Using vRealize Code Stream

vRealize Code Stream 1.0

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

https://docs.vmware.com/

If you have comments about this documentation, submit your feedback to

docfeedback@vmware.com

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

Contents

Using vRealize Code Stream

Using vRealize Code Stream provides information about how to install and configure VMware vRealize Code Stream to automate the release of applications, frequently while leveraging existing tools within the build, development, test, provisioning, and monitoring environments.

Intended Audience

This information is intended for anyone who wants to install and automate the release of applications in various development environments. The information is written for experienced developers and operation teams who are familiar with release automation.

VMware Technical Publications Glossary

VMware Technical Publications provides a glossary of terms that might be unfamiliar to you. For definitions of terms as they are used in VMware technical documentation, go to http://www.vmware.com/support/pubs.

Introducing vRealize Code Stream

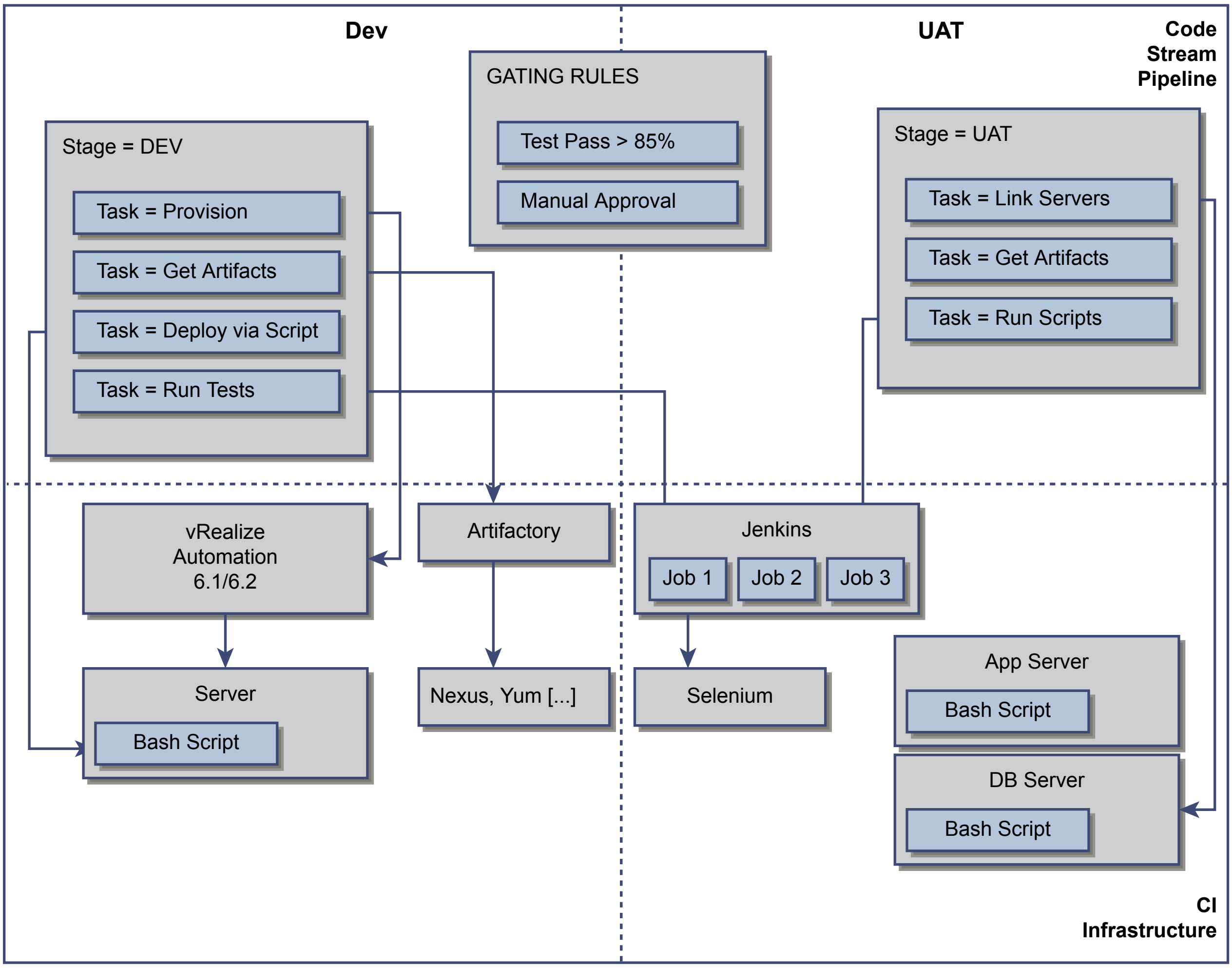
1

vRealize Code Stream automates the software release process by modeling all of the necessary tasks in pipeline templates.

A release pipeline is a sequence of stages, each stage is composed of multiple tasks and environments that the software has to complete before it is released into production. The stages can include development, functional testing, user acceptance test (UAT), load testing (LT), systems integration testing (SIT), staging, and production. Release managers, typically Build and Release Engineers, model pipeline templates.

Each stage in a pipeline includes a set of activities such as provisioning a machine, retrieving an artifact, deploying software, executing a test, creating a manual task, or executing a custom workflow or script. The software changes are promoted to the next stage in the pipeline when they satisfy a set of rules called gating rules. The gating rules include testing rules and compliance rules. Gating rules that are associated with a pipeline are specific to an organization or an application. Users can define gating rules when a pipeline template is created. The environments do not need to be aware of these gating rules.

Figure 1‑1. Main Components of vRealize Code Stream



This chapter includes the following topics:

- Core Architectural Principles
- Roles and Responsibilities of Personas
- Integrating vRealize Code Stream with External Systems
- Key Release Automation Concepts

Core Architectural Principles

vRealize Code Stream works with deployment engines such as vRealize Automation (formerly VMware vCloud Automation Center), vCenter Server, Chef, Puppet, or scripts. vRealize Automation can also integrate with continuous integration frameworks and testing frameworks.

vRealize Code Stream includes an approval engine that can integrate with IT service management products and various commercial or custom approval systems. Based on the type of integration, vRealize Code Stream uses the JFrog Artifactory or VMware vCloud Orchestrator (vRO) (formerly vCO) plug-ins for extensibility. Both of the approval and extensibility components are embedded in the vRealize Automation virtual appliance.

For the supported vRealize Code Stream integrations, see Integrating vRealize Code Stream with External Systems.

Deployment Engines

vRealize Code Stream integrates with a number of provisioning and deployment solutions including vRealize Automation. It can also trigger scripts or vRealize Orchestrator workflows. Support for other provisioning solutions is delivered by plug-ins that VMware, partners, or users publish.

Testing Frameworks

vRealize Code Stream integrates with Jenkins to trigger Jenkins jobs, including test routines. The user interface for test automation and for extensibility uses the vRealize Orchestrator plug-in framework.

A Jenkins job can run test cases that are configured for an application. The Test Acceptance Threshold workflow in the gating rule verifies the results of the Jenkins job and returns the response to the vRealize Code Stream server. Based on the results of the test and the gating rules that have been defined, the build either proceeds to the next stage of the release pipeline or it fails.

Approval Systems

vRealize Code Stream uses vRealize Orchestrator plug-ins for integration with approval systems. Manual approval tasks can be created within the vRealize Automation inbox, but vRealize Code Stream can also integrate with BMC Remedy ITSM, HP Service Manager, ServiceNow, and other ticketing systems. The approval systems integration requires downloading and installing the appropriate vRealize Orchestrator plug-in from the VMware Solution Exchange.

Roles and Responsibilities of Personas

A tenant administrator can assign the release manager, release engineer, and the release dashboard user roles, which are an integral part of release automation.

These roles have various responsibilities when they interact with the product. See Configuring Additional Tenants.

Each stage in the release pipeline defines a set of activities such as initialize, deploy, test, approval, custom tasks, and troubleshoot. The following table lists the roles and responsibilities of the personas.

Table 1-1. Roles and Responsibilities in vRealize Code Stream

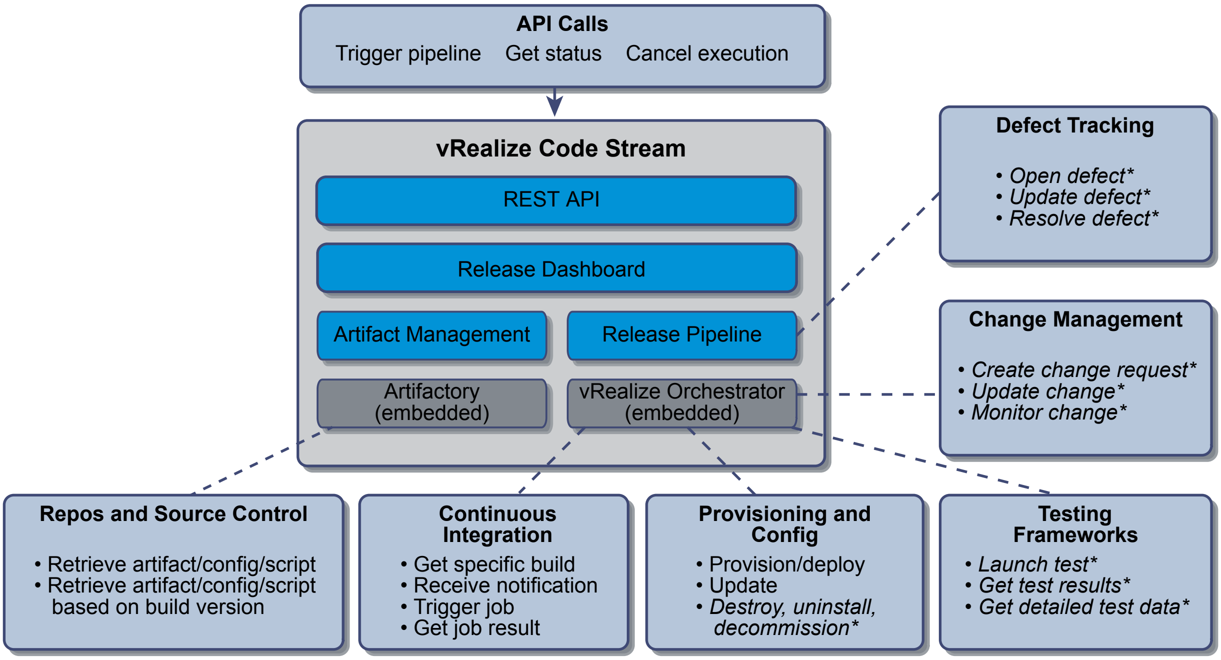
Role	Responsibility
Release Manager	■ Define and create a pipeline template ■ Define different stages in a pipeline template ■ Define access permissions for release engineers ■ Define gating rules ■ Register repositories ■ Register CI and test frameworks ■ Approve promotion to different stages
Release Engineer	■ Retrieve artifacts ■ Deploy artifacts to an environment ■ Trigger a pipeline ■ Monitor pipeline execution ■ View test results ■ Trigger tests ■ Review test results
Release Dashboard User	■ View the artifact, machine, and pipeline data on the dashboard ■ Troubleshoot a failed pipeline

Integrating vRealize Code Stream with External Systems

vRealize Code Stream includes an extensibility framework that supports modular integrations with external systems, without changing the core platform.

Based on the type of external system, different mechanisms are recommended.

Figure 1-2. Supported Integration with External Systems



*Items in italics require custom workflows or scripts

Release Pipeline Integrations

Release pipeline templates support various tasks that can trigger actions in a wide category of systems such as continuous integration, testing frameworks, or defect tracking systems.

You can download vRealize Orchestrator plug-ins from the VMware Solution Exchange. The Artifactory plug-ins are available on the JFrog Web site.

Some integrations are supported natively. Some integrations require downloading an Artifactory or vRealize Orchestrator plug-in. Others require creating custom workflows by using a vRealize Orchestrator plug-in protocol such as HTTP-REST, SOAP, or SSH.

Table 1-2. Supported Integrations

System Category	Integration Mechanism	Native Support	Requires Separate Plug-in	Requires Custom Workflows
Repository	JFrog Artifactory	■ Sonatype Nexus ■ Yum ■ HTTP-browsable repository system	N/A	
Continuous Integration	vRealize Orchestrator	Jenkins	N/A	■ Atlassian Bamboo using the REST plug-in ■ JetBrains TeamCity using the REST plug-in ■ Microsoft Team Foundation Server using the REST plug-in
Provisioning and configuration management	vRealize Orchestrator	■ vRealize Automation (single-machine blueprints) ■ Scripts (BASH or Windows PowerShell)	■ vCenter Server ■ VMware vCloud Director	■ Chef using the REST plug-in ■ Puppet using the REST plug-in
Testing frameworks	vRealize Orchestrator	Any testing frameworks exposed as Jenkins jobs	N/A	Any testing framework that offers a REST or SOAP API
Change management systems	vRealize Orchestrator	N/A	■ BMC Remedy ITSM ■ HP Service Manager ■ ServiceNow	Any commercial or custom change management system that offers a REST or SOAP API
Defect tracking systems	vRealize Orchestrator	N/A	N/A	■ Atlassian JIRA using the REST plug-in ■ Bugzilla using the REST plug-in

Key Release Automation Concepts

Use the following definitions to help you understand the release pipeline modeling and the artifact management workflow.

artifact

A script or the output of a build process. The script can be deployed or upgraded in a given stage.

Artifact types can be configuration files, application bits, or third-party software.

artifact management

A service that manages the artifacts over a range of local and remote repositories.

For example, managing a WAR file stored in the Maven repository.

category

A task type.

Some supported categories are Provision, Custom, Artifact, Deploy, and Test. A task belongs to a provider and a category.

gating rule

A set of rules that must be completed before the software changes are promoted and the next set of tasks starts in the subsequent stage.

The gating rules include testing rules and compliance rules. Gating rules that are associated with a pipeline are specific to an organization and applications.

instance

A vRealize Orchestrator plug-in scenario that captures specific configurations of a provider.

The instance is created by using a vRealize Orchestrator client.

pipeline

A collection of all the stages or environments in which a software change has to pass through independently before it is released into production.

For example, development, test, user acceptance test, load test, staging, and production.

provider

Plug-in vendors that support the categories.

For example, the Provision category is supported by vRealize Automation and vCenter Server providers.

stage

Every stage in the pipeline defines a set of activities.

For example, deploy, test, approval through gating rules, and custom tasks.

task

An activity in a given stage.

For example, provision the machines, resolve the artifact, deploy the artifact, run the test, and so on.

Opening the port in a firewall is a manual task.

vRealize Code Stream Installation

2

vRealize Code Stream shares a platform and common services with vRealize Automation 6.2.

To install vRealize Code Stream, you must set up, configure, and deploy a vRealize Automation appliance. You also need to configure a tenant to assign user roles in vRealize Code Stream.

