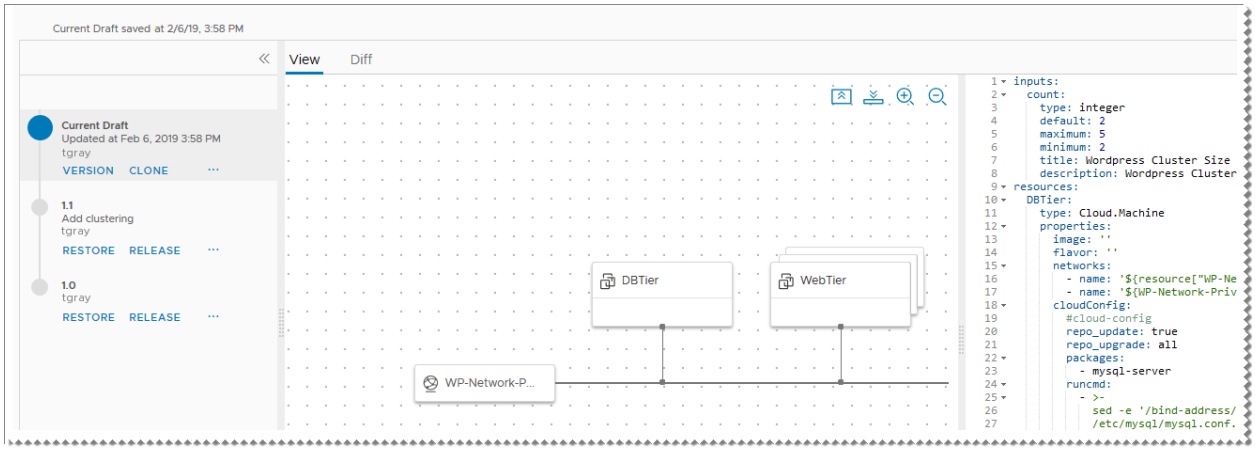
From the design page, click **Version**, and provide a name.

The name must be alphanumeric, with no spaces, and only periods, hyphens, and underscores allowed as special characters.

How to restore 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.



How to release a version to users of vRealize Automation Service Broker

From the design page, click **Version History**.

On the left, select a version and click **Release**. You cannot release the current draft until you version it.

When more than one version of a blueprint is released, vRealize Automation Service Broker uses the most recent one.

How to compare blueprint versions

When changes and versions accumulate, you might want to identify differences among them.

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

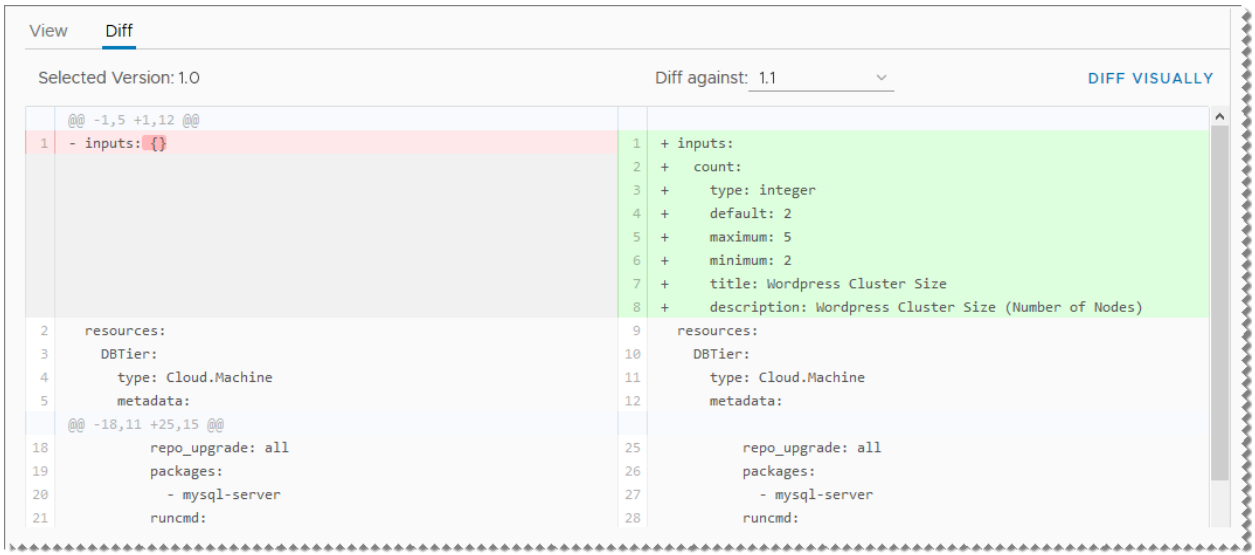
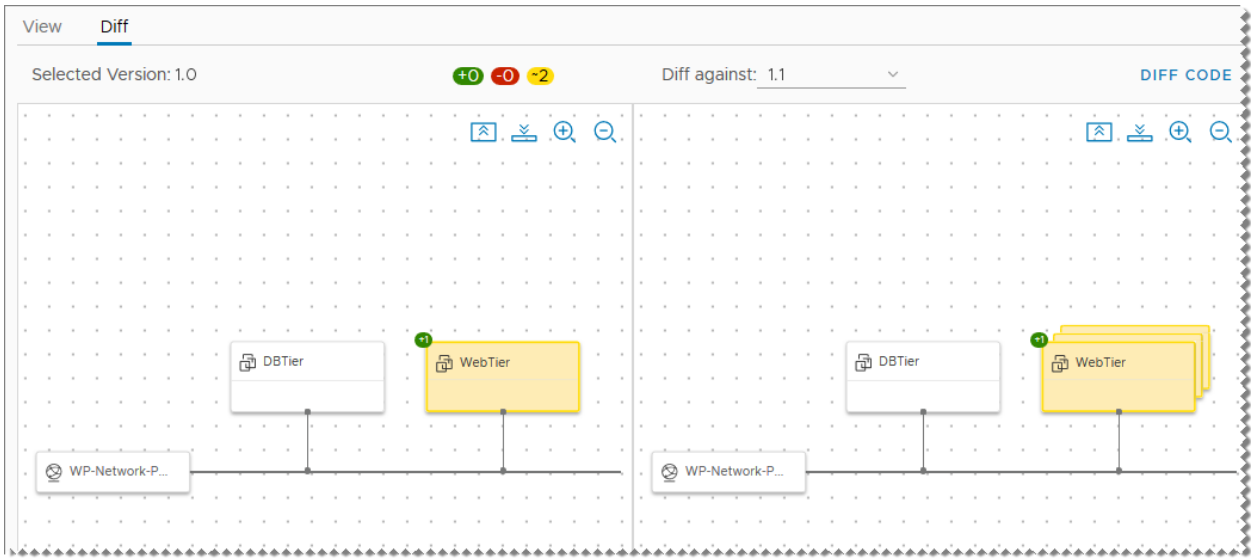


Figure 6-2. Visual Topology Differences



How to clone a blueprint

Although it's not the same as saving a version, from the design page, **Actions > Clone** makes a copy of the current blueprint for alternative development.

How do I customize the names of deployed resources using vRealize Automation 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 vRealize Automation Cloud Assembly project.

For example, your host naming convention is to prefix a resource as *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.

All resource names must be unique. Use the incremental number property to ensure uniqueness. The numbers increment for all deployments, including those that are named by vRealize Automation Cloud Assembly. As your system becomes more robust, the numbering might appear random, but they still ensure uniqueness.

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 blueprint 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.

5. Deploy a blueprint associated with the project to verify that the custom name is applied to the resource.

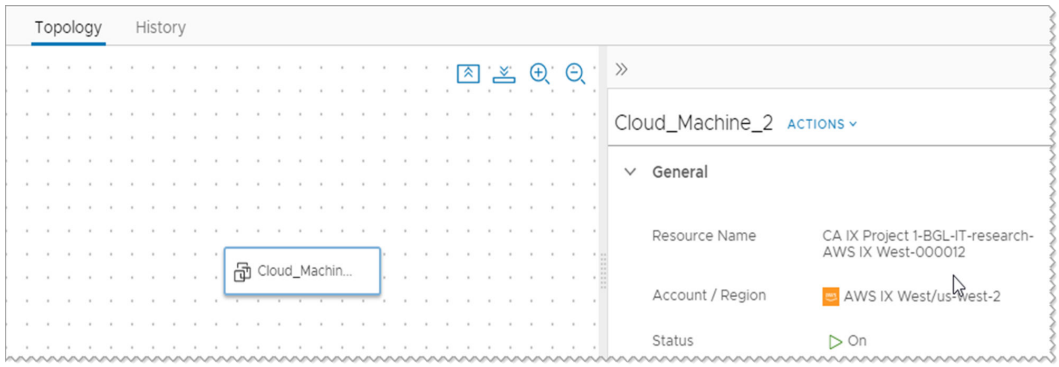
   a. Click the **Blueprints** tab, and then click a blueprint associated with the project.

   b. Deploy the blueprint.

   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.



6 If you deployed a test blueprint to verify the naming convention, you can delete the deployment.

What to do next

Create custom naming templates for your other projects.

What are the vRealize Automation resource properties

The vRealize Automation infrastructure-as-code editor lets you click or hover for syntax and code completion help. To view the complete set of blueprint 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 blueprints.

- vRealize Automation Resource Type Schema on VMware {code}

What are some blueprint code examples

Blueprint code in vRealize Automation 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.

vSphere machine examples in vRealize Automation Cloud Assembly blueprints

These basic examples define vSphere resources within vRealize Automation Cloud Assembly blueprints.

Resource	Example Blueprint

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: >-`
	`  https://bintray.com/vmware/photon/download_file?file_path=2.0%2FRC%2Fova%2Fphoton-custom-hw11-2.0-31bb961.ova`
	`  attachedDisks:`
	`  - source: '${demo-vsphere-disk-001.id}'`

vSphere machine from a linked clone image (append a forward slash and the snapshot name)	`resources:`
	`  demo-machine:`
	`  type: Cloud.vSphere.Machine`
	`  properties:`
	`  imageRef: 'demo-machine/snapshot-01'`
	`  cpuCount: 1`
	`  totalMemoryMB: 1024`

Resource	Example Blueprint

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'`

vSphere machine with multiple NICs	`resources:`
	`  demo-machine:`
	`  type: Cloud.Machine`
	`  properties:`
	`  image: ubuntu`
	`  flavor: small`
	`  networks:`
	`  - name: '${network-01.name}'`
	`  deviceIndex: 0`
	`  - name: '${network-02.name}'`
	`  deviceIndex: 1`
	`  network-01:`
	`  type: Cloud.vSphere.Network`
	`  properties:`
	`  name: network-01`
	`  network-02:`
	`  type: Cloud.vSphere.Network`
	`  properties:`
	`  name: network-02`

vSphere machine from a snapshot image	`resources:`
	`  demo-machine:`
	`  type: Cloud.vSphere.Machine`
	`  properties:`
	`  imageRef: 'demo-machine/snapshot-01'`
	`  cpuCount: 1`
	`  totalMemoryMB: 1024`

vSphere machine with an attached tag in vCenter	`resources:`
	`  demo-machine:`
	`  type: Cloud.Machine`
	`  properties:`
	`  flavor: small`
	`  image: ubuntu`
	`  tags:`
	`  - key: env`
	`  value: demo`

vSphere machine with a customization spec	`resources:`
	`  demo-machine:`
	`  type: Cloud.vSphere.Machine`
	`  properties:`
	`  name: demo-machine`
	`  image: ubuntu`
	`  flavor: small`
	`  customizationSpec: Linux`

Resource	Example Blueprint

vSphere machine with a vSphere network component and static IP address	<pre>resources:
	  demo-network:
	    type: Cloud.vSphere.Network
	    properties:
	      name: demo-network
	  demo-machine:
	    type: Cloud.vSphere.Machine
	    properties:
	      image: ubuntu
	      flavor: small
	      networks:
	        - name: demo-network
	          assignment: static</pre>

vSphere machine with remote access	<pre>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}</pre>

Documented vRealize Automation Cloud Assembly blueprint example

By including a thorough set of comments, this example lets you review the structure and purpose of the sections in a vRealize Automation Cloud Assembly blueprint.

```
# ****************************************************************************
#
# This WordPress blueprint 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 blueprint deploys a minimum of 3 virtual machines and runs scripts
# to install packages.
#
# ****************************************************************************
#
# ----------------------------------------------------------------------------
# Blueprints need a descriptive name and version if
# source controlled in git.
# ----------------------------------------------------------------------------
name: WordPress Blueprint with Comments
formatVersion: 1
version: 1
#
# ----------------------------------------------------------------------------
# Inputs create user selections that appear at deployment time. Inputs
# can set placement decisions and configurations, and are referenced
# later, by the resources section.
# ----------------------------------------------------------------------------
inputs:
#
# ----------------------------------------------------------------------------
# Choose a cloud endpoint. 'Title' is the visible
# option text (oneOf allows for the friendly title). 'Const' is the
# tag that identifies the endpoint, which was set up earlier, under the
# Cloud Assembly Infrastructure tab.
# ----------------------------------------------------------------------------
  platform:
    type: string
    title: Deploy to
    oneOf:
      - title: AWS
        const: aws
      - title: Azure
        const: azure
      - title: vSphere
        const: vsphere
    default: vsphere
#
# ----------------------------------------------------------------------------
# Choose the operating system. Note that the Cloud Assembly
```

```
# Infrastructure must also have an AWS, Azure, and vSphere Ubuntu image
# mapped. In this case, enum sets the option that you see, meaning there's
# no friendly title feature this time. Also, only Ubuntu is available
# here, but having this input stubbed in lets you add more operating
# systems later.
# ---------------------------------------------------------------------
  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 blueprint 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
```

```
    description: Database Username
  userpassword:
    type: string
    pattern: '[a-z0-9A-Z@#$]+'
    encrypted: true
    title: Database Password
    description: 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: '${input.osimage}'
# ----------------------------------------------------------------------
      image: Ubuntu
#
# ----------------------------------------------------------------------
# Hard-coded capacity to use. Note that the Cloud Assembly
# Infrastructure must also have AWS, Azure, and vSphere flavors
# such as small, medium, and large mapped.
# ----------------------------------------------------------------------
      flavor: small
#
# ----------------------------------------------------------------------
# Tag the database machine to deploy to the cloud vendor chosen from the
# user input. Tags are case-sensitive, so 'to_lower' forces the tag to
# lowercase to ensure a match with a site's tagging convention. It's
# important if platform input were to contain any upper case characters.
# ----------------------------------------------------------------------
      constraints:
        - tag: '${"env:" + to_lower(input.platform)}'
#
# ----------------------------------------------------------------------
# Also tag the database machine with any free-form tags that were created
# during user input.
# ----------------------------------------------------------------------
      tags: '${input.Mtags}'
#
# ----------------------------------------------------------------------
# Set the database cluster size by referencing the dbenvsize user
# input. Small is one machine, and large defaults to two.
# ----------------------------------------------------------------------
      count: '${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.
# ----------------------------------------------------------------------
      networks:
        - name: '${resource.WP_Network.name}'
          network: '${resource.WP_Network.id}'
#
# ----------------------------------------------------------------------
# Enable remote access to the database server. Reference the credentials
```

```
# from the user input.
# -----------------------------------------------------------------------
      remoteAccess:
        authentication: usernamePassword
        username: '${input.username}'
        password: '${input.userpassword}'
#
# -----------------------------------------------------------------------
# You are free to add custom properties, which might be used to initiate
# an extensiblity subscription, for example.
# -----------------------------------------------------------------------
      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.
# -----------------------------------------------------------------------
      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 extensiblity 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: '${"env:" + to_lower(input.platform)}'
#
# -----------------------------------------------------------------------
# Add a variable to connect the machine to a network resource based on
# a property binding to another resource. In this case, it's the
# 'WP_Network' network that gets defined further below.
# -----------------------------------------------------------------------
      networks:
        - name: '${resource.WP_Network.name}'
          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:
          - mkdir -p /var/www/html/mywordpresssite && cd /var/www/html && wget
https://wordpress.org/latest.tar.gz && tar -xzf /var/www/html/latest.tar.gz -C /var/www/html/
mywordpresssite --strip-components 1
          - 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 ${resource.DBTier.networks[0].address} -e
"create database wordpress_blog;"
          - mv /var/www/html/mywordpresssite/wp-config-sample.php /var/www/html/
mywordpresssite/wp-config.php
          - sed -i -e s/"define('DB_NAME', 'database_name_here');"/"define('DB_NAME',
'wordpress_blog');"/ /var/www/html/mywordpresssite/wp-config.php && sed -i
-e s/"define('DB_USER', 'username_here');"/"define('DB_USER', 'root');"/ /var/www/html/
mywordpresssite/wp-config.php && sed -i -e s/"define('DB_PASSWORD',
'password_here');"/"define('DB_PASSWORD', 'mysqlpassword');"/ /var/www/html/mywordpresssite/
wp-config.php && sed -i -e s/"define('DB_HOST', 'localhost');"/"define('DB_HOST', '$
{resource.DBTier.networks[0].address}');"/ /var/www/html/mywordpresssite/wp-config.php
          - service apache2 reload
#
# ---------------------------------------------------------------------
# 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 blueprint useful. Note that
# you can also access an API to create blueprints, 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
#
# ************************************************************************
```

Network, security, and load balancer examples in vRealize Automation Cloud Assembly blueprints

These blueprint code examples illustrate some basic network, security, and load balancer configurations.

For examples of detailed network and security 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)

For a complete summary of all blueprint schema options, see vRealize Automation Resource Type Schema.

For more information about network types, see Using networks and network profiles in vRealize Automation Cloud Assembly.

Resource Scenario	Example Blueprint Code

vSphere machine with multiple NICs	`resources:`
	`  demo-machine:`
	`  type: Cloud.vSphere.Machine`
	`  properties:`
	`  image: ubuntu`
	`  flavor: small`
	`  networks:`
	`  - name: '${network-01.id}'`
	`  deviceIndex: 0`
	`  - name: '${network-02.id}'`
	`  deviceIndex: 1`
	`  Cloud_vSphere_Network_1:`
	`  type: Cloud.vSphere.Network`
	`  properties:`
	`  networkType: existing`
	`  name: network-01`
	`  Cloud_vSphere_Network_2:`
	`  type: Cloud.NSX.Network`
	`  properties:`
	`  networkType: existing`
	`  name: network-02`

Public cloud machine to use internal IP instead of public IP	`resources:`
	`  wf_proxy:`
	`  type: Cloud.Machine`
	`  properties:`
	`  image: ubuntu 16.04`
	`  flavor: small`
	`  constraints:`
	`  - tag: 'platform:vsphere'`
	`  networks:`
	`  - name: '${resource.wf_net.id}'`
	`  assignPublicIpAddress: false`

Routed network for NSX-V or NSX-T using the NSX network component type	`Cloud_NSX_Network_1:`
	`  type: Cloud.NSX.Network`
	`  properties:`
	`  networkType: routed`

Tagging NSX logical switches for an outbound network
For more information on this scenario, see community blog post Creating Tags in NSX with Cloud Assembly.
	`Cloud_NSX_Network_1:`
	`  type: Cloud.NSX.Network`
	`  properties:`
	`  networkType: outbound`
	`  tags:`
	`  - key: app`
	`  value: opencart`

Resource Scenario	Example Blueprint Code

Existing security group with a constraint tag applied to a machine NIC
To use an existing security group, enter *existing* for the `securityGroupType` property in the `Cloud.SecurityGroup` component. To create an on-demand security group, enter *new* for the `securityGroupType` property in the `Cloud.SecurityGroup` component.
You can assign tags to a `Cloud.SecurityGroup` component to allocate existing security groups by using tag constraints. Security groups that do not contain tags cannot be used in the blueprint.
	`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`

Resource Scenario	Example Blueprint Code

On-demand network with a 1 arm load balancer	`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:`

Resource Scenario	Example Blueprint Code

```
        - network: '$
{resource.Cloud_NSX_Network_1.id}'
  Provider_LoadBalancer_1:
    type: Cloud.LoadBalancer
    properties:
      name: OC-LB
      routes:
        - protocol: HTTP
          port: '80'
          instanceProtocol: HTTP
          instancePort: '80'
          healthCheckConfiguration:
            protocol: HTTP
            port: '80'
            urlPath: /index.html
            intervalSeconds: 60
            timeoutSeconds: 5
            unhealthyThreshold: 5
            healthyThreshold: 2
      network: '$
{resource.Cloud_NSX_Network_1.id}'
      internetFacing: false
      instances: '${resource["ubuntu-vm"].id}'
  Cloud_NSX_Network_1:
    type: Cloud.NSX.Network
    properties:
      networkType: existing
      constraints:
        - tag: nsxt24prod
```

Puppet-enabled blueprint with username and password access

In this example, you add Puppet configuration management to a blueprint 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 blueprint 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 blueprint examples and vCenter Puppet configuration blueprint examples for more examples of how you can configure different Puppet scenarios in vRealize Automation Cloud Assembly blueprints.

Prerequisites

- Set up a Puppet Enterprise instance on a valid network.

- Add your Puppet Enterprise instance to vRealize Automation Cloud Assembly using the Integrations feature. See Configure Puppet Enterprise integration in vRealize Automation 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 blueprint.

   a Select **Infrastructure > Manage > Integrations**.

   b Click **Add Integration** and select Puppet.

   c Enter the appropriate information on the Puppet configuration page.