During installation, you can apply the vRealize Code Stream license to enable the features in the vRealize Automation appliance. To use the vRealize Code Stream function, a system administrator can deploy or upgrade a vRealize Automation 6.2 appliance and apply a vRealize Code Stream license key.

This chapter includes the following topics:

Installation Worksheets

You can use these worksheets to record important information for reference during the installation process.

One copy of each worksheet is given here. Create additional copies as you need them. Settings are case sensitive.

Table 2‑1. PostgreSQL Database Information

Variable	Value	Example
Host Name (FQDN)		vcac-database-va.mycompany.com
IP		192.168.1.111
Database name	vcac (default)	vcac
Database username	vcac (default)	vcac
Database password	vcac (default)	vcac
Appliance username	administrator@vsphere.local (default)	administrator@vsphere.local
Appliance password		vmware

Table 2‑2. Identity Appliance Information

Variable	Value	Example
Host Name (FQDN)		vcac-sso.mycompany.com
SSO service over HTTPS Incoming Port	7444 (do not change)	7444
IP		192.168.1.104
Username	administrator@vsphere.local (default)	administrator@vsphere.local
Password		vmware

Table 2‑3. Leading cluster vRealize Appliance Information

Variable	Value	Example
Host Name (FQDN)		vcac-va.mycompany.com
SSO service over HTTPS Outgoing Port (default)	7444 (do not change)	7444
IP		192.168.1.105
Username	administrator@vsphere.local (default)	administrator@vsphere.local
Password		vmware

Table 2‑4. Additional vRealize Appliance Information

Variable	Value	Example
Host Name (FQDN)		vcac-va2.mycompany.com
SSO service over HTTPS Outgoing Port (default)	7444 (do not change)	7444
IP		192.168.1.110
Username	administrator@vsphere.local (default)	administrator@vsphere.local
Password		vmware

Table 2-5. IaaS Database Passphrase

Variable	Value	Example
Passphrase (reused in IaaS Installer, Upgrade, and Migration)		myPassphrase

Table 2-6. IaaS Website

Variable	Value	Example
Host Name (FQDN)		iaas-web.mycompany.com
SSO service over HTTPS Outgoing Port (default)		
IP		192.168.1.106
Username		
Password		

Table 2-7. IaaS Model Manager Data

Variable	Value	Example
Host Name (FQDN)		iaas-model-man.mycompany.com
SSO service over HTTPS Outgoing Port (default)		
IP		192.168.1.107
Username		
Password		

Table 2-8. IaaS Model Service

Variable	Value	Example
Host Name (FQDN)		iaas-model-service.mycompany.com
SSO service over HTTPS Outgoing Port (default)		
IP		192.168.1.108
Username		
Password		

Table 2-9. Distributed Execution Managers

Unique Name	Orchestrator/Worker
ex. myuniqueorchestratorname	Orchestrator: Worker:
	Orchestrator: Worker:

Table 2-9. Distributed Execution Managers (Continued)

Unique Name	Orchestrator/Worker
	Orchestrator: Worker:
	Orchestrator: Worker:

Using vRealize Code Stream Installation Checklist

The installation checklist provides a high-level overview of the sequence of tasks you must perform to complete the vRealize Code Stream installation.

Installation Checklist

Use the checklist to track your work as you complete the installation tasks in the order they are listed.

Table 2-10. Installation Tasks

Task	Details
Complete the installation worksheet	Installation Worksheets
Set up your Identity Appliance	Deploy and Configure the Identity Appliance
Configure Single-Sign On for the Identity Appliance	Configure the Identity Appliance
Deploy the vRealize Automation appliance	Deploy the vRealize Appliance
Set up the vRealize Automation appliance	Configure the vRealize Appliance Complete tasks up to step 12.
Add the vRealize Code Stream license key	Apply a vRealize Code Stream License to an Appliance
Set up the Artifactory Server password	Set Up the Artifactory Server Password
Configure Tenant	Configuring Additional Tenants
Add vRealize Code Stream roles	Assign Roles to Identity Store Users or Groups Assign the Release Manager, Release Engineer, and Release Dashboard user roles for modeling and publishing a pipeline.
Register the Default Artifactory Server	Register an Artifactory Server for Artifact Management

Deploy and Configure the Identity Appliance

Download and configure the Identity Appliance to provide Single Sign-On (SSO) capability for the vRealize Automation environment.

You can use the Identity Appliance SSO provided with vRealize Automation or some versions of the SSO provided with vSphere. For information about supported versions, see *vRealize Automation Support Matrix*.

1 Deploy the Identity Appliance

 The Identity Appliance is a preconfigured virtual appliance that provides single sign-on capabilities. You download the Identity Appliance and deploy it into vCenter Server or ESX/ESXi inventory.

2 Enable Time Synchronization on the Identity Appliance

 You must synchronize the clocks on the Identity Appliance server, the vRealize Automation server, and Windows servers to ensure a successful installation.

3 Configure the Identity Appliance

 The Identity Appliance provides Single-Sign On (SSO) capability for vRealize Automation users. SSO is an authentication broker and security token exchange that interacts with the enterprise identity store (Active Directory or OpenLDAP) to authenticate users. A system administrator configures SSO settings to provide access to the vRealize Automation.

Deploy the Identity Appliance

The Identity Appliance is a preconfigured virtual appliance that provides single sign-on capabilities. You download the Identity Appliance and deploy it into vCenter Server or ESX/ESXi inventory.

Exact steps for this procedure vary depending on whether you use the native or Web vSphere client. Also, specific steps can vary depending on the your data center configuration.

Prerequisites

- Download the Identity Appliance from the VMware Web site.
- Log in to the vSphere client as a user with **system administrator** privileges.

Procedure

1 In the vSphere client, select **File > Deploy OVF Template**.

2 Browse to the Identity Appliance file with the `.ova` or `.ovf` extension and click **Open**.

3 Click **Next**.

4 Click **Next** on the OVF Template Details page.

5 Accept the license agreement and click **Next**.

6 Type a unique virtual appliance name according to the IT naming convention of your organization in the **Name** text box, select the datacenter and location to which you want to deploy the virtual appliance, and click **Next**.

7 Follow the prompts until the Disk Format page appears.

8 Verify on the Disk Format page that enough space exists to deploy the virtual appliance and click **Next**.

9 Follow the prompts to the Properties page.

 The options that appear depend on your vSphere configuration.

10 Configure the values on the Properties page.

   a Type the root password to use when you log in to the virtual appliance console in the **Enter password** and **Confirm password** text boxes.

   b Type the fully qualified domain name of the virtual machine in the **Hostname** text box, even if you are using DHCP.

   c Configure the networking properties.

   d Select or uncheck the SSH service checkbox to choose whether SSH service is enabled for the appliance.

   This value is used to set the initial status of the SSH service in the appliance. You can change this setting from the appliance management console when you configure the appliance.

11 Click **Next**.

12 If the **Power on after deployment** option is available on the Ready to Complete page, select it and click **Finish**.

13 Restart the machine.

14 Verify that the fully qualified domain name can be resolved against the IP address of the Identity Appliance by opening a command prompt and pinging the FQDN.

Enable Time Synchronization on the Identity Appliance

You must synchronize the clocks on the Identity Appliance server, the vRealize Automation server, and Windows servers to ensure a successful installation.

If you see certificate warnings during this procedure, continue past them.

Prerequisites

Deploy the Identity Appliance.

Procedure

1 Navigate to the Identity Appliance management console by using its fully qualified domain name, https://*identity-hostname.domain.name*:5480/.

2 Log in by using the user name root and the password you specified when you deployed the Identity Appliance.

3 Select **Admin > Time Settings**.

4 Select an option from the **Time Sync Mode** menu.

Option	Action
Use Time Server	Select **Use Time Server** from the **Time Sync Mode** menu to use Network Time Protocol . For each time server that you are using, type the IP address or the host name in the **Time Server** text box.
Use Host Time	Select **Use Host Time** from the **Time Sync Mode** menu to use VMware Tools time synchronization. You must configure the connections to Network Time Protocol servers before you can use VMware Tools time synchronization.

5 Click **Save Settings**.

6 Click **Refresh**.

7 Verify that the value in **Current Time** is correct.

You can change the time zone as required from the Time Zone Setting page on the **System** tab.

Configure the Identity Appliance

The Identity Appliance provides Single-Sign On (SSO) capability for vRealize Automation users. SSO is an authentication broker and security token exchange that interacts with the enterprise identity store (Active Directory or OpenLDAP) to authenticate users. A system administrator configures SSO settings to provide access to the vRealize Automation.

Note If you plan to use the vRealize Automation migration tool, you must specify a Native Active Directory when you configure the appliance.

Native Active Directories have the following characteristics:

- Use Kerberos to authenticate
- Do not require a search base, making it easier to find the correct Active Directory store
- Can be used only with the default tenant

Prerequisites

Enable Time Synchronization on the Identity Appliance.

Procedure

1 Navigate to the Identity Appliance management console by using its fully qualified domain name, https://*identity-hostname.domain.name*:5480/.

2 Continue past the certificate warning.

3 Log in with the user name root and the password you specified when the appliance was deployed.

4 Click the **SSO** tab.

The red text is a prompt, not an error message.

5 Type the password to assign to the system administrator in the **Admin Password** and **Repeat password** text boxes.

The **System Domain** text field has the value vsphere.local, which is the local default domain for the Identity Appliance. The default tenant is created with this name and the system administrator is administrator@vsphere.local. Record the user name and password in a secure place for later use.

6 Click **Apply**.

It can take several minutes for the success message to appear. Do not interrupt the process.

7 When the success message appears, click the **Host Settings** tab.

8 Verify that the **SSO Hostname** does not include the SSO port, :7444.

9 Go to the **SSL Configuration** panel.

10 Select the certificate type from the **Certificate Action** menu. If you are using a PEM encoded certificate, for example for a distributed environment, select **Import**.

Certificates that you import must be trusted and must also be applicable to all instances of vRealize Appliance and any load balancer by using Subject Alternative Name (SAN) certificates.

Note If you use certificate chains, specify the certificates in the following order:

- The client/server certificate signed by the intermediate CA certificate
- One or more intermediate certificates
- A root CA certificate

Option	Action
Import	a Copy the certificate values from BEGIN PRIVATE KEY to END PRIVATE KEY, including the header and footer, and paste them in the **RSA Private Key** text box. b Copy the certificate values from BEGIN CERTIFICATE to END CERTIFICATE, including the header and footer, and paste them in the **Certificate Chain** text box. c (Optional) If your certificate uses a pass phrase to encrypt the certificate key, copy the pass phrase and paste it in the **Pass Phrase** text box.
Generate Certificate	a Type a common name for the self-signed certificate in the **Common Name** text box. You can use the fully qualified domain name of the virtual appliance (*hostname.domain.name*) or a wild card, such as *.*mycompany*.com. b Type your organization name, such as your company name, in the **Organization** text box. c Type your organizational unit, such as your department name or location, in the **Organizational Unit** text box. d Type a two-letter ISO 3166 country code, such as US, in the **Country** text box.
Keep Existing	Leave the current SSL configuration. Select this option to preserve your settings.

11 Click **Apply Settings**.

After a few minutes the certificate details appear on the page.

12 Join the Identity Appliance to your Native Active Directory domain.

For migration, you must configure Native Active Directory. If you are not migrating, Native Active Directory is optional.

a Click the **Active Directory** tab.

b Type the domain name of the Active Directory in **Domain Name**.

c Enter the credentials for the domain administrator in the **Domain User** and **Password** text boxes.

d Click **Join AD Domain**.

13 Click the **Admin** tab.

14 Verify that the SSH settings are correct.

When **SSH service enabled** is selected, SSH is enabled for all but the root user. Select or uncheck **Administrator SSH login enabled** to enable or disable SSH login for the root user.

The SSO host is initialized. If Identity Appliance does not function correctly after configuration, redeploy and reconfigure the appliance. Do not make changes to the existing appliance.

Deploy and Configure the vRealize Appliance

The vRealize Appliance is a preconfigured virtual appliance that deploys the vRealize Appliance server and Web console (the user portal). It is delivered as an open virtualization format (OVF) template. The system administrator downloads the appliance and deploys it into the vCenter Server or ESX/ESXi inventory.

1 Deploy the vRealize Appliance

To deploy the vRealize Appliance, a system administrator must log in to the vSphere client and select deployment settings.

2 Enable Time Synchronization on the vRealize Appliance

Clocks on the Identity Appliance server, vRealize Automation server, and Windows servers must be synchronized to ensure a successful installation.

3 Configure the vRealize Appliance

To prepare the vRealize Appliance for use, a system administrator configures the host settings, generates an SSL certificate, and provides SSO connection information.

Deploy the vRealize Appliance

To deploy the vRealize Appliance, a system administrator must log in to the vSphere client and select deployment settings.

Prerequisites

- Download the vRealize Appliance from the VMware Web site.
- Log in to the vSphere client as a user with **system administrator** privileges.

Procedure

1. Select **File > Deploy OVF Template** from the vSphere client.
2. Browse to the vRealize Appliance file you downloaded and click **Open**.
3. Click **Next**.
4. Click **Next** on the OVF Template Details page.
5. Accept the license agreement and click **Next**.
6. Type a unique virtual appliance name according to the IT naming convention of your organization in the **Name** text box, select the datacenter and location to which you want to deploy the virtual appliance, and click **Next**.
7. Follow the prompts until the Disk Format page appears.
8. Verify on the Disk Format page that enough space exists to deploy the virtual appliance and click **Next**.
9. Follow the prompts to the Properties page.

   The options that appear depend on your vSphere configuration.
10. Configure the values on the Properties page.

    a Type the root password to use when you log in to the virtual appliance console in the **Enter password** and **Confirm password** text boxes.

    b Type the fully qualified domain name of the virtual machine in the **Hostname** text box, even if you are using DHCP.

    c Configure the networking properties.

    d Select or uncheck the SSH service checkbox to choose whether SSH service is enabled for the appliance.

    This value is used to set the initial status of the SSH service in the appliance. You can change this setting from the appliance management console when you configure the appliance.
11. Click **Next**.
12. If the **Power on after deployment** option is available on the Ready to Complete page, select it and click **Finish**.
13. Restart the machine.
14. Open a command prompt and ping the FQDN to verify that the fully qualified domain name can be resolved against the IP address of vRealize Appliance.

Enable Time Synchronization on the vRealize Appliance

Clocks on the Identity Appliance server, vRealize Automation server, and Windows servers must be synchronized to ensure a successful installation.

If you see certificate warnings during this process, continue past them to finish the installation.

Prerequisites

Deploy the vRealize Appliance.

Procedure

1. Navigate to the vRealize Appliance management console by using its fully qualified domain name, https://*vcac-va-hostname.domain.name*:5480/.
2. Log in with the user name root and the password you specified when the appliance was deployed.
3. Select **Admin > Time Settings**.
4. Select an option from the **Time Sync Mode** menu.