Confguration	Description	Example Value
Hostname	Host name or IP address of the Puppet primary machine	Puppet-Ubuntu
SSH Port	SSH port for communication between vRealize Automation Cloud Assembly and Puppet primary machine. (Optional)	NA
Autosign secret	The shared secret configured on the Puppet primary machine that nodes should provide to support autosign certificate requests.	User specific
Location	Indicate whether the Puppet primary machine is on a private or public cloud. **Note** Cross cloud deployment is supported only if there is connectivity between the deployment compute resource and the Puppet primary machine.	
Cloud proxy	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.	NA
Username	SSH and RBAC user name for Puppet primary machine.	User specific. YAML value is '${input.username}'
Password	SSH and RBAC password for Puppet primary machine.	User specific YAML value is '${input.password}'
Use sudo commands for this user	Select to use sudo commands for the procidd.	true
Name	Puppet primary machine name.	PEMasterOnPrem
Description		

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 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.

```
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 blueprint examples

There are several options for configuring blueprints to support Puppet based configuration management on AWS compute resources.

Puppet management on AWS with username and password

Example of...	Sample Blueprint YAML

authentication of cloud configuration on any supported Amazon Machine Image.	<pre>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: |
	        #cloud-config
	        ssh_pwauth: yes
	        chpasswd:
	          list: |
	            ${input.username}:${input.password}
	          expire: false
	        users:
	          - default
	          - name: ${input.username}
	            lock_passwd: false
	            sudo: ['ALL=(ALL) NOPASSWD:ALL']
	            groups: [wheel, sudo, admin]
	            shell: '/bin/bash'
	            ssh-authorized-keys:
	              - ssh-rsa
	AAAAB3NzaC1yc2EAAAADAQABAAABAQDytVL+Q6/+vGbmkXoRpX
	dmettem@dmettem-m01.vmware.com
	        runcmd:
	          - echo "Defaults:${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</pre>

Authentication of cloud configuration on a custom Amazon Machine Image with an existing user.	<pre>inputs:
	  username:
	    type: string
	    title: Username
	    default: puppet
	  password:
	    type: string
	    title: Password
	    encrypted: true
	    default: VMware@123</pre>

Example of...	Sample Blueprint 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

Example of...	Sample Blueprint YAML

remoteAccess.authentication authentication on AWS with generatedPublicPrivateKey acces.	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 blueprint examples

There are several options for configuring blueprints 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 6-2.

Example of...	Sample Blueprint YAML

YAML code for Puppet on a vSphere OVA with username and password authentication.	<pre>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: ${input.username}
	            lock_passwd: false
	            sudo: ['ALL=(ALL) NOPASSWD:ALL']
	            groups: [wheel, sudo, admin]
	            shell: '/bin/bash'
	            ssh-authorized-keys:
	              - ssh-rsa
	AAAAB3NzaC1yc2EAAAADAQABAAABAQDytVL+Q6+vGbmkXoRpX
	dmettem@dmettem-m01.vmware.com
	        runcmd:
	          - echo "Defaults:${input.username}</pre>

YAML code for Puppet on a vSphere OVA with username and password authentication on the compute resource.	<pre>inputs:
	  username:
	    type: string
	    title: Username
	    default: puppet</pre>

Table 6-2. (continued)

Example of...	Sample Blueprint YAML
	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: ${input.username} lock_passwd: false sudo: ['ALL=(ALL) NOPASSWD:ALL'] groups: [wheel, sudo, admin] shell: '/bin/bash' ssh-authorized-keys: - ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQDytVL+Q6+vGbmkXoRpX dmettem@dmettem-m01.vmware.com runcmd: - echo "Defaults:${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 6-2. (continued)

Example of...	Sample Blueprint YAML

	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}'

Puppet on vSphere with generated PublicPrivateKey authentication

Table 6-3.

Example of...	Sample Blueprint YAML

YAML code for Puppet on a vSphere OVA with generated PublicPrivateKey authentication on the compute resource.	inputs: {}
	resources:
	  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
	  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}'
	  - echo "Defaults:${input.username}

How to use the vRealize Automation Cloud Assembly Marketplace

To jumpstart your resource library, download files from the vRealize Automation Cloud Assembly Marketplace.

The Marketplace provides finished blueprints and open virtualization images that are managed on the VMware Solution Exchange. Solution Exchange files that are tagged with `cloud assembly` appear under the vRealize Automation Cloud Assembly Marketplace tab.

How to access the Marketplace

In vRealize Automation Cloud Assembly, select **Infrastructure > Connections > Integrations**. Click **Add Integration**, click **My VMware**, and provide your My VMware account credentials.

How to download and use Marketplace blueprint files

In the **Marketplace** tab, click **Get**, and accept the blueprint EULA. Then, you can add the blueprint to a vRealize Automation Cloud Assembly project, or simply download it. You can upload a blueprint in the **Blueprints** 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 blueprint that you add to the team project. You then customize the blueprint for your resource environment, and release it. Then, you import the blueprint into the vRealize Automation Service Broker catalog so that your team can deploy it.

How to download and use 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 blueprint 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 blueprint code that needs to point to the OVF image.

How to extend and automate 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 vRealize Automation 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 vRealize Automation 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 vRealize Automation Cloud Assembly with your existing vRealize Orchestrator environment, you can use workflows in your extensibility subscriptions.

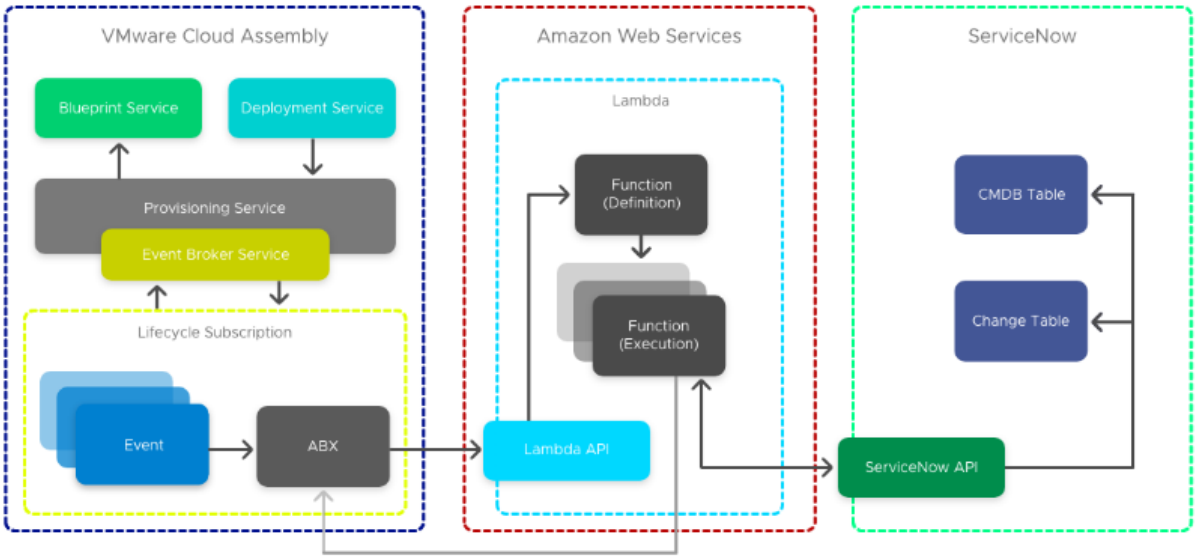
Extensibility action subscriptions

You can assign an extensibility action to a vRealize Automation 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 vRealize Automation Cloud Assembly with an Enterprise ITSM, like ServiceNow.

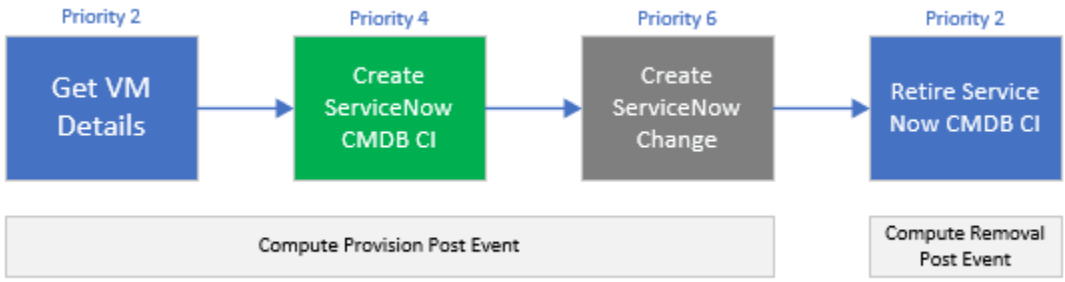


Enterprise users commonly integrate their Cloud Management Platform with an IT Service Management (ITSM) and Configuration Management Database (CMDB) platform for compliance. Following this example, you can integrate vRealize Automation Cloud Assembly with ServiceNow for CMDB and ITSM by using extensibility action scripts.

Note You can also integrate ServiceNow with vRealize Automation 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 vRealize Automation 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 vRealize Automation services 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.

Create ServiceNow Change

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 blueprint property: **`event.data["customProperties"]["enable_servicenow"] === "true"`**

  Note This property exists on blueprints that require a ServiceNow integration.

- Installed Python application.

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.

```
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.

```
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
        outputs = {}
        outputs['customProperties'] = inputs['customProperties']
        outputs['customProperties']['serviceNowSysId'] = serviceNowSysId;
        return outputs
```

4 Run the Creation action script.

```
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

vRealize Automation 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:

```
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/"+tableName+"/{0}".format(sys_id)
    headers = {'Content-type': 'application/json', 'Accept': 'application/json'}
    payload = {
        'state': 'Retired'
        }

    results = requests.put(
        url,
        json=payload,
```

```
        headers=headers,
        auth=(inputs['username'], inputs['password'])
    )
    print(results.text)
```

For more information on how you can use extensibility actions to integrate ServiceNow in vRealize Automation Cloud Assembly, see Extending Cloud Assembly with Action Based Extensibility for ServiceNow Integration.

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.

Parameter	Description
Action Name	Extensibility action name, preferably with 'TagVM' as a prefix or suffix.
Project	Project to test the extensibility action against.
Action Template	Tag VM
Runtime	Python
Script Source	Write Script

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**.

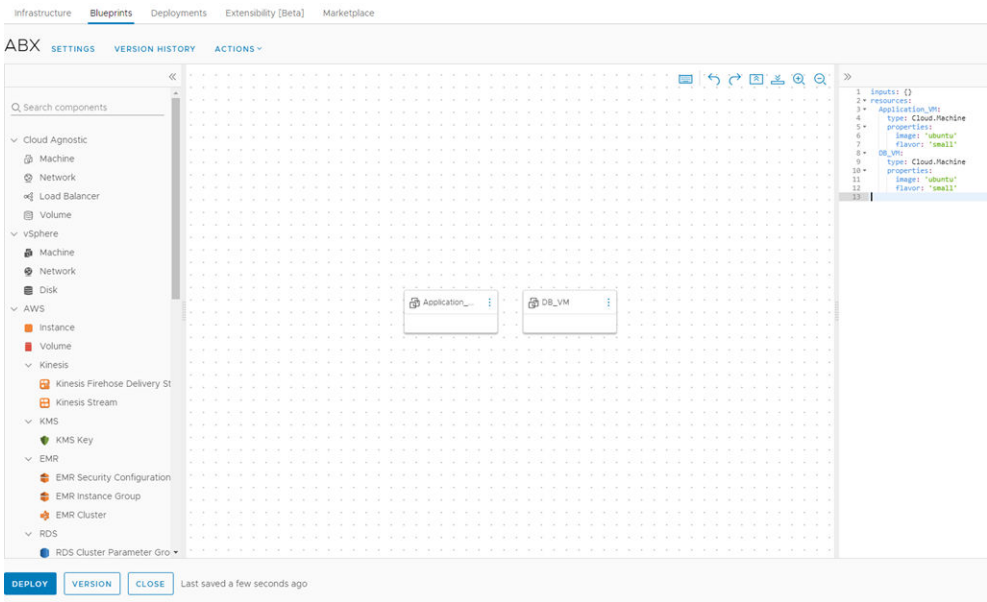
8 Click **New Subscription**.

9 Enter the following subscription details.

Detail	Setting
Event Topic	Select an event topic related to the tagging phase of the VM. For example, Compute Allocation. **Note** Tags must be part of the subscription schema.
Blocking	Set the timeout for the subscription to 1 minute.
Runnable Item Type	Select an extensibility action runnable type.
Runnable id	Select your custom extensibility action.

10 To save your custom extensibility action subscription, click **Create**.

11 Create a **New Blueprint** containing two virtual machines: Application VM and DB VM.



12 To deploy the VMs, click **Deploy**.

13 During deployment, verify that the event is initiated and the extensibility action is run.

14 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 vRealize Automation 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 on particular events provided by the Event Broker Service (EBS).

You can create these extensibility action scripts of code within vRealize Automation Cloud Assembly and assign them to subscriptions. Similarly to workflows, the extensibility action script triggers when an event included in an extensibility subscription occurs. Extensibility action scripts are used for more lightweight and simple automation of tasks and steps. They are also hosted on the cloud as opposed workflows which are hosted on-premises by using a vRealize Orchestrator server. For more information on integrating vRealize Automation Cloud Assembly with a vRealize Orchestrator server, see Configure 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 both Node.js and Python run-time environments and relies 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 vRealize Automation Cloud Assembly. For information on getting started with Amazon Web Services Lambda, see ABX: Serverless Extensibility of Cloud Assembly Services.

Note Extensibility actions are project-specific.

How do I create extensibility actions

With vRealize Automation 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.

  Note Writing user-defined code in the extensibility action editor might require an active Internet connection.

- 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 or Create a ZIP package for Node.js 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.

Procedure

1 Select **Extensibility > Library > Actions**.

2 Click **New Action**.

3 Enter a name for your action and select a project.

4 Click **Next**.

5 Search and select an action template.

Note To create a custom action without using an action template, select **Custom Script**.

New configurable parameters appear.

6 Select **Write script** or **Import package**.

7 Select an action **Runtime**.

8 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*.

9 Define the **Input** and **Output** parameters of the action.

10 (Optional) Add application dependencies to the action.

Note For actions imported from a ZIP package, application dependencies are added automatically.

11 To define timeout and memory limits, enable the **Set custom timeout and limits** option.

12 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**.

Export and import extensibility actions

With vRealize Automation 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.

      Alternate You can also import action scripts by selecting the **Import package** option directly from the action editor.

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 vRealize Automation 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 vRealize Automation 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 vRealize Automation Cloud Assembly for use in extensibility actions.

Furthermore, you can use a ZIP package in scenarios where modules associated with dependencies in your action script cannot be resolved by the vRealize Automation Cloud Assembly service, such as when your environment lacks Internet access.

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 vRealize Automation 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 vRealize Automation 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 vRealize Automation 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 vRealize Automation Cloud Assembly for use in extensibility actions.

Furthermore, you can use a ZIP package in scenarios where modules associated with dependencies in your action script cannot be resolved by the vRealize Automation Cloud Assembly service, such as when your environment lacks Internet access.

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.

   a. Create a `package.json` file with dependencies in your script folder. See Creating a package.json file and Specifying dependencies and devDependencies in a package.json file.

   b. Open a command-line shell.

   c. Navigate to the folder that you created for the action script and dependencies.

   ```
   cd /home/user1/zip-action
   ```

   d. Install your `package.json` file in the script folder by running the following command:

   ```
   npm install --production
   ```

   Note This command creates a `node_modules` directory in your folder.

4 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.

```
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.

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 **Extensiblity > 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**.

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 vRealize Automation Cloud Assembly extensibility subscriptions.

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 vRealize Automation Cloud Assembly extensibility subscriptions.

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.

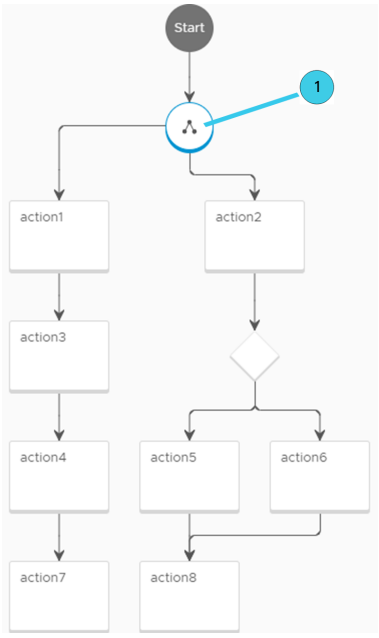
```
version: "1"
flow:
 flow_start:
  next: action1
action1:
  action: <action_name>
  next: action2
action2:
  action: <action_name>
  next: flow_end
```

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>
```



1 Fork Element

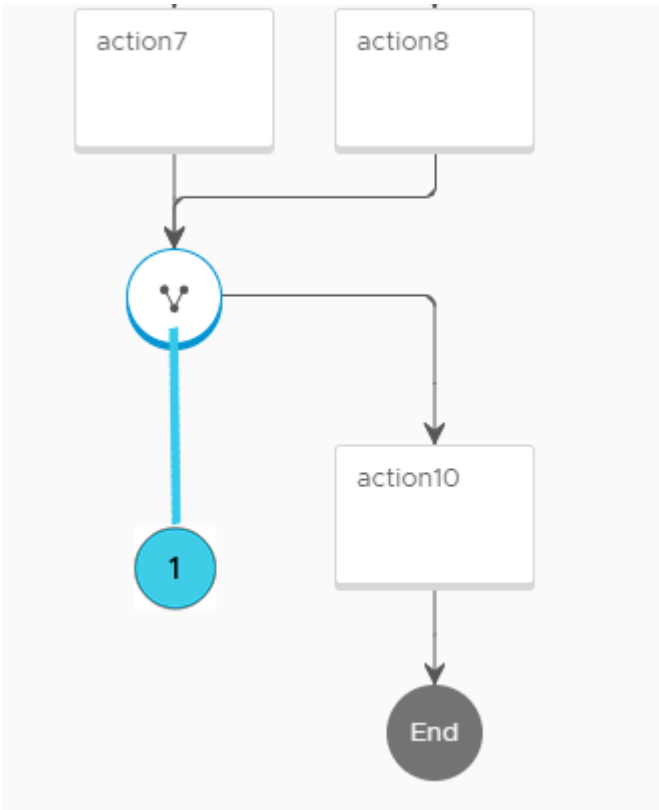
Note You can loop back to a previous action by assigning it as the `next:` action. For example, instead of `next: flow_end` to end your action flow, you can enter `next: action1` to rerun action1 and restart the sequence of actions.

Join action flows

Multiple extensibility action scripts or flows that join pathways together and contribute to the same output.

```
version: "1"
action7:
 action: <action_name>
 next: joinElement
action8:
 action: <action_name>
 next: joinElement
joinElement:
 join:
   type: all
   next: action10
action10:
 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.



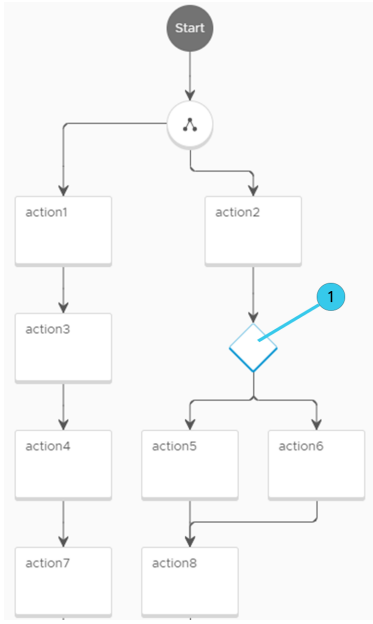
1 Join Element

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
action3:
   action: <action_name>
   next: action4
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>
```



1 Switch element

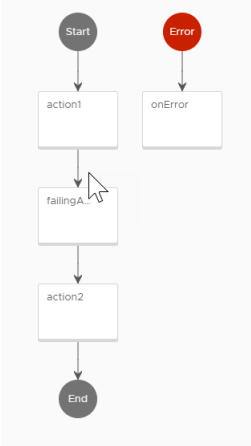
Note You can loop back to a previous action by assigning it as the `next:` action. For example, instead of `next: flow_end` to end your action flow, you can enter `next: action1` to rerun action1 and restart the sequence of actions.

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:

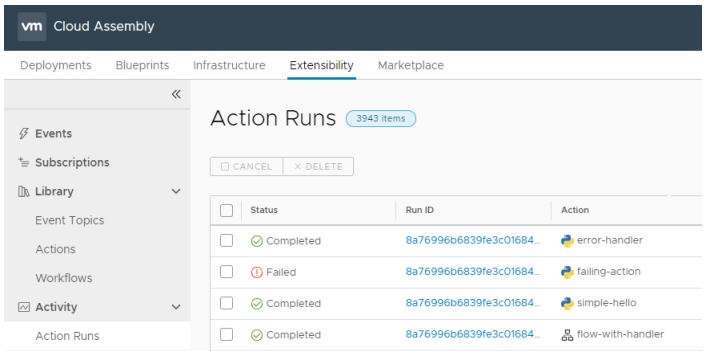
- Specified error message of the failed action.
- Action flow inputs.



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.

```
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**. Also, you can filter the list of action runs by one or more properties at once. To view additional details of an individual action run, click the Run ID.

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.

Problem	Solution
Deployment Failure	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.
Code Failure	These failures are a result of invalid scripts or code. Use the Action run logs to troubleshoot and correct the invalid scripts.

Extensibility workflow subscriptions

You can use your vRealize Orchestrator hosted workflows with vRealize Automation 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 subscription schema defines the format of the payload for Event Broker Service (EBS) messages. In order 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 **Extensiblity > Subscriptions**.

2 Click **New Subscription**.

3 Create a subscription with the following parameters:

Parameter	Value
Name	RenameVM
Event topic	Select an event topic suitable for the desired vRealize Orchestrator integration. For example, compute allocation.
Blocking/Non-blocking	Non-blocking
Runnable item	Select a vRealize Orchestrator runnable type.
Runnable ID	Select the desired workflow. For example, Set VM name.