Option	Action
Use Time Server	Select **Use Time Server** from the **Time Sync Mode** menu to use Network Time Protocol . For each time server that you are using, type the IP address or the host name in the **Time Server** text box.
Use Host Time	Select **Use Host Time** from the **Time Sync Mode** menu to use VMware Tools time synchronization. You must configure the connections to Network Time Protocol servers before you can use VMware Tools time synchronization.

5. Click **Save Settings**.
6. Verify that the value in **Current Time** is correct.

   You can change the time zone as required from the Time Zone Setting page on the **System** tab.
7. (Optional) Click **Time Zone** from the **System** tab and select a system time zone from the menu choices.

   The default is Etc/UTC.
8. Click **Save Settings**.

Configure the vRealize Appliance

To prepare the vRealize Appliance for use, a system administrator configures the host settings, generates an SSL certificate, and provides SSO connection information.

Prerequisites

Enable Time Synchronization on the vRealize Appliance.

Procedure

1. Navigate to the vRealize Appliance management console by using its fully qualified domain name, https://*vcac-va-hostname.domain.name*:5480/.
2. Continue past the certificate warning.
3. Log in with user name root and the password you specified when you deployed vRealize Appliance.
4. Select **vRA Settings > Host Settings** and select **Resolve Automatically** to view the name of the currently specified host.

5 (Optional) If you want to change the host name, select **Update Host** and enter the fully qualified domain name, *vra-hostname.domain.name*, of the vRealize Appliance in the **Host Name** text box. If you are using a load balancer, enter the fully qualified domain name for the load balancer server.

6 Go to the **SSL Configuration** panel.

7 Select the certificate type from the **Certificate Action** menu. If you are using a PEM encoded certificate, for example for a distributed environment, select **Import**.

Certificates that you import must be trusted and must also be applicable to all instances of vRealize Appliance and any load balancer through the use of Subject Alternative Name (SAN) certificates.

Note If you use certificate chains, specify the certificates in the following order:

- Client/server certificate signed by the intermediate CA certificate
- One or more intermediate certificates
- A root CA certificate

Option	Action
Import	a Copy the certificate values from BEGIN PRIVATE KEY to END PRIVATE KEY, including the header and footer, and paste them in the **RSA Private Key** text box. b Copy the certificate values from BEGIN CERTIFICATE to END CERTIFICATE, including the header and footer, and paste them in the **Certificate Chain** text box. For multiple certificate values, include a BEGIN CERTIFICATE header and END CERTIFICATE footer for each certificate. c (Optional) If your certificate uses a pass phrase to encrypt the certificate key, copy the pass phrase and paste it in the **Pass Phrase** text box.
Generate Certificate	a Type a common name for the self-signed certificate in the **Common Name** text box. You can use the fully qualified domain name of the virtual appliance (*hostname.domain.name*) or a wild card, such as *.*mycompany*.com. If you use a load balancer, you need to specify the FQDN of the load balancer or a wildcard that matches the name of the load balancer. Do not accept a default value if one is shown, unless it matches the host name of the virtual appliance. b Type your organization name, such as your company name, in the **Organization** text box. c Type your organizational unit, such as your department name or location, in the **Organizational Unit** text box. d Type a two-letter ISO 3166 country code, such as US, in the **Country** text box.
Keep Existing	Leave the current SSL configuration. Select this option to preserve your settings.

8 Click **Save Settings** to save host information and SSL configuration.

9 Configure the SSO settings that the vRealize Appliance uses to interact with the Identity Appliance. These settings must match the settings you entered when configuring the Identity Appliance.

   a Click **SSO**.

   b Type the fully qualified domain name of the Identity Appliance, *identity-va-hostname.domain.name* in the **SSO Host** text box. Do not use an https:// prefix.

      For example, `vcac-sso.mycompany.com`.

   c The default port number, 7444, is displayed in the **SSO Port** text box. Edit this value if you are using a non-default port.

   d Do not modify the default tenant name, `vsphere.local`, in the **SSO Default Tenant** text box.

   e Type the default administrator name `administrator@vsphere.local` in the **SSO Admin User** text box.

   f Type the SSO administrator password in the **SSO Admin Password** text box. The password must match the password you specified in the SSO settings for the Identity Appliance.

   g Click **Save Settings**.

      After a few minutes, a success message appears and SSO Status is updated to Connected.

   h (Optional) Select **Apply Branding** to apply vRealize Automation branding to your installation.

      Use this option if you are installing from vCenter and want to use vRealize Automation instead of vCenter branding.

   i (Optional) If the spinner does not stop within a few minutes, exit the appliance, close the browser, and log in again.

10 If you plan to deploy your PostgreSQL database on a standalone host, specify the database information.

   a Click **Database**.

   b Specify the host, port, database name (the default is vcac), and the database authentication information for the PostgreSQL database.

   c Click **Save Settings**.

11 If you see the message `Error restarting VCAC server` after you click **Save Settings**, ignore the message and continue with the next step.

12 Click **Messaging**. The configuration settings and status of messaging for your appliance is displayed. Do not change these settings.

13 Click the **Telemetry** tab.

   You can choose to participate in the Customer Experience Improvement Program. You can unsubscribe from the program at any time.

   - Select **Enable** to activate the Program.
   - Deselect **Enable** to unsubscribe from the Program.

When you enable the Program, vRealize Automation attempts to establish a connection to https://vmware.com and to automatically discover any proxy server you might have configured for your vRealize Automation deployment.

14 Click **Services**.

   The following services must be running before you can log in to the console. Depending on your site configuration, this can take about 10 minutes.

   - authorization
   - authentication
   - eventlog-service
   - shell-ui-app
   - branding-service
   - plugin-service

   Note You can log in to the appliance and run `tail –f /var/log/vcac/catalina.out` to monitor startup of the services.

15 Configure the license to enable the Infrastructure tab on the vRealize Automation console.

   a Click **vRA Settings > Licensing**.

   b Click **Licensing**.

   c Type a valid vRealize Automation license key that you downloaded when you downloaded the installation files, and click **Submit Key**.

   Note If you experience a connection error, you might have a problem with the load balancer. Check network connectivity to the load balancer.

16 Confirm that you can log in to the vRealize Automation console.

   a Open a browser and navigate to https://*vcac-hostname.domain.name*/vcac.

   b Accept the vRealize Automation certificate.

   c Accept the SSO certificate.

   d Log in with administrator@vsphere.local and the password you specified when you configured SSO.

      The console opens to the Tenants page on the **Administration** tab. A single tenant named vsphere.local appears in the list.

You have finished the deployment and configuration of your vRealize Appliance. If the appliance does not function correctly after configuration, redeploy and reconfigure the appliance. Do not make changes to the existing appliance.

Apply a vRealize Code Stream License to an Appliance

When you apply the vRealize Code Stream standalone license to a vRealize Automation Appliance, you enable the vRealize Code Stream functions. If you enter the vRealize Code Stream and vRealize Automation licenses, you can enable the vRealize Automation and vRealize Code Stream features.

You can use the vRealize Code Stream standalone license to enable the Artifact Management, Release Management, Release Dashboard, Approval Services, and Advanced Service Designer features. You can apply the vRealize Code Stream license in conjunction with the vRealize Automation Standard, Advanced, or Enterprise licenses.

Overlapping features in vRealize Automation and vRealize Code Stream are combined. For example, if you apply a vRealize Code Stream license to an existing vRealize Automation appliance, the common Advanced Service Designer feature works as is in the appliance.

Prerequisites

Verify that the vRealize Automation appliance is set up. See Configure the vRealize Appliance.

Procedure

1 Open the vRealize Automation Appliance management console with the fully qualified domain name, https:// vra-va-*hostname.domain.name*:5480/.

2 Log in as the root user.

3 Select **vRA Settings > Licensing**.

4 Enter a valid vRealize Code Stream license key and click **Submit Key**.

   The default Artifactory server is enabled when the valid license key is accepted.

   The vRealize Code Stream license includes the Artifactory Pro version.

5 Confirm that you can log in to the vRealize Automation console.

   a Open a Web browser.

   b Navigate to https://vra-*hostname.domain.name*/vcac.

What to do next

Configure a repository in the Artifactory server. See the JFrog Web site https://www.jfrog.com/confluence/display/RTF/Configuring+Repositories.

Set Up the Artifactory Server Password

The default Artifactory server is enabled when you apply the vRealize Code Stream license to the appliance. For security purposes, change the default login credentials.

The vRealize Code Stream license includes the Artifactory Pro version.

Procedure

1. Open a Web browser.
2. Enter the **`https://vra-`*`hostname`*`/artifactory/`** URL.
3. Log in using the default username **`vmadmin`** and password **`vmware`**.
4. Select the **Admin** tab and click **Users** in the left pane.
5. Select the **vmadmin** user name.
6. Change the default login credentials.
7. Click **Save.**
8. Log out.

Configure an External Disk Partition for the Artifactory Server

If you plan to store large binaries you can configure external disk partition to the default Artifactory server.

The local storage in the appliance is 25GB.

Prerequisites

- Verify that you have a vRealize Automation host.
- Verify that the NFS share path is available with adequate storage.

Procedure

1. SSH into the vRealize Automation host using root@*VRA-host-name*.
2. Type the vRealize Automation appliance password.
3. Stop the artifactory server.

   ```
   /opt/jfrog/artifactory/bin/artifactroy.sh stop
   ```

4. Mount the NFS share to the vRealize Automation host.

   ```
   mount NFS-server-host-name:/host/NFS-share-host-path /mount/VRA-path
   ```

5. Navigate to the `/opt/jfrog/artifactory/misc/db/postgresql.properties` file.
6. Uncomment the binary.provider.type entry in the file.

   binary.provider.type=filesystem

7. Add the new NFS share path in the file.

   ```
   binary.provider.filesystem.dir=NFS-share-path
   ```

8. Restart the artifactory server.

   ```
   /opt/jfrog/artifactory/bin/artifactroy.sh start
   ```

Configuring Additional Tenants

You create the default tenant when you install vRealize Automation, but you can create additional tenants to represent business units in an enterprise or companies that subscribe to cloud services from a service provider.

Tenancy Overview

A tenant is an organizational unit in a vRealize Automation deployment. A tenant can represent a business unit in an enterprise or a company that subscribes to cloud services from a service provider.

Each tenant has its own dedicated configuration. Some system-level configuration is shared across tenants.

Table 2-11. Tenant Configuration

Configuration Area	Description
Login URL	Each tenant has a unique URL to the vRealize Automation console. ■ The default tenant URL is in the following format: https://*hostname*/vcac ■ The URL for additional tenants is in the following format: https://*hostname*/vcac/org/*tenantURL*
Identity stores	Each tenant requires access to one or more directory services, such as OpenLDAP or Microsoft Active Directory servers, that are configured to authenticate users. You can use the same directory service for more than one tenant, but you must configure it separately for each tenant.
Branding	A tenant administrator can configure the branding of the vRealize Automation console including the logo, background color, and information in the header and footer. System administrators control the default branding for all tenants.
Notification providers	System administrators can configure global email servers that process email notifications. Tenant administrators can override the system default servers, or add their own servers if no global servers are specified.
Business policies	Administrators in each tenant can configure business policies such as approval workflows and entitlements. Business policies are always specific to a tenant.
Service catalog offerings	Service architects can create and publish catalog items to the service catalog and assign them to service categories. Services and catalog items are always specific to a tenant.
Infrastructure resources	The underlying infrastructure fabric resources, for example, vCenter servers, Amazon AWS accounts, or Cisco UCS pools, are shared among all tenants. For each infrastructure source that vRealize Automation manages, a portion of its compute resources can be reserved for users in a specific tenant to use.

About the Default Tenant

When the system administrator configures single sign-on during the installation of vRealize Automation, a default tenant is created with the built-in system administrator account to log in to the vRealize Automation console. The system administrator can then configure the default tenant and create additional tenants.

The default tenant supports all of the functions described in Tenant Configuration. In the default tenant, the system administrator can also manage system-wide configuration, including global system defaults for branding and notifications, and monitor system logs.

The default tenant is the only tenant that supports native Active Directory authentication. All other tenants must use Active Directory over OpenLDAP.

User and Group Management

All user authentication is handled through single sign-on. Each tenant has one or more identity stores, such as Active Directory servers, that provide authentication.

The system administrator performs the initial configuration of single sign-on and basic tenant setup, including designating at least one identity store and a tenant administrator for each tenant. Thereafter, a tenant administrator can configure additional identity stores and assign roles to users or groups from the identity stores.

Tenant administrators can also create custom groups within their own tenant and add users and groups defined in the identity store to custom groups. Custom groups, like identity store groups and users, can be assigned roles or designated as the approvers in an approval policy.

Tenant administrators can also create business groups within their tenant. A business group is a set of users, often corresponding to a line of business, department or other organizational unit, that can be associated with a set of catalog services and infrastructure resources. Users, identity store groups, and custom groups can be added to business groups.

Comparison of Single-Tenant and Multitenant Deployments

vRealize Automation supports deployments with either a single tenant or multiple tenants. The configuration can vary depending on how many tenants are in your deployment.

System-wide configuration is always performed in the default tenant and can apply to one or more tenants. For example, system-wide configuration might specify defaults for branding and notification providers.