4 To save your subscription, click **Create**.

5 Assign and activate your subscription by creating a blueprint or deploying an existing blueprint.

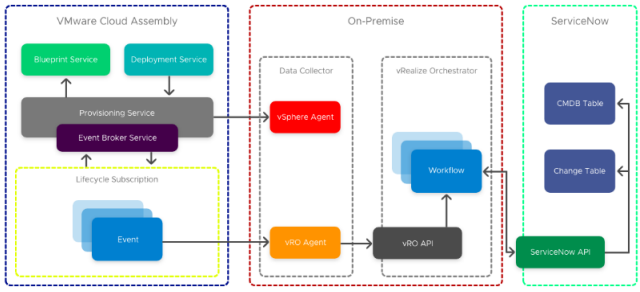
What to do next

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 vRealize Automation 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 vRealize Automation 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 vRealize Automation Cloud Assembly.

Note You can also integrate ServiceNow with vRealize Automation 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 CAS IaaS API with additional virtual machine properties outside of the payload.

Prerequisites

- A standalone or clustered vRealize Orchestrator environment.
- A vRealize Orchestrator integration in vRealize Automation Cloud Assembly. For information on integrating a standalone vRealize Orchestrator with vRealize Automation Cloud Assembly, see Configure 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 CAS API token in the same location, as the configuration file from Step 1.

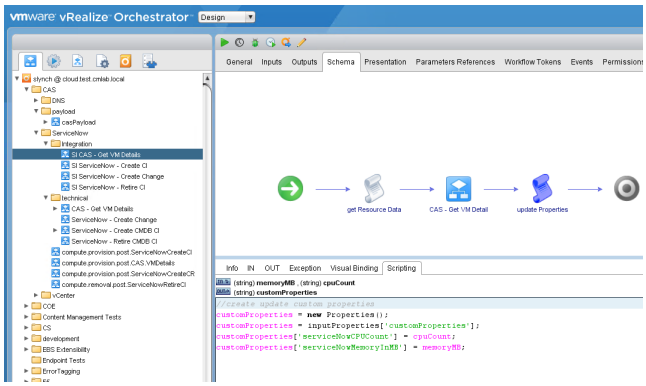
   Note The CAS API token has an expiration.
3. Create a workflow in vRealize Orchestrator with the provided script element. This script references and locates a REST Host. It also standardizes REST actions that use an optional parameter of a token, which is added as an extra authorization header.

```
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,attribute
Name)

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"
}

System.log("cpuCount: " + vm["customProperties"]["cpuCount"]);
```

```
System.log("memoryInMB: " + vm["customProperties"]["memoryInMB"]);

cpuCount =  vm["customProperties"]["cpuCount"];
memoryMB = vm["customProperties"]["memoryInMB"];
```

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.

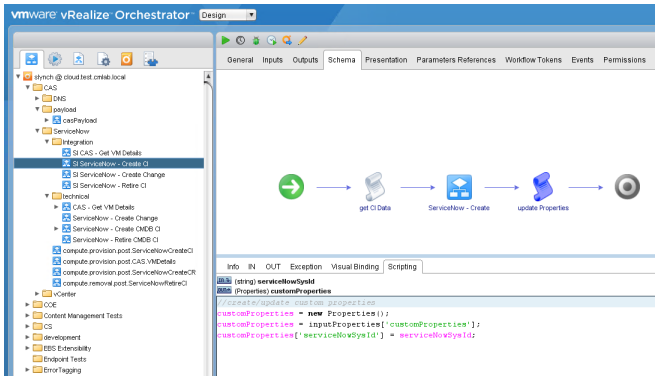


4 Add the ServiceNow CMDB Create CI script element to your workflow. This element locates the ServiceNow REST Host using the configuration item, creates a REST operation for the `cmdb_ci_vmware_instance` table, creates a string of content object based on workflow inputs for post data, and outputs the returned `sys_id`.

```
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,attribute
Name)

//REST Template
var opName = "serviceNowCreatCI";
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_vm_vmware 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

vRealize Automation Cloud Assembly is successfully integrated with ITSM ServiceNow. For more information on how you can use workflows to integrate ServiceNow in vRealize Automation Cloud Assembly, see Extending Cloud Assembly with vRealize Orchestrator for ServiceNow Integration.

Learn more about workflow subscriptions

By using an vRealize Orchestrator integration with vRealize Automation Cloud Assembly, you can extend the life cycles of applications with workflows.

vRealize Automation Cloud Assembly includes vRealize Orchestrator integration that you can use to import and link existing on-premises workflows to a subscription. These workflows are maintained on the vRealize Orchestrator server. You can create, modify, and delete workflows only using the vRealize Orchestrator client.

Best practices for creating vRealize Orchestrator workflows

A workflow subscription is based on a specific topic schema. 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 reply schema.

How do I track workflow runs

The **Workflow Runs** tab shows you a log of subscription triggered workflows and their status.

You can view the log of workflow runs using **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.

Problem	Cause	Solution
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 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 blueprint and deployments through the specified project.

Extensibility terminology

As you work with Extensibility and subscriptions within vRealize Automation Cloud Assembly, you might encounter some terminology that is specific to the subscriptions and event broker service.

Table 6-4. Extensibility Terminology

Term	Description
Event Topic	Describes a set of events that have 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.
Event	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.
Event Broker Service	The service that dispatches messages published by a producer to the subscribed consumers.
Payload	The event data that contains all the relevant properties related to that Event Topic.
Subscription	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.
Subscriber	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.
System Administrator	A user with privileges to create, read, update, and delete tenant workflow subscriptions and system workflow subscriptions using vRealize Automation Cloud Assembly.
Workflow Subscription	Specifies the event topic and conditions that trigger a vRealize Orchestrator workflow.
Action Subscription	Specifies the event topic and conditions that trigger an extensibility action to run.
Workflow	A vRealize Orchestrator workflow that is integrated within vRealize Automation Cloud Assembly. You can link these workflows to events within subscriptions.
Extensibility Action	A streamlined script of code that can run after a triggering event within a subscription. Extensibility actions are similar to workflows, but are more lightweight. Extensibility actions can customized from within vRealize Automation Cloud Assembly.
Action Runs	Accessible through the **Action Runs** tab. An action run is a detailed log of extensibility actions that have run in response to triggering events.

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.

vRealize Automation 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 vRealize Automation Cloud Assembly

vRealize Automation 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 vRealize Automation 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 6-5. Cloud Assembly Event Topics

Event Topic	Blockable	Description
`Blueprint.configuration`	No	Issued when a blueprint configuration event like the creation or deletion of a blueprint occurs, and can be useful for notifying external systems of such events.
`Blueprint.version.configuration`	No	Issued when a new blueprint versioning event occurs, such as the creation, release, de-release, or restore of a version. This event topic can be useful with integrations of third-party version control systems.
`Compute allocation`	Yes	Events issued before the allocation of `resourcenames` and `hostselections`. Both of these properties can be modified at this stage.
`Compute post provision`	Yes	Events issued after a resource was provisioned successfully.
`Compute post removal`	Yes	Events issued after a compute resource was removed.
`Compute provision`	Yes	Events that occur before the resource is provisioned at the hypervisor. **Note** You can change the allocated IP address.
`Compute removal`	Yes	Events issued before the resource is removed.

Table 6-5. Cloud Assembly Event Topics (continued)

Event Topic	Blockable	Description
Compute reservation	Yes	Events issued at the time of reservation. **Note** You can change the placement order.
Deployment action completed	Yes	Issued after a deployment action is finished.
Deployment action requested	Yes	Issued before a deployment action is competed.
Deployment completed	Yes	Issued after the deployment of a blueprint or catalog request.
Deployment onboarded	No	Issued when a new deployment is on-boarded.
Deployment requested	Yes	Issued before the deployment of a blueprint or catalog request.
Deployment resource action completed	Yes	Issued after the deployment of a resource action.
Deployment resource action requested	Yes	Issued before the deployment of a resource action.
Deployment resource completed	Yes	Issued after the provisioning of a deployment resource.
Deployment resource requested	Yes	Issued before the provisioning of a deployment resource.
Disk allocation	Yes	Issued for the preallocation of disk resources.
Disk post removal	Yes	Issued after a disk resource is deleted.
Disk post resize	Yes	Issued after a disk resource is resized.
EventLog	Yes	Logging related events.
Kubernetes cluster allocation	Yes	Issued for the preallocation of resources for a Kubernetes cluster.
Kubernetes cluster post provision	Yes	Issued after a Kubernetes cluster is provisioned.
Kubernetes cluster post removal	Yes	Issued after a Kubernetes cluster is deleted.
Kubernetes cluster provision	Yes	Issued before a Kubernetes cluster is provisioned.
Kubernetes cluster removal	Yes	Issued before the process of deleting a Kubernetes cluster is initiated.
Load balancer post provision	Yes	Issued after the provisioning of a load balancer.

Table 6-5. Cloud Assembly Event Topics (continued)

Event Topic	Blockable	Description
`Network Configure`	Yes	Events issued when the network is configured during compute allocation. **Note** The Network Configure topic supports multiple IP addresses/NICs.
`Project Lifecycle`	No	Events issued when a project is created, updated, or deleted.

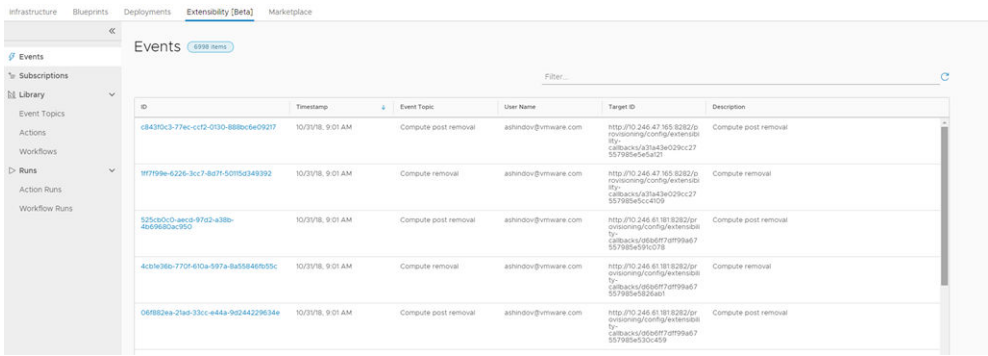
Event Schema

After you add an event topic, you can view the event schema. This schema defines the structure of the event's payload, or `inputProperties`.

Extensibility event log

The extensibility events tab shows you a list of all events that have occurred within your environment.

You can view the extensibility event log using **Extensibility > Events**. Also, you can filter the list of events by one or more properties at once. To view additional details of an individual event, click the event's ID.



Create an extensibility subscription

Using vRealize Orchestrator integration or extensibility actions with vRealize Automation 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/action only triggers if the boolean expression is 'true'. This is helpful when you want to control when events and actions are triggered.

Tip In the filter events in topic text box, use the "Alt + Space" keys on Windows or "Option + Space" keys on Mac to display filter options.

Prerequisites

- Cloud administrator user role
- If using vRealize Orchestrator workflows:
  - The library of the embedded vRealize Orchestrator Client or the library of any integrated external vRealize Orchestrator instance.
- If 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 Integration**.
3. Enter the details of your subscription.
4. Select an **Event Topic**.
5. (Optional) Set conditions for the event topic.
6. (Optional) If applicable, configure the blocking behavior for the event topic.
7. Click **Runnable item** and select **vRO Workflow** or **ABX Action** from the drop-down menu.
8. Select a workflow or extensibility action that you want to run in your subscription.
9. (Optional) To define the project scope of the extensibility subscription, disable **Any Project**, and click **Add Projects**.
10. Click **Create** save your subscription.

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 blueprint to link and use the subscription. Also, you can verify the status of the workflow run in the **Extensibility** tab within vRealize Automation 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 checking available inputs for your vRealize Orchestrator workflow. To view all parameters of your virtual machine, use the following script with your workflow:

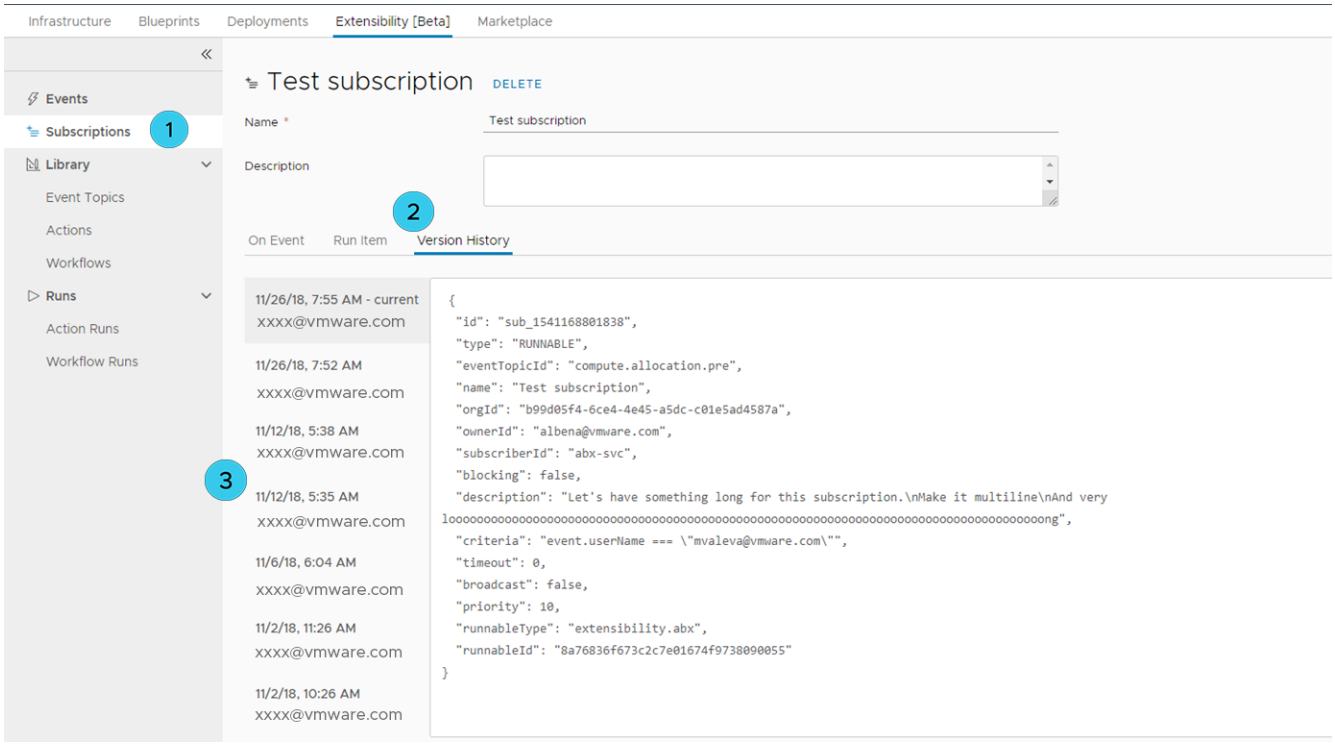
```
function dumpProperties(props,lvl){
         var keys = props.keys;
         var prefix = ""
     for (var i=0; i<lvl; i++){
                  prefix = prefix + "";
          }
          for (k in keys){
                 var key = keys[k];
                 var value = props.get(keys[k])
                 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 can show you the change history of your subscription with the user and date of the change. If your subscription fails or is running incorrectly, the version history can help identify the cause.



1

Open your subscription from the **Subscriptions** tab.

2

To view the version history, click **Version History**.

3

You can click each change entry to view the corresponding subscription code associated with the change.

Managing vRealize Automation Cloud Assembly deployments

7

As a vRealize Automation Cloud Assembly blueprint developer, you use the Deployment tab to manage your deployments. You can troubleshoot failed provisioning processes, make changes, and destroy unused deployments.

Deployments are the provisioned instances of blueprints. The Deployments tab displays your successful and failed deployments. You use the page to manage your successful deployments or to begin troubleshooting any failed requests.

Working with deployment cards

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.

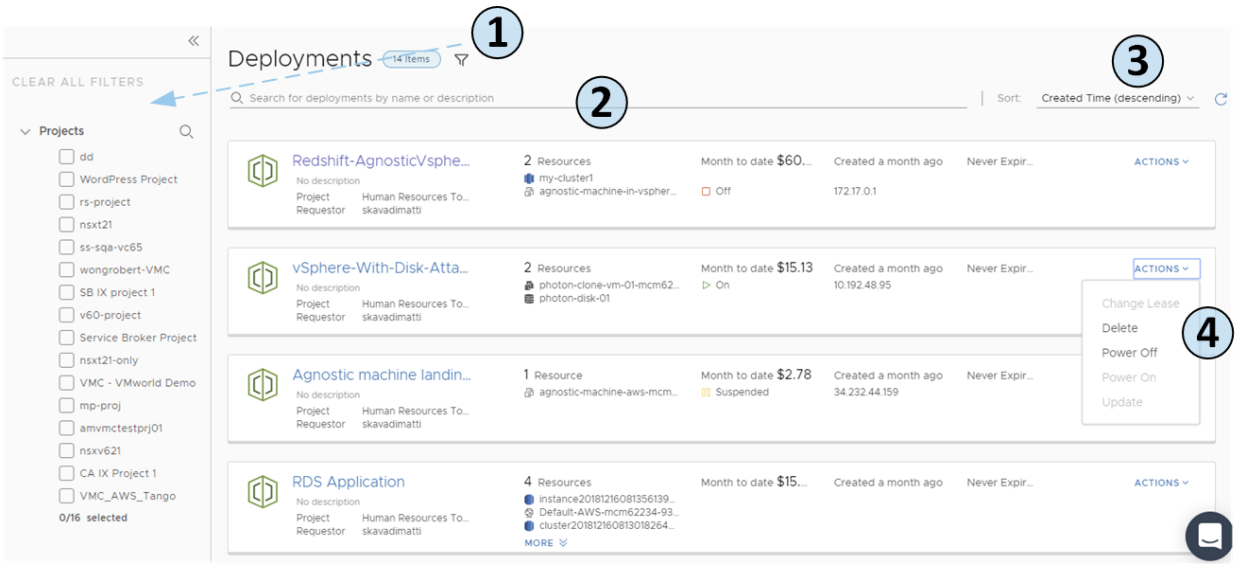
1 Filter your requests based on attributes.

2 Search for deployments based on keywords or requestor.

3 Sort the list to order by time or name.

4 Run deployment-level actions on the deployment, including deleting unused deployments to reclaim resources.

You can also see deployment costs, expiration dates, and status.



This chapter includes the following topics:

- How do I monitor active deployments in vRealize Automation Cloud Assembly
- What can I do if a vRealize Automation Cloud Assembly deployment fails
- How do I manage the life cycle of a completed vRealize Automation Cloud Assembly deployment
- What actions can I run on vRealize Automation Cloud Assembly deployments

How do I monitor active deployments in vRealize Automation Cloud Assembly

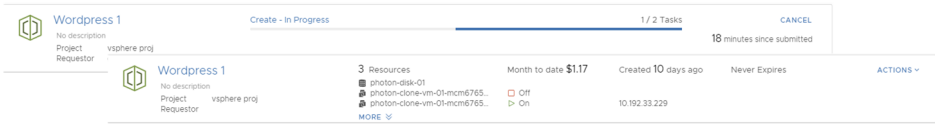
After you deploy a vRealize Automation Cloud Assembly blueprint, you can monitor your request to ensure that the resources are provisioned and running. Beginning with the deployment card, you can verify the provisioning of your resources. Next, you can examine the deployment details.

Procedure

1 Click **Deployments** and locate your in-process deployment card using the filter and search, if needed.

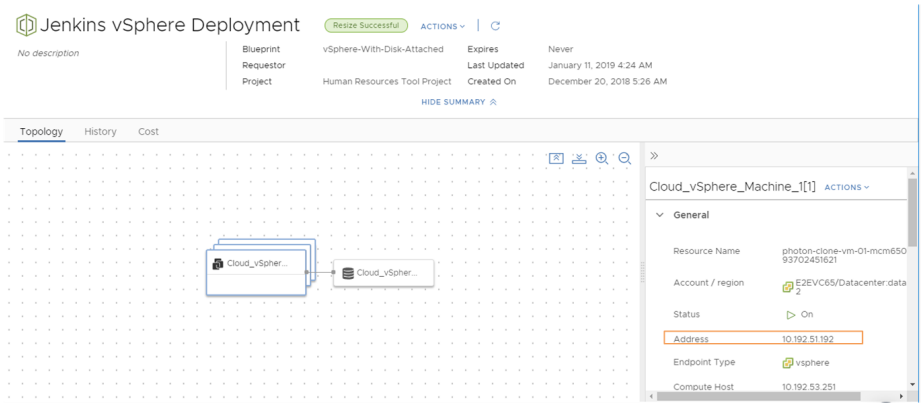
2 Review the card status.

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.



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 vRealize Automation Cloud Assembly deployment.
- If your deployment fails, see What can I do if a vRealize Automation Cloud Assembly deployment fails.

What can I do if a vRealize Automation Cloud Assembly deployment fails

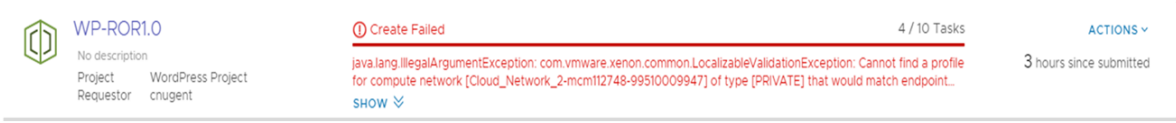
Your deployment request might fail for many reasons. It might be due to network traffic, a lack of resources on the target cloud provider, or a flawed deployment specification. Or, the deployment succeeded, but it does not appear to be working. You can use vRealize Automation Cloud Assembly to examine 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 vRealize Automation Cloud Assembly.