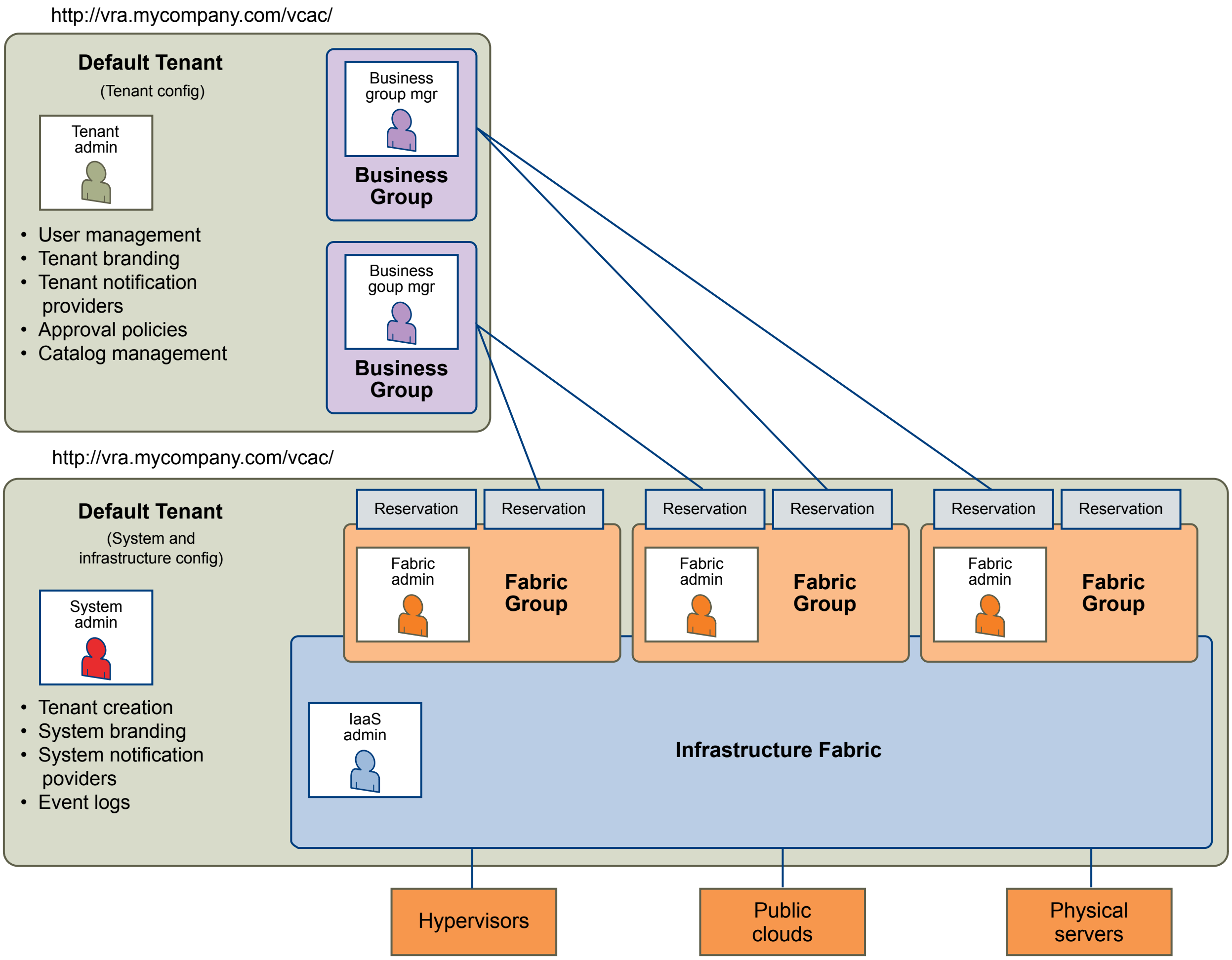
Infrastructure configuration, including the infrastructure sources that are available for provisioning, can be configured in any tenant and is shared among all tenants. The infrastructure resources, such as cloud or virtual compute resources or physical machines, can be divided into fabric groups managed by fabric administrators. The resources in each fabric group can be allocated to business groups in each tenant by using reservations.

Single-Tenant Deployment

In a single-tenant deployment, all configuration can occur in the default tenant. Tenant administrators can manage users and groups, configure tenant-specific branding, notifications, business policies, and catalog offerings.

All users log in to the vRealize Automation console at the same URL, but the features available to them are determined by their roles.

Figure 2-1. Single-Tenant Example



Note In a single-tenant scenario, it is common for the system administrator and tenant administrator roles to be assigned to the same person, but two distinct accounts exist. The system administrator account is always administrator@vsphere.local. The tenant administrator must be a user in one of the tenant identity stores, such as *username*@mycompany.com.

Multitenant Deployment

In a multitenant environment, the system administrator creates tenants for each organization that uses the same vRealize Automation instance. Tenant users log in to the vRealize Automation console at a URL specific to their tenant. Tenant-level configuration is segregated from other tenants and from the default tenant. Users with system-wide roles can view and manage configuration across multiple tenants.

There are two main scenarios for configuring a multi-tenant deployment.

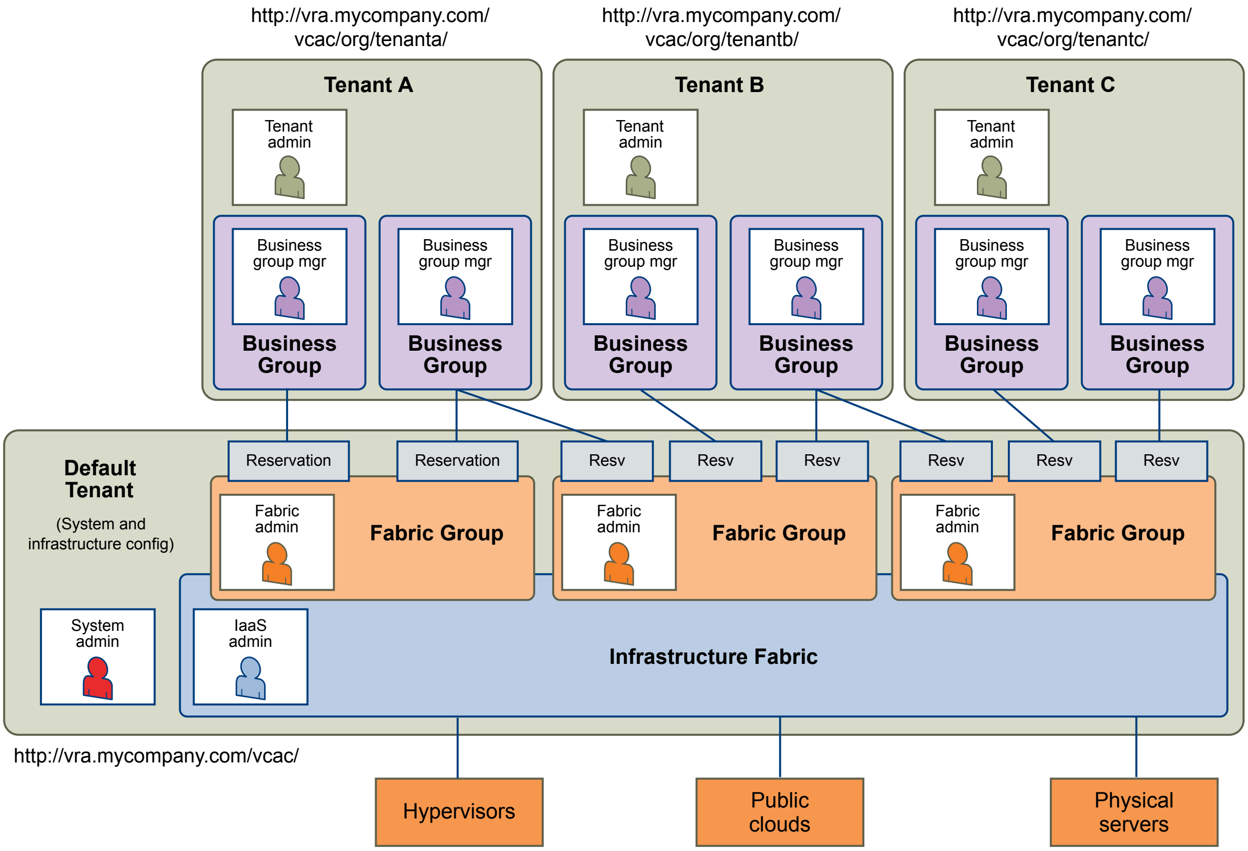
Table 2-12. Multitenant Deployment Examples

Example	Description
Manage infrastructure configuration only in the default tenant	In this example, all infrastructure is centrally managed by IaaS administrators and fabric administrators in the default tenant. The shared infrastructure resources are assigned to the users in each tenant by using reservations.
Manage infrastructure configuration in each tenant	In this scenario, each tenant manages its own infrastructure and has its own IaaS administrators and fabric administrators. Each tenant can provide its own infrastructure sources or can share a common infrastructure. Fabric administrators manage reservations only for the users in their own tenant.

The following diagram shows a multitenant deployment with centrally managed infrastructure. The IaaS administrator in the default tenant configures all infrastructure sources that are available for all tenants. The IaaS administrator can organize the infrastructure into fabric groups according to type and intended purpose. For example, a fabric group might contain all virtual resources, or all Tier One resources. The fabric administrator for each group can allocate resources from their fabric groups. Although the fabric administrators exist only in the default tenant, they can assign resources to business groups in any tenant.

Note Some infrastructure tasks, such as importing virtual machines, can only be performed by a user with both the fabric administrator and business group manager roles. These tasks might not be available in a multitenant deployment with centrally managed infrastructure.

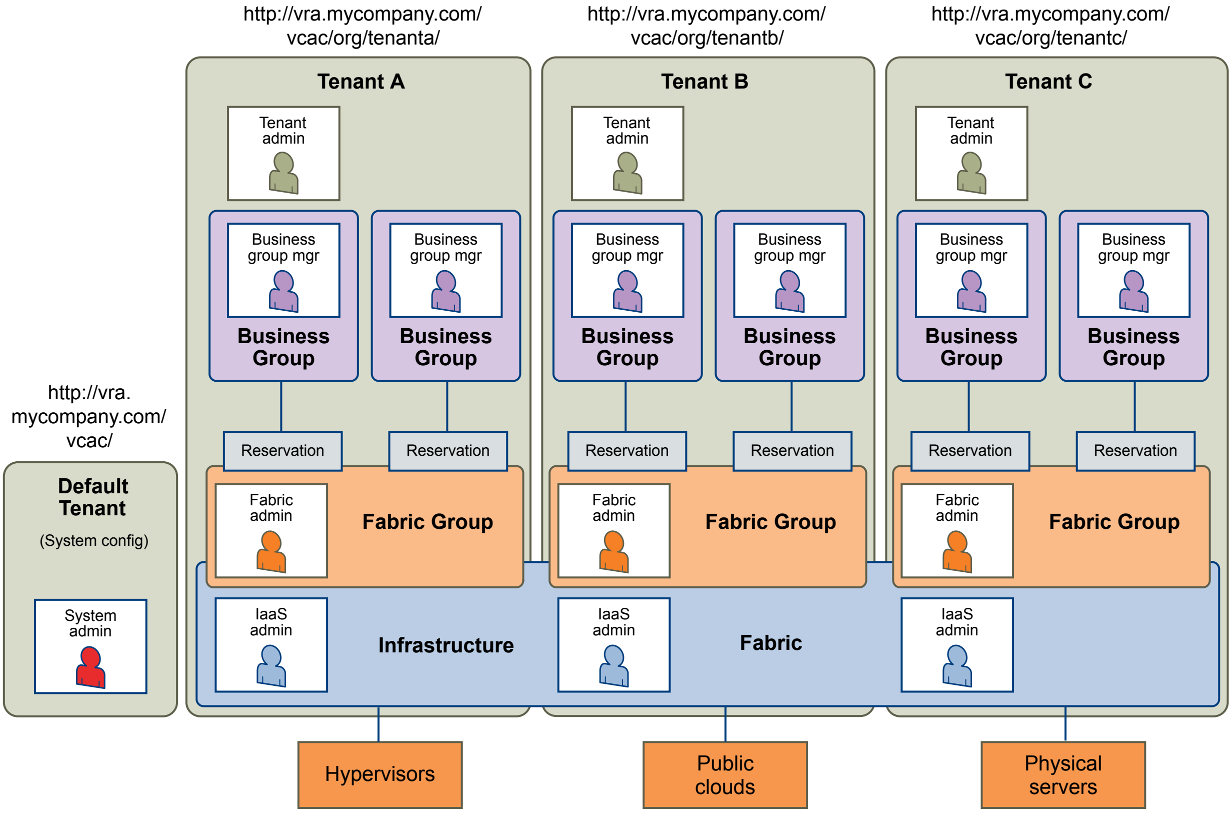
Figure 2-2. Multitenant Example with Infrastructure Configuration Only in Default Tenant



The following diagram shows a multitenant deployment where each tenant manages their own infrastructure. The system administrator is the only user who logs in to the default tenant to manage system-wide configuration and create tenants.

Each tenant has an IaaS administrator, who can create fabric groups and appoint fabric administrators with their respective tenants. Although fabric administrators can create reservations for business groups in any tenant, in this example they typically create and manage reservations in their own tenants. If the same identity store is configured in multiple tenants, the same users can be designated as IaaS administrators or fabric administrators in each tenant.

Figure 2-3. Multitenant Example with Infrastructure Configuration in Each Tenant



Create and Configure a Tenant

System administrators create tenants and specify basic configuration such as name, login URL, identity stores, and administrators.

Prerequisites

Log in to the vRealize Automation console as a **system administrator**.

Procedure

1 Specify Tenant Information

The first step to configuring a tenant is to add the new tenant to vRealize Automation and create the tenant-specific access URL.

2. Configure Identity Stores

   Each tenant must be associated with at least one identity store. Identity stores can be OpenLDAP or Active Directory. Use of Native Active Directory is also supported for the default tenant.

3. Appoint Administrators

   You can appoint one or more tenant administrators and IaaS administrators from the identity stores you configured for a tenant.

Specify Tenant Information

The first step to configuring a tenant is to add the new tenant to vRealize Automation and create the tenant-specific access URL.

Prerequisites

Log in to the vRealize Automation console as a **system administrator**.

Procedure

1. Select **Administration > Tenants**.
2. Click the **Add** icon (✚).
3. Enter a name in the **Name** text box.
4. (Optional) Enter a description in the **Description** text box.
5. Type a unique identifier for the tenant in the **URL Name** text box.

   This URL token is used to create tenant-specific URLs to access vRealize Automation.
6. (Optional) Type an email address in the **Contact Email** text box.
7. Click **Submit and Next**.

Your new tenant is saved and you are automatically directed to the **Identity Stores** tab for the next step in the process.

Configure Identity Stores

Each tenant must be associated with at least one identity store. Identity stores can be OpenLDAP or Active Directory. Use of Native Active Directory is also supported for the default tenant.

Prerequisites

Specify Tenant Information.

Procedure

1. Click the **Add** icon (✚).
2. Enter a name in the **Name** text box.
3. Select the type of identity store from the **Type** drop-down menu.

4 Type the URL for the identity store in the **URL** text box.

For example, `ldap://ldap.mycompany.com:389` .

5 Type the domain for the identity store in the **Domain** text box.

6 (Optional) Type the domain alias in the **Domain Alias** text box.

The alias allows users to log in by using *userid@domain-alias* rather than *userid@identity-store-domain* as a user name.

7 Type the Distinguished Name for the login user in the **Login User DN** text box.

Use the display format of the user name, which can include spaces and is not required to be identical to the user ID.

For example, `cn=Demo Admin,ou=demo,dc=dev,dc=mycompany,dc=com`.

8 Type the password for the identity store login user in the **Password** text box.

9 Type the group search base Distinguished Name in the **Group Search Base DN** text box.

For example, `ou=demo,dc=dev,dc=mycompany,dc=com`.

10 (Optional) Type the user search base Distinguished Name in the **User Search Base DN** text box.

For example, `ou=demo,dc=dev,dc=mycompany,dc=com`.

11 Click **Test Connection**.

Check that the connection is working.

12 Click **Add**.

13 (Optional) Repeat Step 1 to Step 12 to configure additional identity stores.

14 Click **Next**.

Your new identity store is saved and associated with the tenant. You are directed to the **Administrators** tab for the next step in the process.

Appoint Administrators

You can appoint one or more tenant administrators and IaaS administrators from the identity stores you configured for a tenant.

Tenant administrators are responsible for configuring tenant-specific branding, as well as managing identity stores, users, groups, entitlements, and shared blueprints within the context of their tenant. IaaS Administrators are responsible for configuring infrastructure source endpoints in IaaS, appointing fabric administrators, and monitoring IaaS logs.

Prerequisites

- Configure Identity Stores.
- Before you appoint IaaS administrators, you must install IaaS. For more information about installation, see *Installation and Configuration*.

Procedure

1. Type the name of a user or group in the **Tenant Administrators** search box and press Enter.

   Repeat this step to appoint additional tenant administrators.

2. Type the name of a user or group in the **Infrastructure Administrators** search box and press Enter.

   Repeat this step to appoint additional IaaS administrators.

3. Click **Update**.

Managing Users

Tenant administrators create and manage custom groups and grant and manage user access rights to the vRealize Automation console.

Add Identity Store

vRealize Automation uses identity stores to authenticate users. Each tenant is associated with at least one identity store when it is created, but you can add new ones if necessary.

When you delete an identity store, this removes the roles assigned to users from this store, the roles assigned to users from custom groups, and the information about which services are available to this user. Entries for entitlements and business groups are not affected.

Prerequisites

Log in to the vRealize Automation console as a **tenant administrator**.

Procedure

1. Select **Administration > Identity Stores**.
2. Click the **Add** icon (✚).
3. Enter a name in the **Name** text box.
4. Select the type of the identity store from the **Type** drop-down menu.
5. Enter the following Identify Store configuration options.

Option	Action
URL	Enter the URL for the identity store. For example, `ldap://10.141.64.166:875`.
Domain	Enter the domain for the identity store.
(Optional) Domain Alias	Enter the domain alias.
Login User DN	Enter the login user Distinguished Name. For example, `cn=demoadmin,ou=demo,dc=dev,dc=mycompany,dc=com`.
Password	Enter the password for the identity store login user.
Group Search Base DN	Enter the group search base Distinguished Name. For example, `ou=demo,dc=dev,dc=mycompany,dc=com`.
User Search Base DN	Enter the user search base Distinguished Name.

6 Click **Test Connection**.

7 Click **Add**.

What to do next

Assign Roles to Identity Store Users or Groups.

Assign Roles to Identity Store Users or Groups

Tenant administrators grant users access rights by assigning roles to users or groups.

Prerequisites

Log in to the vRealize Automation console as a **tenant administrator**.

Procedure

1 Select **Administration > Users & Groups > Identity Store Users & Groups**.

2 Enter a user or group name in the **Search** box and press Enter.

   Do not use an at sign (@), backslash (\), or slash (/) in a name. You can optimize your search by typing the entire user or group name in the form user@domain.

3 Click the name of the user or group to which you want to assign roles.

4 Select one or more roles from the Add Roles to this User list.

   The Authorities Granted by Selected Roles list indicates the specific authorities you are granting.

5 (Optional) Click **Next** to view more information about the user or group.

6 Click **Update**.

Users who are currently logged in to the vRealize Automation console must log out and log back in to the vRealize Automation console before they can navigate to the pages to which they have been granted access.

What to do next

Optionally, you can create your own custom groups from users and groups in your identity stores. See Create a Custom Group.

Create a Custom Group

Tenant administrators can create custom groups by combining other custom groups, identity store groups, and individual identity store users.

You can assign roles to your custom group, but it is not necessary in all cases. For example, you can create a custom group called Machine Specification Approvers, to use for all machine pre-approvals. You can also create custom groups to map to your business groups so that you can manage all groups in one place. In those cases, you do not need to assign roles.

Prerequisites

Log in to the vRealize Automation console as a **tenant administrator**.

Procedure

1. Select **Administration > Users & Groups > Custom Groups**.
2. Click the **Add** icon (✚).
3. Enter a group name in the **New Group Name** text box.
4. (Optional) Enter a description in the **New Group Description** text box.
5. Select one or more roles from the Add Roles to this Group list.

   The Authorities Granted by Selected Roles list indicates the specific authorities you are granting.
6. Click **Next**.
7. Add users and groups to create your custom group.
   1. Enter a user or group name in the **Search** box and press Enter.

      Do not use an at sign (@), backslash (\), or slash (/) in a name. You can optimize your search by typing the entire user or group name in the form user@domain.
   2. Select the user or group to add to your custom group.
8. Click **Add**.

Users who are currently logged in to the vRealize Automation console must log out and log back in to the vRealize Automation console before they can navigate to the pages to which they have been granted access.

Register an Artifactory Server for Artifact Management

To use artifact management in a release pipeline, you must connect to an Artifactory server.

With artifact management, you can specify an artifact by name and search type from the server, but not by location or unique identifier. Artifact management monitors the physical location and identity of artifacts and supplies the required artifact during the pipeline execution.

Prerequisites

- Log in to the vRealize Automation console as a tenant administrator.
- Verify that the Artifactory server is not using the default login credentials. See Set Up the Artifactory Server Password.
- Verify that the tenant role is assigned for the **Release Automation** and **Dashboard** tabs to appear in the appliance user interface. See Configuring Additional Tenants.
- Verify that the Artifactory server is configured. See the *Artifactory User Guide* on the jFrog Web site.

Procedure

1. Select **Administration > Artifact Management**.
2. Enter the name of the Artifactory server.

   You can add notes that apply to the server in the Description section.
3. Enter the URL of the Artifactory server.

   The server URL format is https://vra-*hostname*/artifactory/.
4. Enter the login credentials of the Artifactory server.
5. Click **Validate** to verify that the appliance can connect to the Artifactory server.
6. Click **Update** to save your changes.

The release pipeline can access the Artifactory server.

Registering Plug-in Instances for Release Pipeline

3

You must define and configure the plug-ins before you create a task in a release pipeline.

An instance of a plug-in must be created in the vRealize Orchestrator client to enable it in the release pipeline.

This chapter includes the following topics:

Register a Jenkins Server

You can run tests or other jobs by using the Jenkins plug-in. Any Jenkins job in the Jenkins server can be used in release automation with this plug-in enabled. You use this plug-in to select a Jenkins job during the modeling of a release pipeline, configure the input and output properties, and execute the job as part of the release pipeline.

Prerequisites

- Verify that the Jenkins server is available and configured with or without SSL.
- Verify that the Jenkins jobs are created in the Jenkins server with the input string parameter, VRCSTestExecutionId.

Procedure

1 Log in to the vRealize Orchestrator client to add a Jenkins instance.

2 Select**Library** > **RPTestJenkins** > **Configuration** > **Add a RPTestJenkins Instance.**

3 Right-click the **Add an RPTestJenkins Instance** workflow and select **Start Workflow.**

4 Enter the server configuration details.

Option	Description
Name	Jenkins instance name
User Credentials	User name and password for the Jenkins server

Option	Description
URL	Host URL as *protocol://host:port*
Polling Interval	Time that the task must wait to check the progress
Request Retry Count	Number of times to retry the scheduled build request for the Jenkins server
Retry Wait Time	Seconds to wait before retrying the build request for the Jenkins server

5 Click **Submit**.

6 Click the **Yes** button to import the certificate and click **Submit**.

7 Click the **Release Automation** tab in vRealize Automation to continue with the task configuration.

Register a vRealize Automation Server

You can provision software by using a vRealize Automation plug-in.

Prerequisites

Verify that you have configured the following components in vRealize Automation.

- Choosing an Endpoint Scenario see *IaaS Configuration for Virtual Platforms*.
- Create a Fabric Group see *IaaS Configuration for Virtual Platforms*.
- Create a Business Group see *IaaS Configuration for Virtual Platforms*.
- Create a Reservation see *IaaS Configuration for Virtual Platforms*.
- Create a Reservation Policy see *IaaS Configuration for Physical Machines*.
- Create a Network Profile see *IaaS Integration for Multi-Machine Services*.
- Create a Blueprint see *IaaS Configuration for Virtual Platforms*.
- Publish a Blueprint see *IaaS Configuration for Virtual Platforms*.

Procedure

1 Log in to the vRealize Orchestrator client to create an instance.

2 Select **Library** > **RPProvisionVCAC** > **Configuration** > **Add a VCAC Instance**.

3 Right-click the **Add a VCAC Instance workflow** and select **Start Workflow**.

4 Enter the server configuration details.

Option	Description
Name	vRealize Automation instance name.
URL	Host URL. The URL format is https://serverIP.eng.vcloud.com/.
User Credentials	User name and password for the vRealize Automation server.
Polling Interval	Time that the task must wait to check the progress.
Wait time for IP assignment	Time that the task must wait for the IP addresses to be assigned to a machine before it can report the failure.

5 Click **Submit**.

6 Click the **Yes** button to import the certificate, and click **Submit**.

7 Select the **Release Automation** tab invRealize Automation to continue with the task configuration.

Register a vRealize Orchestrator Workflow for a Custom Task

With the vRealize Orchestrator workflow plug-in for the custom task, you can use any of the custom vRealize workflows from release automation. You can select the custom workflow, configure it, and use it to model the pipeline and execute from release automation.

Prerequisites

- Verify that the vRealize Automation server is installed with vRealize Code Stream.
- Verify that the workflow for custom workflow is created.

Procedure

1 Log in to the vRealize Orchestrator client to create a workflow.

2 Select **Library** > **Tagging** > **Tag workflow**.

   The Tag workflow is required for any custom user workflow.

   By default, Manual Task workflow is provided.

3 Right-click the **Tag workflow** and select **Start Workflow**.

4 Select **custom workflow** in the Tagged workflow.

   For example, the tagged workflow can be Deploy Spring Travel.

5 Enter the tag as **CUSTOM**.

6 Enter the value as **CUSTOM**.

7 Click the **Yes** button for the Global tag and click **Submit**.

8 Click the **Release Automation** tab in vRealize Automation to continue with the task configuration.

Register a vRealize Orchestrator Workflow for Gating Rule

With the vRealize Orchestrator workflow plug-in for Gating Rules, you can use any workflow that you want to use as a gating rule workflow from release automation. You can select the custom workflow, configure it with the GATING_RULE tag, and use it to model the pipeline and execute from release automation.

Prerequisites

- Verify that the vRealize Automation server is installed with vRealize Code Stream.
- Verify that the workflow for GATING_RULE Workflow is created.

Procedure

1 Log in to the vRealize Orchestrator client to create a workflow.

2 Select **Library** > **Tagging** > **Tag workflow**.

3 Right-click the **Tag workflow** and select **Start Workflow.**

4 Enter the Tag as **GATING_RULE**.

5 Enter the value as **GATING_RULE**.

6 Click the **Yes** button for the Global tag and click **Submit**.

7 Click the **Release Automation** tab in vRealize Automation to continue with the task configuration.

Using Release Automation 4

The software development life cycle includes work phases before it moves to production. As the software changes move closer to production, the quality checks and approval policies become stringent. This process is enforced to ensure that no disruptions occur in the production environment.

vRealize Code Stream enables central IT to host and manage new application workloads being driven by lines of business and development operation teams. Application teams can independently use vRealize Code Stream to automate and streamline their software release process while continuing to use their preferred provisioning and deployment tools.

vRealize Code Stream also enables applications or operations teams to model their software release process in a release pipeline. A release pipeline is a sequence of stages where each stage is composed of multiple tasks and environments that the software has to pass through before it is released to production. The stages can include development, functional testing, user acceptance test, load testing, systems integration testing, staging, and production.

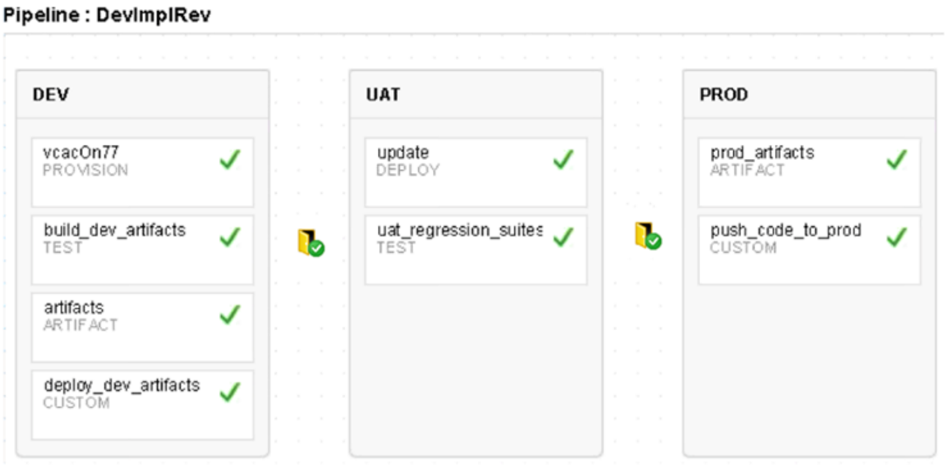
In each stage, teams might also use different kinds of development and management tools. Using different tools results in needing to build a product that is extensible and that can integrate easily with a variety of existing tools. To satisfy this need for flexibility, vRealize Code Stream offers a model-driven, open, and extensible architecture. With its catalog of plug-ins, you can integrate with existing tools, including build and integration systems, testing frameworks, provisioning, deployment engines, change management systems, and so on.

Each stage in a pipeline includes a set of activities such as provisioning a machine, retrieving an artifact, deploying software, running a test, creating a manual task, or running a custom workflow or script. The software changes are promoted to the next stage in the pipeline when they satisfy a set of rules called gating rules. The gating rules include testing rules and compliance rules. Gating rules that are associated with a pipeline are specific to an organization or an application. Users can define gating rules when a pipeline template is created. The environments do not need to be aware of these gating rules.

Modeling a Pipeline Checklist

A release pipeline is a collection of stages where each stage represents a deployment environment. For example the development, test, user acceptance test (UAT), load test (LT), systems integration testing (SIT), and staging environments that a software change has to pass through independently before it is released.

Sample Release Pipeline with Stages



The number of stages and configuration of each of these stages varies based on the application, whether the release is major, minor, or patch, or organization release policies.

To track your work as you complete the modeling tasks use the topics in the order they are given.

1 Create a Pipeline

   You can add, edit, view, clone, and delete the release pipeline from the **Pipelines** tab. You can have multiple stages and build IDs for a single release pipeline.

2 Using Pipeline Task Variables

   Pipeline variables are runtime variables that are output from a previous task and are available for consumption by other tasks.

3 Configure an Artifact Task

   An artifact task lets you search the library of binaries in different repositories that you can deploy on a virtual machine.