Procedure

1 To determine if a request failed, click the **Deployments** tab and locate the deployment card.

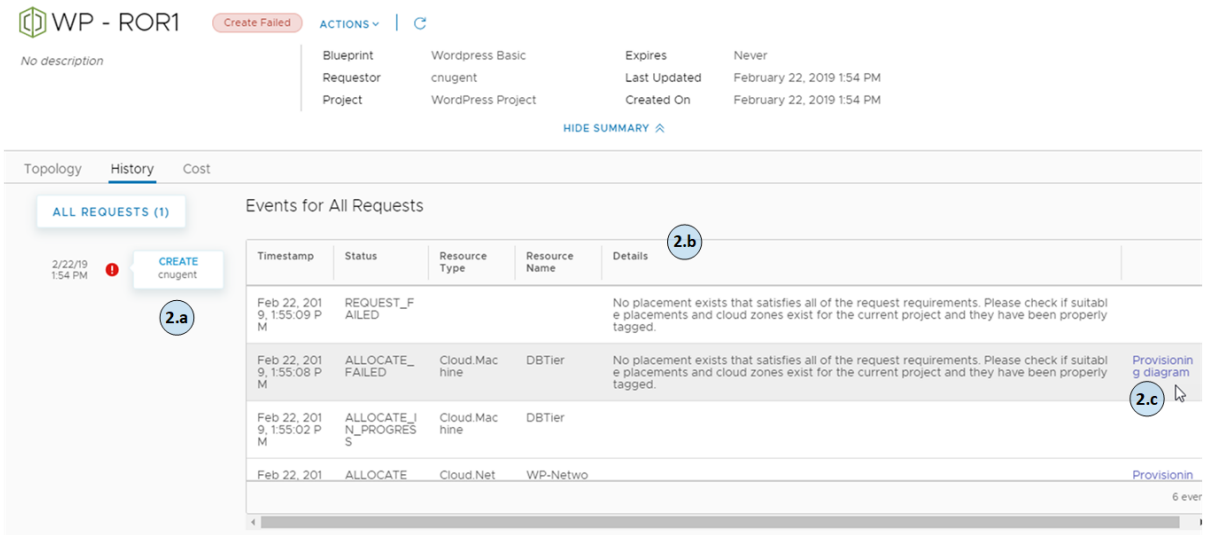


Failed deployments are indicated on the card.

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.



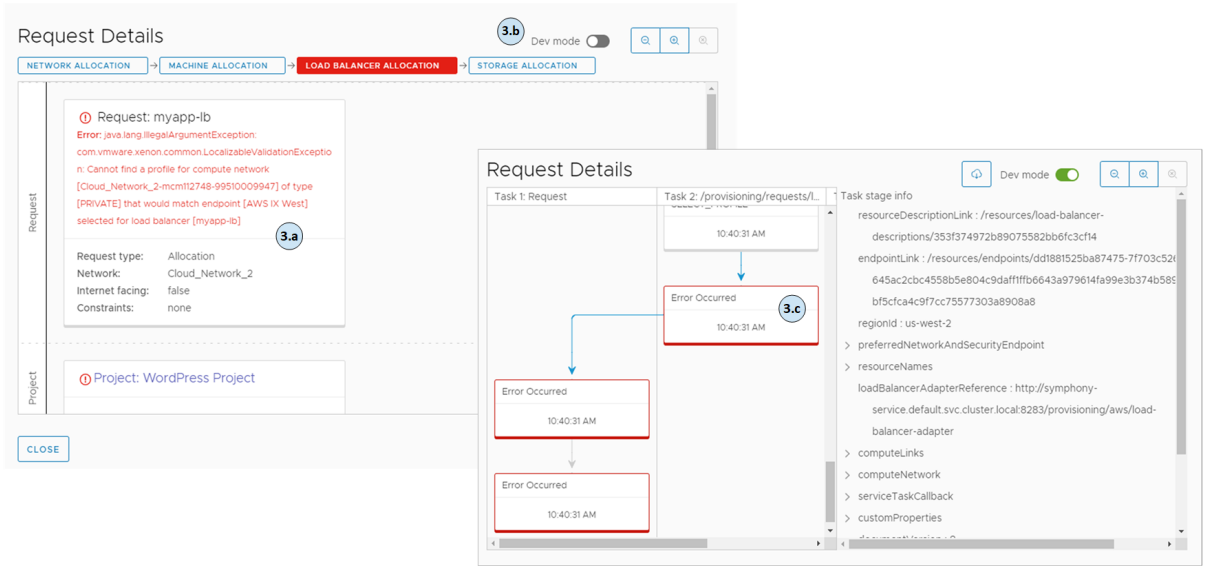
a Review the event tree to see where the provisioning process failed. This tree is useful when you modify a deployment, but the change fails.

The tree also shows when you run deployment actions. You can use the tree troubleshoot failed changes.

b The **Details** provides a more verbose version of the error message.

c If the requested item was a vRealize Automation Cloud Assembly blueprint, the link to the right of the message opens vRealize Automation 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.



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.

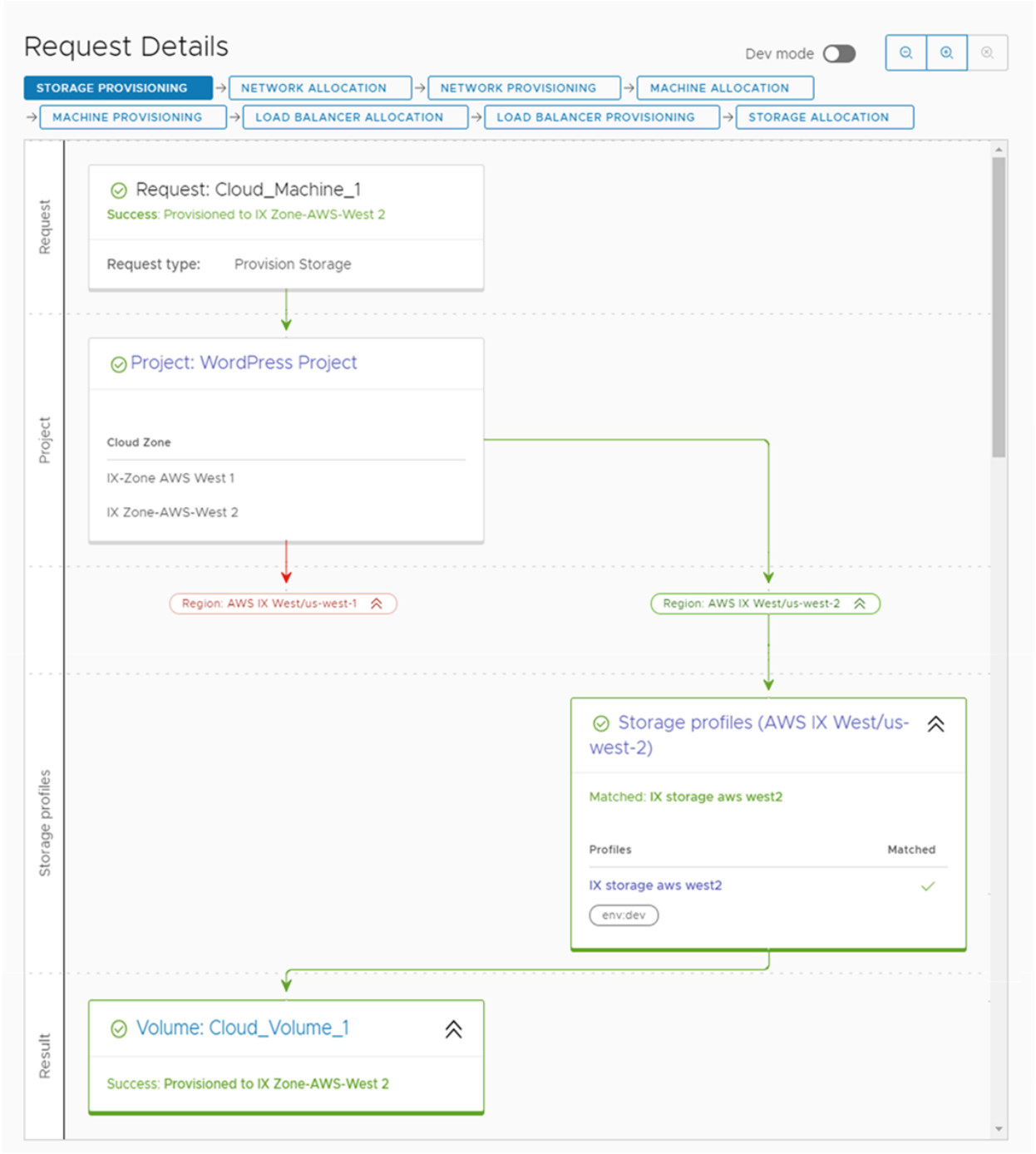
c Click the card to review the deployment script.

4 Resolve the errors and redeploy the blueprint.

The errors might be in the blueprint construction or they might be related to how your infrastructure is configured.

What to do next

When the errors are resolved and the blueprint 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 vRealize Automation 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

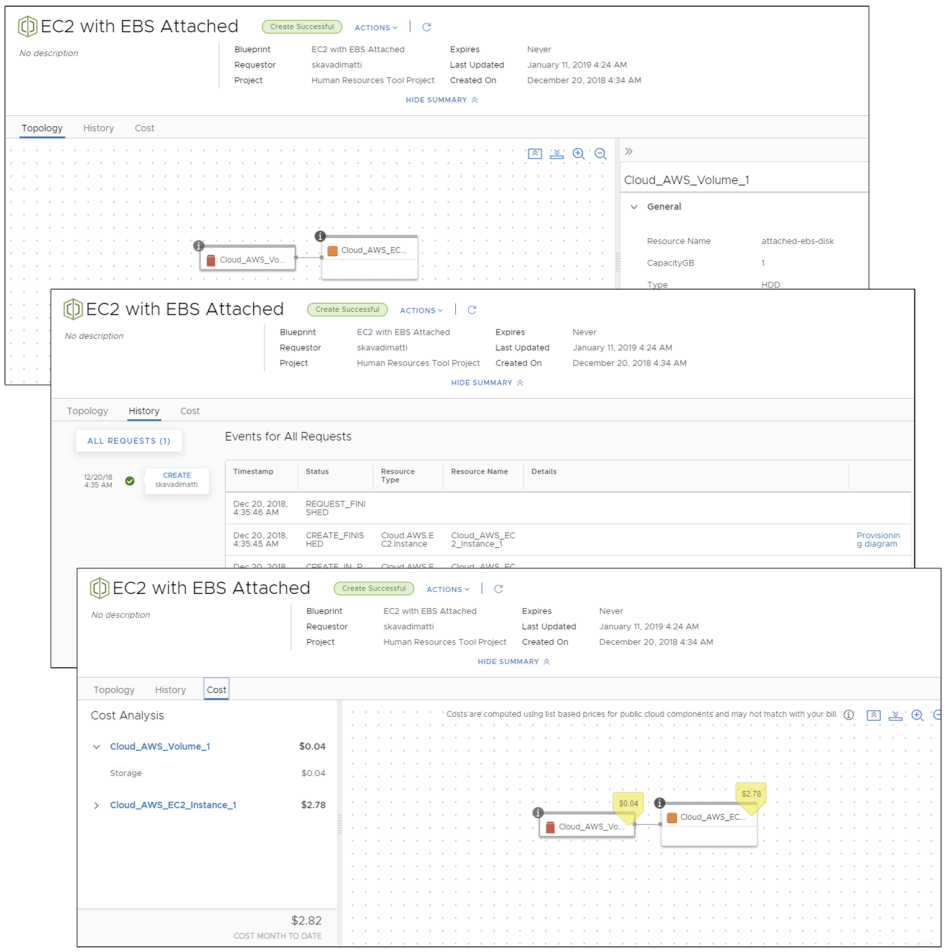
1 Click **Deployments** and locate your deployment.