4 Configure a Custom Script Task

   You can use a custom script task to run a script that resides on a remote host. After the script runs, you can monitor the script progress and capture the script response, which you can pass as input to other release pipeline tasks.

5 Configure a Custom Task

   A custom task allows any custom activity that you can start with a vRealize Orchestrator workflow or script. This activity can also be a manual approval.

6 Configure a Deploy Task

   A deploy tasks deploys or updates the artifacts to the machines.

7 Configure a Provision Task

   A provision task provisions the machines. The plug-in provider for this task is vRealize Automation. Based on the provider, specific configuration is required. You can create machines from a single machine blueprint for a provision task. The task output is an array of machines.

8 Configure a Test Task

   A test task lets you test the deployment or application.

9 Configure Gating Rules

Gating rules are a set of criteria that each stage must pass to proceed to the subsequent stage. You can configure gating rules based on your requirement for a pipeline.

10 Activate and Run a Pipeline

You can run a pipeline that is activated. After you activate the pipeline, pipeline modeling is complete and you can run it.

Create a Pipeline

You can add, edit, view, clone, and delete the release pipeline from the **Pipelines** tab. You can have multiple stages and build IDs for a single release pipeline.

These pipelines form a single application or a module. You can model a pipeline with multiple stages and tasks within a stage. You can use each to depict the release cycle for an application or a module.

Prerequisites

- Verify that SSO, OVA, and OVF are installed.
- Apply the license for release automation.
- Verify that you have available predefined vRealize Automation blueprints, workflows, scripts, or test jobs that perform tasks that the pipelines trigger.
- Verify that the artifacts in the Artifactory server repository are prepopulated so that you can use the Artifact Management capabilities.
- Verify that the applicable plug-ins are registered. See Chapter 3 Registering Plug-in Instances for Release Pipeline.

Procedure

1 Log in to the vRealize Automation appliance.

2 Click **Add** to create a pipeline.

3 Enter an appropriate name and description for the pipeline.

4 Click **Add** to define the input properties for the pipeline.

These properties are required only if you want to pass certain parameters at the time of triggering a pipeline run. You can reference these parameter values across all stages during modeling. The run time values are applied during the run.

Option	Description
Name	Enter a property name. The name cannot include an underscore.
Description	Enter a description for the property.
Value	Enter a property value. For example, ABC-876541.

5 (Optional) Select the check box if you do not want this pipeline to run concurrently.

The pipeline model reuses the same set of virtual machines to deploy a software change. The concurrent run overwrites the deployed change.

6 Add the email addresses of recipients who receive event notifications during the pipeline run.

The email addresses are of the users who have access to the same vRealize Automation appliance.

7 Enter the applicable tags.

A tag is useful in grouping pipeline models or runs.

For example, you can use a tag to filter a pipeline model or run view.

8 Click **Stages** to continue pipeline creation.

9 Click **Add Stage** to add multiple stages to this pipeline.

For example, you can create development, test, and QE stages.

10 Click **Save as Draft** to save the pipeline.

11 Familiarize yourself with the input and output details required to create tasks.

12 Create tasks for every pipeline stage.

You can add multiple tasks to a stage. The Artifact, Custom, Deploy, Provision, and Test tasks are supported.

Certain tasks might depend on tasks that precede them in the workflow. You can move tasks up or down depending on the workflow. Tasks are run sequentially and you can configure a task's input to depend on the output from an earlier task.

Using Pipeline Task Variables

Pipeline variables are runtime variables that are output from a previous task and are available for consumption by other tasks.

vRealize Code Stream uses task output variables and constant variables.

Task Output Variables

You can use the $VAR notation to have a task consume parameters that are available as pipeline runtime variables or the output produced from other tasks. You can select task output variables when you configure Release Automation. When these variables need to be consumed as a direct input to any field in the UI, you can set them as `BUILD_ID = 123`, where 123 is the value of $BUILD_ID. The variable must be consumed as $BUILD_ID.

When this is consumed as vcac-123.war, where 123 is pulled from $BUILD_ID, it must be consumed as vcac-${BUILD_ID}.war.

Constant Variables

Plug-ins can use constant variables as part of input configurations. These variables are replaced at runtime with the values where they are placed.

When the plug-in writers implement the plug-in spec API, it displays ReleasePluginServiceException exception.

```
public ResponseResult execute(ExecutionContext executionContext)
```

The executionContext contains the getTrackingProperties() method, which is populated by default with a unique UUID combination. The combination contains the executionId of the vRealize Orchestrator and the task ID that can be used in the external systems to tie their execution with the execution of the release pipeline.

Pipeline Task Input Properties

You can provide input parameters to the script as a simple key-value pair.

Script Properties

You can use scripts to configure any task. You can customize the host and the script details to any deployment requirement.

Table 4-1. Host and Script Details

Property	Description
Execute on Host	■ User defined hosts. You can enter one or more IP Addresses or DNS hostname or release pipeline variables as one of the valid hosts. ■ Host group variables. A variable represents a group of hosts. ■ Host group filter criteria: You can choose any host group and apply filter criteria to restrict the script to run only on selected hosts. The property uses the following filter criteria. ■ ALL HOSTS. Run the script on all the hosts that the host group variable gives. ■ STARTS WITH. From the host group, filter machines whose names start with the prefix as given by the user. The filter search criteria is case sensitive. ■ ENDS WITH. Filter machines whose names end with the prefix as given by the user. ■ EXACT MATCH. Run the script only on the machine whose name exactly matches the DNS hostname or IP Address.
Host Username and Host Password	Used to make a remote connection to the host. All the hosts must have common credentials.
Execute Script File	Specify the script file name and the path that already resides on the host.
Script Type	The script task supports running a BASH or POWERSHELL script.

Script Input Parameters

The script task allows you to configure input parameters for a script as a simple key-value pair. The parameter name should exactly match the variable that is being used in the script. You can configure an artifact input parameter or a general input parameter to customize the script.

Artifact Input Parameter

These parameters are used to map artifacts to a script task, which enables the script to download artifacts from a repository to the host. The script calls the download URL or the repository URL for an artifact by mapping an artifact input property. For example, VCAC_APP_DOWNLOAD_URL.

To map the artifact input parameters, select Artifact Source, which is a variable that contains a group of artifacts. This variable is generated as part of output from the release pipeline artifact task.

Table 4-2. Parameters

Parameter	Description
Parameter Name	Name of the variable as referred to in the script.
Parameter Value	The artifact name in the parameter value must match the name specified in the artifact task.

General Input Parameter

You can configure other parameters that must be passed to the script as a simple key-value pair. For example, APPLICATION_PORT.

Sample Script

The following example shows a script that configures a vRealize Automation application.

```
$ cat configureAppServer.sh
echo "Configure app server";
echo "VCAC Application Download URL: $VCAC_APP_DOWNLOAD_URL";

wget -O VCAC_APP_DOWNLOAD_URL
echo "Configuring VCAC application";

echo "Starting application on port: $APLICATION_PORT";

MACHINE_IP = ifconfig | sed -En 's/127.0.0.1//;s/.*inet (addr:)?(([0-9]*\.)(3)[0-9]*).*/\2/p'
printf "Application URL: $MACHINE_IP:$APPLICATION_PORT/vcac/" > $SCRIPT_RESPONSE_FILE
```

The script refers to the VCAC_APP_DOWNLOAD_URL environment variable to determine what version of the VCAC artifact to download from the repository. The artifact input parameter to the script should be the VCAC_APP_DOWNLOAD_URL parameter.

```
wget -O VCAC_APP_DOWNLOAD_URL
echo "Configuring VCAC application";
```

For the script to be able to share data with other release pipeline tasks, it must write the contents to a response file. The response file contents are stored in the script output variable.

```
MACHINE_IP = ifconfig | sed -En 's/127.0.0.1//;s/.*inet (addr:)?(([0-9]*\.){3}[0-9]*).*/\2/p'
printf "Application URL: $MACHINE_IP/vcac/" > $SCRIPT_RESPONSE_FILE
```

Script Output

You can configure a script task output variable that is visible to other release pipeline tasks to consume the script response. This variable contains the runtime data, which was written to $SCRIPT_RESPONSE_FILE.

Script Logs

After the script runs, the EXECUTION_ID_FOLDER run folder is created in the <SCRIPT_NAME>_RUN folder . This folder contains the logFile, responseFile, exitStatusFile files, and a clone of the actual script that is being run. For example, script_params.txt, script_log.txt, script_response.txt.

The run folder is in the following locations.

Table 4-3. Run Folder Locations

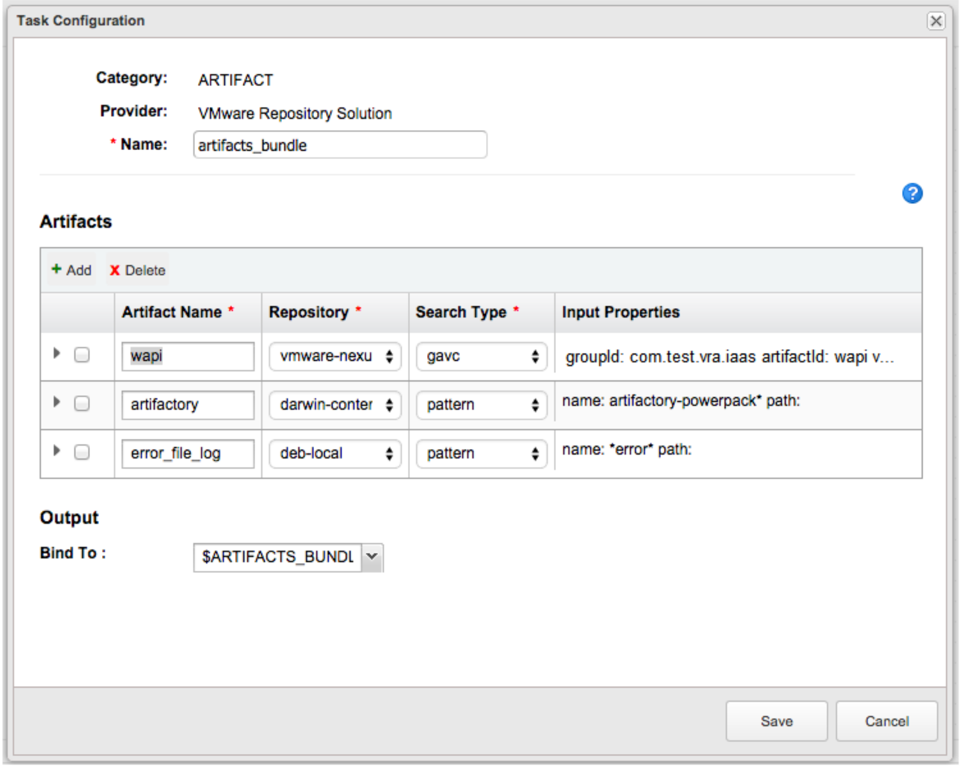
Folder	Location
Bash script	/var/tmp/SCRIPT_NAME_RUN/EXECUTION_ID_FOLDER
Powershell Script	C:\Users\<USER_NAME>\AppData\Local\Temp\SCRIPT_NAME_RUN/EXECUTION_ID_FOLDER

Configure an Artifact Task

An artifact task lets you search the library of binaries in different repositories that you can deploy on a virtual machine.

When you include an artifact task in a stage, you can run a pipeline execution every time you develop new code that affects that artifact. The search output parameter from varied source repositories is always the same, which includes a repository name, a download URL, and size information, if available.

Figure 4-1. Sample Artifact Task



Category: ARTIFACT

Provider: VMware Repository Solution

* Name: artifacts_bundle

Artifacts

	Artifact Name *	Repository *	Search Type *	Input Properties
☐	wapi	vmware-nexu	gavc	groupId: com.test.vra.iaas artifactId: wapi v...
☐	artifactory	darwin-conter	pattern	name: artifactory-powerpack* path:
☐	error_file_log	deb-local	pattern	name: *error* path:

Output

Bind To : $ARTIFACTS_BUNDL

Prerequisites

- Verify that a pipeline is available. See Create a Pipeline.
- Verify that an Artifactory server is configured. See Register an Artifactory Server for Artifact Management.

Procedure

1 Select **Release Automation**.

2 Select the pipeline to configure from the **Pipeline** tab.

3 Select **Edit** > **Stages**.

4 Select **Add Task**.

5 Select **Artifact** from the Category drop-down menu.

6 Assign the Provider from the drop-down menu.

The provider is the repository that you configured.

7 Enter a name for the artifact task and click **OK**.

8 Select the new task from the stage column.

9 Confirm to save the pipeline for configuration.

10 Select an **Artifactory repository** from the drop-down menu.

11 Determine a search type and click **Validate** to verify that the artifact exists in the repository.

The search type depends on your repository and artifact. If the search shows that an artifact does not exist, verify that the search parameters are accurate.

For example, you can search for the Maven artifacts using the gavc search type.

Search Type	Description
gavc	Provide the group ID, artifact ID, version ID, and classifier parameters for the search. ■ The group ID is the org that published the artifact. For example, org.springframework. ■ The artifact ID is the identifier of the artifact. For example, travel. ■ The version ID parameter supports the LATEST and version-SNAPSHOT keywords. For example, enter `4.1-SNAPSHOT` to get the snapshot of a version. The classifier supports the wild card character asterisk (*). For example, release*, *release*, or *.
pattern	Provide the repository name and path parameters for the search. ■ The name parameter supports the wild card character asterisk (*). For example, test*, test.*, *.jar, or *. ■ The path parameter supports the wild card character asterisk (*) for local repositories. For example, path/*/release searches under the path directory in the /test/release and /dev/release folders for the artifacts only in those folders. Each wildcard (*) represents one level in the folder structure.
properties	Defined in the Artifactory server to tag artifacts with custom user properties. These properties can be any string values. An artifact can have multiple properties and these properties can have multiple values. You can use these properties instead of the actual group ID, version, or specific path to locate an artifact in the repository. For example, the Property field can be artifactory.licenses and the Value field can be BSD .
build	Builds cannot be created in the Artifactory user interface. A build must be pushed from your CI server to the Artifactory user interface. ■ The build Name is the name of the job that is run on your CI system. For example, Jenkins-release. ■ The build Number supports the LATEST or a build status keyword . For example, you can search for builds with the Prod status. ■ The name lets you filter an artifact from the list of artifacts in a specific build. For example, public-API.jar or public-*.jar

12 Provide a unique name prefixed with a dollar sign ($) to bind to for the Output Parameters.

13 (Optional) Click **Add** to include another artifact and configure the parameters.

14 Click **Save**.

Configure a Custom Script Task

You can use a custom script task to run a script that resides on a remote host. After the script runs, you can monitor the script progress and capture the script response, which you can pass as input to other release pipeline tasks.

Prerequisites

- Verify that you defined and configured the plug-ins. An instance of this plug-in must be created in vRealize Orchestrator to enable the plug-ins in the pipeline. See Chapter 3 Registering Plug-in Instances for Release Pipeline.
- Verify that an artifact task is configured. See Configure an Artifact Task.
- Verify that your script exists on a remote host that is configured in the script task. The script must have executable permission for a remote user to run it.
- Verify that the folder where the script exists has permission to allow a file to be created.

Procedure

1 Select **Release Automation**.

2 Select the pipeline to configure from the **Pipeline** tab.

3 Select **Edit** > **Stages**.

4 Select **Add Task**.

5 Select **Custom** from the Category drop-down menu.

6 Select **Custom Script** from the Provider drop-down menu.

7 Enter a name for this custom task and click **OK.**

8 Select the new task from the stage column.

9 Confirm to save the pipeline for configuration.

10 Enter the script properties.

Option	Description
Execute on Host	Select the host type. ■ User defined host: IP Address of the host. You can enter multiple IP Addresses and pipeline input variables. For example, the input variable machine_ip can be 10.116.7.1. The user defined host can also have $machine_ip along with static hostname and IP Address. ■ Variables: A variable output of vRealize Automation provisioning.
User credentials	Enter the Host User name and Host Password.

Option	Description
Execute Script File path	Enter the Execute Script File path for the `.sh` script file. The host files must have a common location and the same file name for all the hosts.
Script Type	Select the supported Bash or PowerShell script type.

11 Enter the Script Input parameters.

a Select the Artifact Source from the drop-down menu.

b Click **Add** to define the Artifact input parameters.

c Click **Add** to define the General Input Parameters.

12 Provide a unique name prefixed with a dollar sign ($) to bind to for the Output Parameters.

13 Click **Save**.

Example: Task Output Format for a Custom Script Task

The following task output format is an example for a custom script task.

```
[
  {
    "description": "Host Response",
    "name": "10.110.15.205",
    "value": "Sample Response of Linux Host 01. Sample Response of Linux Host 01.",
    "type": "STRING"
  },
  {
    "description": "Host Response",
    "name": "10.110.15.206",
    "value": "Sample Response of Linux Host 02. Sample Response of Linux Host 02.",
    "type": "STRING"
  }
]
```

Configure a Custom Task

A custom task allows any custom activity that you can start with a vRealize Orchestrator workflow or script. This activity can also be a manual approval.

Prerequisites

Verify that you created a workflow and tagged it with the CUSTOM keyword in the vRealize Orchestrator Workflow Designer.

Procedure

1 Select **Release Automation**.

2 Select the pipeline to configure from the **Pipeline** tab.

3 Select **Edit** > **Stages**.

4. Select **Add Task**.
5. Select **Add Task**.
6. Select **Custom** from the Category drop-down menu.
7. Select **Custom Workflow** from the Provider drop-down menu.
8. Enter a name for the custom task and click **OK**.
9. Select the new task from the stage column.
10. Confirm to save the pipeline for configuration.
11. Select the Workflow Name from the drop-down menu.
12. Enter the Input Parameters.

    For example, enter **`vsphere.local\administrators`** for the Task Group DN Name and enter the task details.

13. Provide a unique name prefixed with a dollar sign ($) to bind to for the Output Parameters.
14. Click **Save**.

Example: Task Output Format for a Custom Workflow Task

The following task output format is an example for a custom workflow task.

```
[{
                "name": "result",
                "type": "STRING",
                "description": "Result of workflow run.",
                "value": ""
},
{
                "name": "message",
                "type": "STRING",
                "description": "Message",
                "value": ""
}]
```

Configure a Deploy Task

A deploy tasks deploys or updates the artifacts to the machines.

For example, the plug-in provider for this task is a script. This script can be a Bash or PowerShell script.

Prerequisites

Verify that you have created a valid script .

Procedure

1. Select **Release Automation**.
2. Select the pipeline to configure from the **Pipeline** tab.

3 Select **Edit** > **Stages**.

4 Select **Add Task**.

5 Select **Deploy** from the Category drop-down menu.

6 Select **Custom Script** from the Provider drop-down menu.

7 Enter a name for the deploy task and click **OK**.

8 Select the new task from the stage column.

9 Confirm to save the pipeline for configuration.

10 Enter the script properties.

Option	Description
Execute on Host	Select the host type. ■ User defined host: IP Address of the host. You can enter multiple IP Addresses. ■ Variables: A variable output of vRealize Automation provisioning.
User credentials	Enter the Host User name and Host Password.
Execute Script File path	Enter the Execute Script File path for the `.sh` script file. The host files must have a common location and the same file name for all the hosts.
Script Type	Select the supported Bash or PowerShell script type.

11 Enter the Script Input parameters.

a Select the Artifact Source from the drop-down menu.

b Click **Add** to define the Artifact input parameters.

c Click **Add** to define the General Input Parameters.

12 Provide a unique name prefixed with a dollar sign ($) to bind to for the Output Parameters.

13 Click **Save**.

Example: Task Output Format for a Custom Script Task

The following task output format is an example for a custom script task.

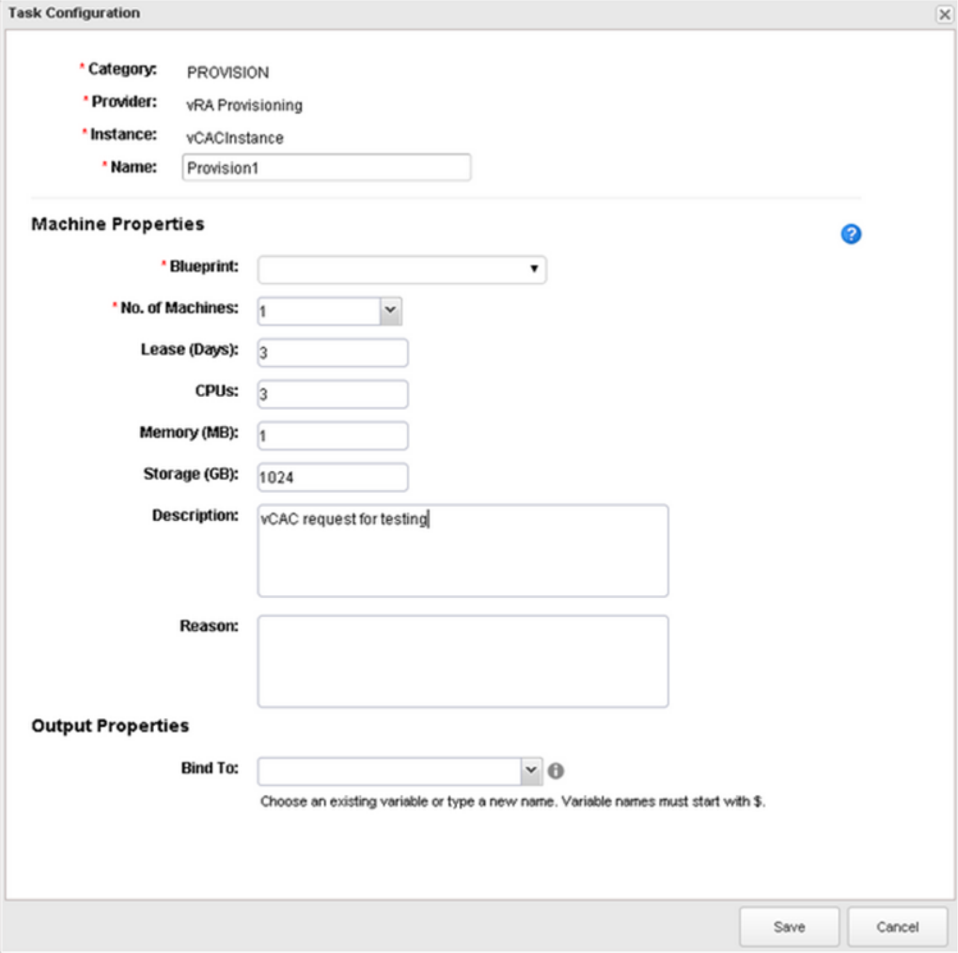
```
[
  {
    "description": "Host Response",
    "name": "10.110.15.205",
    "value": "Sample Response of Linux Host 01. Sample Response of Linux Host 01.",
    "type": "STRING"
  },
  {
    "description": "Host Response",
    "name": "10.110.15.206",
    "value": "Sample Response of Linux Host 02. Sample Response of Linux Host 02.",
    "type": "STRING"
  }
]
```

Configure a Provision Task

A provision task provisions the machines. The plug-in provider for this task is vRealize Automation. Based on the provider, specific configuration is required. You can create machines from a single machine blueprint for a provision task. The task output is an array of machines.

Assign the catalog item to a service. The service must be added to the entitlement for the Business group proxy user.

Figure 4-2. Sample Provision Task



Task Configuration

* Category: PROVISION

* Provider: vRA Provisioning

* Instance: vCACInstance

* Name: Provision1

Machine Properties

* Blueprint:

* No. of Machines: 1

Lease (Days): 3

CPUs: 3

Memory (MB): 1

Storage (GB): 1024

Description: vCAC request for testing

Reason:

Output Properties

Bind To:

Choose an existing variable or type a new name. Variable names must start with $.

Prerequisites

- Verify that you used vRealize Orchestrator to register the vRealize Automation server. See Register a vRealize Automation Server.
- Verify that you completed the following tasks in vRealize Automation.
  - Choosing an Endpoint Scenario see *IaaS Configuration for Virtual Platforms*.

- Create a Fabric Group see *IaaS Configuration for Virtual Platforms*.
- Create a Business Group see *IaaS Configuration for Virtual Platforms*.
- Create a Reservation see *IaaS Configuration for Virtual Platforms*.
- Create a Reservation Policy see *IaaS Configuration for Physical Machines*.
- Create a Network Profile see *IaaS Integration for Multi-Machine Services*.
- Create a Blueprint see *IaaS Configuration for Virtual Platforms*.
- Publish a Blueprint see *IaaS Configuration for Virtual Platforms*.

Procedure

1 Select **Release Automation**.

2 Select the pipeline to configure from the **Pipeline** tab.

3 Select **Edit** > **Stages**.

4 Select **Add Task**.

5 Select **Provision** from the Category drop-down menu.

6 Select **vRA Provisioning** from the Provider drop-down menu.

7 Select the vRealize Automation server you registered in vRealize Orchestrator from the drop-down menu.

8 Enter a name for the provision task and click **OK**.

9 Select the new task from the stage column.

10 Confirm to save the pipeline for configuration.

11 Enter the machine properties.

Option	Description
Blueprint	Select the published vRealize Automation blueprint.
Number of Machines	Enter the number of machines to be provisioned that are pipeline or global variables.
(Optional) Configuration Details	Enter the Lease (Days), CPUs, Memory (MB), Storage (GB), and Description.

12 Provide a unique name prefixed with a dollar sign ($) to bind to for the Output Parameters.

This variable contains details of the machines successfully provisioned by this task. You can use this variable in succeeding tasks to retrieve the machine details.

13 Click **Save**.

Example: Task Output Format for a Provisioning Task

The following task output format is an example for a provisioning task.

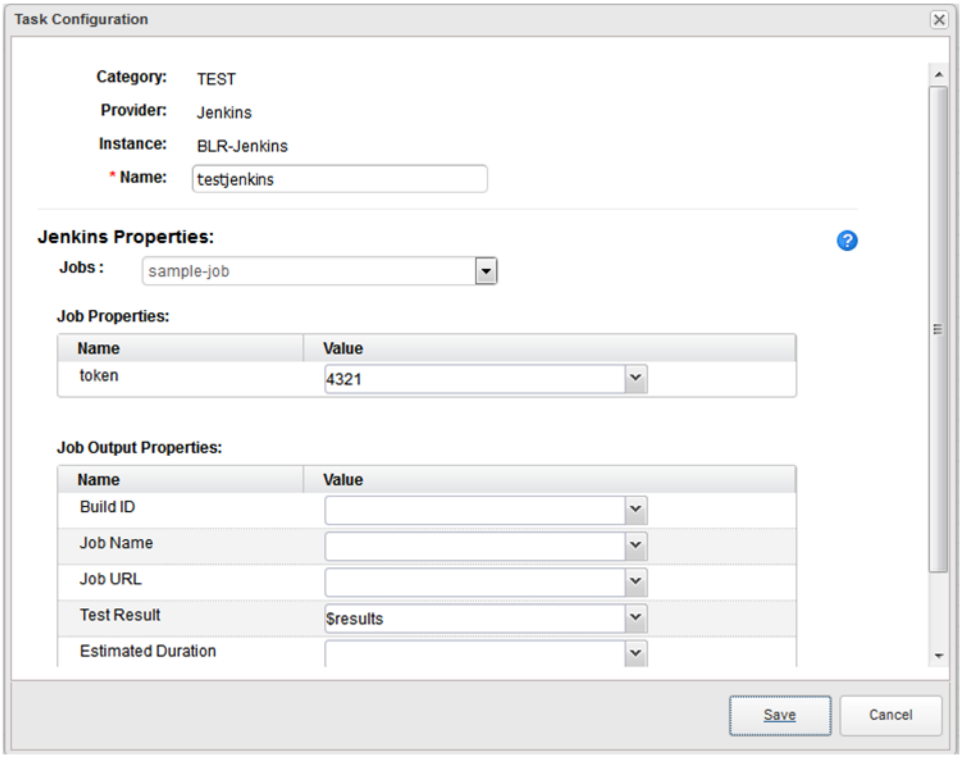
```
[
    {
        "name": "vcac-prov01",
        "value": {
            "memory": 1024,
            "machineId": "f9ee3f71-c5d0-4138-9520-24e15e376d13",
            "hostIp": ["10.72.12.56"],
            "cpu": 1,
            "endLease": 1410478627000,
            "storageSize": 4,
            "startLease": 1410392227000,
            "blueprintId":" f9ee3f71-c5d0-4138-3476-24e15e376f36",
            "provider":"iaas-service"
        },
        "type": "MACHINE"
    }
]
```

Configure a Test Task

A test task lets you test the deployment or application.

For example, the plug-in provider for this task is Jenkins. The task output is a test result. If you configure a Jenkins test job to fail if test failures occur, then the release pipeline also fails. The Jenkins plug-in version 1.519 or later is supported.

Figure 4-3. Sample Test Task



Category: TEST
Provider: Jenkins
Instance: BLR-Jenkins
* Name: testjenkins

Jenkins Properties:
Jobs : sample-job

Job Properties:

Name	Value
token	4321

Job Output Properties:

Name	Value
Build ID	
Job Name	
Job URL	
Test Result	$results
Estimated Duration	

Prerequisites

- Verify that you used vRealize Orchestrator to register the Jenkins server. See Chapter 3 Registering Plug-in Instances for Release Pipeline
- Verify that you modified the appropriate jobs in the Jenkins server with the following parameters.
  - Create a parameter with the string vRCSTestExecutionId.
  - (Optional) Token - Select the trigger builds remotely. Enter an authentication token that is the input for the pipeline.