2 To access the deployment details, click the deployment name.

You can use the Topology tab to visualize the deployment structure and resources.

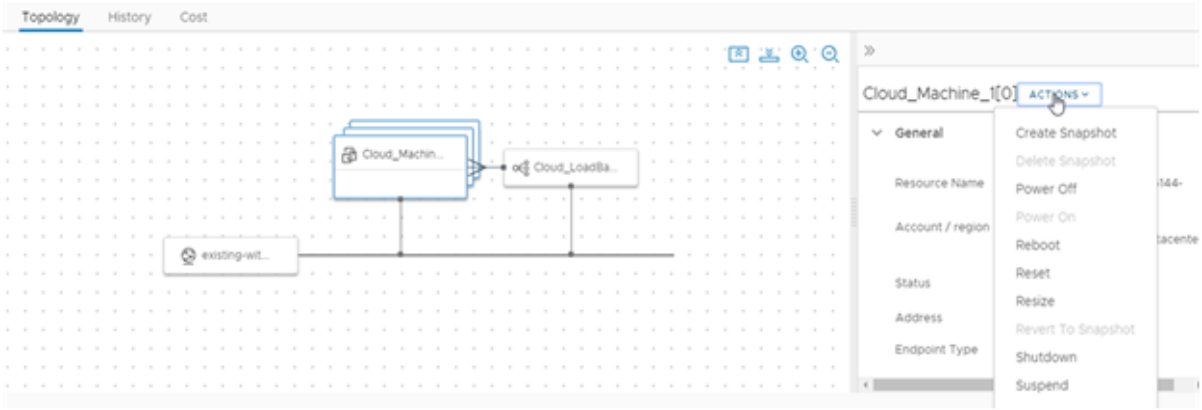
The History tab includes all the provisioning events and any events related to actions that you run after the requested item is deployed. If there are any problems with the provisioning process, the History tab events will help you with troubleshooting the failures.

The Cost tab provides the current cost of some components since they were deployed.



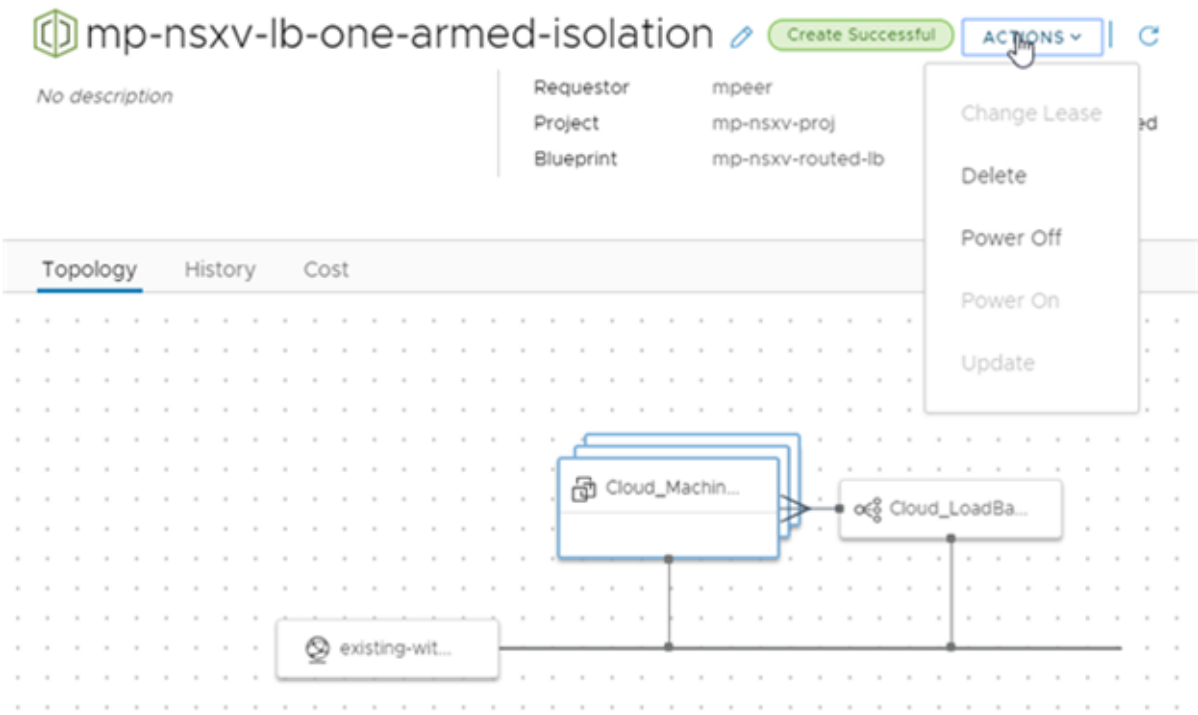
3 If you determine that a deployment is too costly in its current configuration and you want to resize a component, select the component on the topology page and then select **Actions > Resize** on the component page.

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.



What to do next

To learn more about possible actions, see What actions can I run on vRealize Automation Cloud Assembly deployments.

What actions can I run on vRealize Automation Cloud Assembly deployments

After you deploy blueprints, you can run actions in vRealize Automation 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 vRealize Automation Service Broker. See How do I entitle consumers to Service Broker day 2 action policies

Table 7-1. List of possible actions

Action	Applies to these resource types	For these cloud accounts or integrations	Description
Add Disk	Machines	■ Amazon Web Service ■ Google Cloud Platform ■ Microsoft Azure ■ VMware vSphere	Add additional disks to existing virtual machines.
Change Lease	Deployments	■ Amazon Web Service ■ Microsoft Azure ■ VMware vSphere	Change the lease expiration date and time. When a lease expires, the deployment is destroyed and the resources are reclaimed. Lease policies are set in vRealize Automation Service Broker.
Connect to Remote Console	Machines	■ VMware vSphere	Open a remote session on the selected machine. Review the following requirements for a successful connection. ■ As a deployment consumer, verify that the provisioned machine is powered on.
Create Snapshot	Machines	■ Google Cloud Platform ■ VMware vSphere	Create a snapshot of the virtual machine. If you are allowed only two snapshots in vSphere and you already have them, this command is not available until you delete a snapshot.
Delete	Deployments	■ Amazon Web Service ■ Google Cloud Platform ■ Microsoft Azure ■ VMware vSphere	Destroy a deployment. All the resources are deleted and the reclaimed. 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.

Table 7-1. List of possible actions (continued)

Action	Applies to these resource types	For these cloud accounts or integrations	Description
	Machines and load balancers	■ Amazon Web Service ■ Microsoft Azure ■ VMware vSphere	Delete a machine or load balancer from a deployment. This action might result in an unusable deployment.
Delete Snapshot	Machines	■ VMware vSphere ■ Google Cloud Platform	Delete a snapshot of the virtual machine.
Edit Tags	Deployments	■ Amazon Web Service ■ Microsoft Azure ■ VMware vSphere	Add or modify resource tags that are applied to individual deployment resources.
Power Off	Deployments	■ Amazon Web Service ■ Microsoft Azure ■ VMware vSphere	Power off the deployment without shutting down the guest operating systems.
	Machines	■ Amazon Web Service ■ Google Cloud Platform ■ Microsoft Azure ■ VMware vSphere	Power off the machine without shutting down the guest operating systems.
Power On	Deployments	■ Amazon Web Service ■ Microsoft Azure ■ VMware vSphere	Power on the deployment. If the resources were suspended, normal operation resumes from the point at which they were suspended.
	Machines	■ Amazon Web Service ■ Google Cloud Platform ■ Microsoft Azure ■ VMware vSphere	Power on the machine. If the machine was suspended, normal operation resumes from the point at which the machine was suspended.
Reboot	Machines	■ Amazon Web Service ■ VMware vSphere	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.
Reconfig ure	Load Balancers	■ Amazon Web Service ■ Microsoft Azure ■ Google Cloud Platform ■ VMware vSphere	Change the load balancer protocol, port, health configuration, and member pool settings.
Remove Disk	Machines	■ Amazon Web Service ■ Google Cloud Platform ■ Microsoft Azure ■ VMware vSphere	Remove disks from existing virtual machines.
Reset	Machines	■ Amazon Web Service ■ Google Cloud Platform ■ VMware vSphere	Force a virtual machine restart without shutting down the guest operating system.

Table 7-1. List of possible actions (continued)

Action	Applies to these resource types	For these cloud accounts or integrations	Description
Resize	Machines	■ Amazon Web Service ■ Microsoft Azure ■ Google Cloud Platform ■ VMware vSphere	Increase or decrease the CPU and memory of a virtual machine.
Resize Boot Disk	Machines	■ Amazon Web Service ■ Google Cloud Platform ■ Microsoft Azure ■ VMware vSphere	Increase or decrease the size of your boot disk medium.
Resize Disk	Storage disk	■ Amazon Web Service ■ Google Cloud Platform	Increase the capacity of a storage disk.
Restart	Machines	■ Microsoft Azure	Shut down and restart a running machine.
Revert to Snapshot	Machines	■ Google Cloud Platform ■ VMware vSphere	Revert to a previous snapshot of the machine. You must have an existing snapshot to use this action.
Run Puppet Task	Managed resources	■ Puppet Enterprise	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.
Shutdown	Machines	■ VMware vSphere	Shut down the guest operating system and power off the machine. VMware Tools must be installed on the machine to use this action.
Suspend	Machines	■ Microsoft Azure ■ VMware vSphere	Pause the machine so that it cannot be used and does not consume any system resources other than the storage it is using.
Update	Deployments	■ Amazon Web Service ■ Microsoft Azure ■ VMware vSphere	Change the deployment based on the input parameters.
Update Tags	Machines and disks	■ Amazon Web Service ■ Microsoft Azure ■ VMware vSphere	Add, modify, or delete a tag that is applied to an individual resource.