Procedure

1. Select **Release Automation**.
2. Select the pipeline to configure from the **Pipeline** tab.
3. Select **Edit** > **Stages**.
4. Select **Add Task**.
5. Select **Test** from the Category drop-down menu.
6. Select **Jenkins** from the Provider drop-down menu.
7. Select the Jenkins server you registered in vRealize Orchestrator from the drop-down menu.
8. Enter a name for the test task and click **OK**.
9. Select the new task from the stage column.

10. Confirm to save the pipeline for configuration.
11. Select the Jenkins job from the Jobs drop-down menu.

    Only the jobs that have the two parameters mentioned in the prerequisites are available in the menu.
12. Enter the Jenkins job properties and job output properties.
13. Click **Save**.

Example: Output Format for a Test Task

The following task output format is an example for a test task.

```
[
        {
            "name": "buildId",
            "type": "STRING",
            "description": "Build Id",
            "value": "4"
        },
        {
            "name": "jobUrl",
            "type": "STRING",
            "description": "Job Url",
            "value": "http://10.112.75.139:8080/job/dummy/4"
        },
        {
            "name": "estimatedDuration",
            "type": "NUMBER",
            "description": "Estimation time to complete the build",
            "value": "2332343"
        },
        {
            "name": "jobName",
            "type": "STRING",
            "description": "Job Name",
            "value": "dummy"
        },
        {
            "name": "testResult",
            "type": "JSON",
            "description": "Job Name",
            "value": {
                "totalCount": 40,
                "skipCount": 0,
                "failureCount": 0,
                "successCount": 40
            }
        }
    ]
```

Configure Gating Rules

Gating rules are a set of criteria that each stage must pass to proceed to the subsequent stage. You can configure gating rules based on your requirement for a pipeline.

If the gating rule is not configured, then the process proceeds to the next stage regardless of the outcome of tasks in the current stage. The workflows are tagged as GATING_RULE and are listed in the drop-down menu.

A workflow is depicted as a sequence of operations. The workflow consists of an orchestrated and repeatable pattern of business activity enabled by the systematic organization of resources into processes that transform materials, provide services, or process information.

The default workflows are Approval and Test Acceptance Threshold.

Table 4-4. Default Workflows

Workflow	Description
Approval	This workflow is tagged with a GATING_RULE tag. You must provide the approval group DN value and the approval message for a stage to continue or stop. The appropriate user receives an email to approve or reject this workflow. After the user submits or rejects the notification that appears in My Inbox, the workflow either continues or stops.
Test Acceptance Threshold	This workflow is tagged with a GATING_RULE tag. You must provide the threshold percentage value and the test result for a task to succeed the gating rule to the next level. This workflow accepts the test result from a Jenkins test task. Based on the configured threshold for passed tests criteria, it allows the pipeline run to proceed to the subsequent stage.

Prerequisites

- Verify that the last task of any stage is a test, custom script, or custom workflow.
- Verify that a custom script output is converted to create a JSON and then passed to the gating rule.

  The Jenkins output can be passed directly to the gating rule, as the following example shows.

  ```
  {
  "totalCount": 40,
  "skipCount": 0,
  "failureCount": 0,
  "successCount": 40
  }
  ```

Procedure

1 Select **Release Automation**.

2 Select the pipeline to configure from the **Pipeline** tab.

3 Select **Edit** > **Stages**.

4 Click the Gating Rule icon (🚪) next to a stage.

5 Select **If outcome of a vCloud Orchestrator workflow is successful** to create the gating rule.

6 Select the appropriate workflow from the drop-down menu.

   a Enter the approval group DN value, `vsphere.local\administrators`, in the Approval text box.

      You can also edit the approval message. DN is any valid DN name that is configured in the identity store tied to vRealize Automation.

   b Enter the threshold percentage and the test result value in the Test Acceptance Threshold text box.

7 Click **Save**.

Activate and Run a Pipeline

You can run a pipeline that is activated. After you activate the pipeline, pipeline modeling is complete and you can run it.

You can view a pipeline run on the **Pipeline Executions** tab. The draft state signifies that the Release Manager is still modeling the pipeline.

Multilevel information is provided during the pipeline run.

- Level 1.Status at pipeline level which is the description of the current activity.
- Level 2. Status at the task or stage level that displays the current task progress and the executed task status.
- Level 3. Provide detailed information at task level.

Prerequisites

- Verify that a pipeline is created in a draft state. See Create a Pipeline
- Verify that the pipeline is activated. A draft state pipeline cannot be run.
- Verify that predefined vRealize Automation blueprints, workflows, scripts are created, and test jobs that perform tasks that the pipeline can trigger.
- Prepopulate artifacts in the Artifactory server repository to use the vRealize Code Stream Artifact Management capabilities.

Procedure

1 Select **Release Automation**.

2 Select the pipeline to configure from the **Pipeline** tab.

3 Click **Activate** to create a vRealize Orchestrator workflow.

   Every time a pipeline is activated, a new version of the pipeline is created.

   A list of the available pipelines appears.

4 Select the pipeline.

5 Click **Execute**.

   The Execute Pipeline dialog box appears.

6 Enter the description and properties.

7 Click **OK** to run a release pipeline.

What to do next

You can track the status of the pipeline and troubleshoot failed pipelines. See Chapter 5 Using the Experimental Release Dashboard Feature.

Using the Experimental Release Dashboard Feature

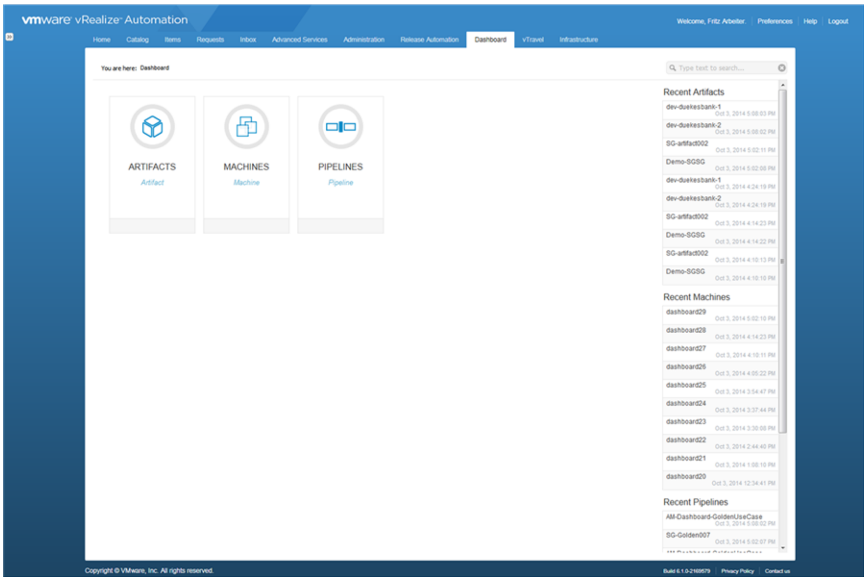
5

When a release pipeline process is complete, the data for the artifacts, machines, and pipelines is available in the release dashboard, which is an experimental feature. The release dashboard home page summarizes the total number of artifacts and machines. It also lists the number of successful, in progress, or failed pipelines.

As a Release Dashboard user you can view artifacts, machines, and pipelines, and track the cross dependencies between these resource types to facilitate troubleshooting when deployments fail or exceed the expected deployment time threshold. See Roles and Responsibilities of Personas.

The Artifacts resource section lists all of the artifacts deployed by the executed pipelines. When you select an artifact, the associated details and the details of the parent pipelines' execution instances that deployed this artifact instance are displayed. The Machines resource section lists all of the machines deployed through an executed pipeline by using the vRealize Automation machine provisioning. The Pipeline resource section lists each instance of the executed pipeline.

On the right side of the home page, you can view the recently updated resources, a summary of the pipeline details, and the status of the related machines and artifacts.



Recent Artifacts

dev-duekesbank-1	Oct 3, 2014 5:08:03 PM
dev-duekesbank-2	Oct 3, 2014 5:08:02 PM
SG-artifact002	Oct 3, 2014 5:02:11 PM
Demo-SGSG	Oct 3, 2014 5:02:08 PM
dev-duekesbank-1	Oct 3, 2014 4:24:19 PM
dev-duekesbank-2	Oct 3, 2014 4:24:19 PM
SG-artifact002	Oct 3, 2014 4:14:23 PM
Demo-SGSG	Oct 3, 2014 4:14:22 PM
SG-artifact002	Oct 3, 2014 4:10:13 PM
Demo-SGSG	Oct 3, 2014 4:10:10 PM

Recent Machines

dashboard29	Oct 3, 2014 5:02:10 PM
dashboard28	Oct 3, 2014 4:14:23 PM
dashboard27	Oct 3, 2014 4:10:11 PM
dashboard26	Oct 3, 2014 4:05:22 PM
dashboard25	Oct 3, 2014 3:54:47 PM
dashboard24	Oct 3, 2014 3:37:44 PM
dashboard23	Oct 3, 2014 3:30:08 PM
dashboard22	Oct 3, 2014 2:44:40 PM
dashboard21	Oct 3, 2014 1:08:10 PM
dashboard20	Oct 3, 2014 12:34:41 PM

Recent Pipelines

AM-Dashboard-GoldenUseCase	Oct 3, 2014 5:08:02 PM
SG-Golden007	Oct 3, 2014 5:02:07 PM

To troubleshoot a failed pipeline, you can navigate to the Pipelines page and click the failed pipeline or navigate to the Machines page to identify the related machines that might have caused the deployment to fail. A user can view each artifact in the pipeline and associated pipelines to verify that the pipeline did not fail because of that artifact.

Managing Release Automation 6

After you execute a release pipeline, you can monitor the status from the Dashboard home page. You can also delete a task, create a duplicate pipeline, or delete a pipeline from the **Release Automation** tab.

You can perform the following tasks to manage release automation.

This chapter includes the following topics:

Delete a Task

You can delete any task in a stage that is not required in the pipeline. Deleting a task might affect the subsequent tasks that refer to the task's output.

Prerequisites

Verify that you have an existing task. See Create a Pipeline.

Procedure

1 Select a pipeline on the **Pipelines** tab.

2 Click **Edit**.

3 On the Edit Pipeline page, click **Stages**.

4 Click the gear icon (⚙) next to the task in a stage.

5 Click **Delete**.

Copy a Pipeline

Copying or cloning a pipeline creates the replica of a pipeline with a different pipeline name. This replication creates a pipeline without affecting the existing pipeline model. The configuration of the source and target pipelines does not change.

Prerequisites

Verify that you have created a pipeline. See Create a Pipeline.

Procedure

1. Select the pipeline to copy on the **Pipelines** tab.
2. Click **Copy**.

   The Copy Pipeline window appears.
3. Enter an appropriate name for the pipeline replica.
4. Enter the description for the pipeline .
5. Click **OK**.

   A copy of the pipeline is created and is listed on the **Pipelines** tab. You can edit this pipeline to make further modifications.

Delete a Pipeline

Deleting a pipeline removes the pipeline from the pipeline model.

Prerequisites

Verify that you have an existing pipeline. See Create a Pipeline.

Procedure

1. Select the pipeline to delete on the **Pipelines** tab.
2. Click **Delete**.

   A confirmation message appears.
3. Click **Yes** to delete the selected pipeline.

Call vRealize Code Stream REST API from Script and Java 7

You can call the REST APIs from Script and Java to query and run pipelines programmatically. The Script and Java codes do not have any dependencies.

REST API from Script

The following example is script code that you call from the vRealize Code Stream API.

```
#!/bin/bash

echo "#### Sample script to query and trigger a pipeline execution ####"
#Server host address refers to the host on which Code Stream server is setup. Eg: codestream.abc.com
read -p "vRealize Code Stream Server Host: " server_host

#user name and password with which you login on Code Stream server Eg: jane.doe@abc.com
read -p "Username: " username
read -p "Password: " password

#tenant name can be obtained from your system administrator if not known already
read -p "Tenant: " tenant
echo "-------------------------------------------------------"

#fetch the pipeline details and subsequently trigger an execution
#enter the pipeline name for which you want to trigger an execution
read -p "Release pipeline name:" pipeline_name

#pipeline param JSON is the input required for the pipeline execution. for a single pipeline parameter
'token', the JSON input would look like:
# Eg: {"description":"test run","pipelineParams":[{"name":"token","type":"STRING","value":"4321"}]}
read -p "Enter the pipeline param JSON:" pipeline_params

#A SSO token is required to make any calls to the Code Stream server. Token can be obtained easily by
passing the credentials as follows
host_url="https://$server_host/identity/api/tokens"
response=$(curl -s -X POST -H 'Content-Type: application/json' -H 'Accept: application/json' --
insecure -d '{"username": "'"$username"'", "password": "'"$password"'", "tenant": "'"$tenant"'"}'
$host_url)
#token can be extracted from the JSON response as follows
token=`echo $response | sed -n 's/.*"id":"\([^}]*\)",.*}/\1/p'`

#with the token obtained, subsequent calls can be made to the code stream server (a token has an
expiry so renewal might be required if the same token is reused beyond expiry)
```

```
pipeline_fetch_url="https://$server_host/release-management-service/api/release-pipelines?name=
$pipeline_name"
response=$(curl -s -X GET -H "Content-Type: application/json" -H "Accept: application/json" -H
"Authorization: Bearer $token" -k $pipeline_fetch_url)
pipeline_id=`echo $response | sed -n 's/.*"id":"\([^"]*\)",.*stages.*/\1/p'`
#echo "pipeline id: $pipeline_id"

#with the pipeline id, an execution can be triggered as follows
execute_pipeline_url="https://$server_host/release-management-service/api/release-
pipelines/$pipeline_id/executions"
echo "executing pipeline:$pipeline_name :[$pipeline_id]"
response=$(curl -s -X POST -H "Content-Type: application/json" -H "Accept: application/json" -H
"Authorization: Bearer $token" -k -d "$pipeline_params" $execute_pipeline_url)
echo "Response to execute pipeline => $response"
```

REST API from Java

The following example is Java code called from the vRealize Code Stream API. The Java code must be packaged with open source dependencies.

```
package samples;

import com.google.gson.Gson;
import org.apache.http.Header;
import org.apache.http.HttpResponse;
import org.apache.http.client.HttpClient;
import org.apache.http.client.methods.HttpGet;
import org.apache.http.client.methods.HttpPost;
import org.apache.http.conn.ClientConnectionManager;
import org.apache.http.conn.scheme.Scheme;
import org.apache.http.conn.scheme.SchemeRegistry;
import org.apache.http.conn.ssl.SSLSocketFactory;
import org.apache.http.conn.ssl.TrustStrategy;
import org.apache.http.entity.StringEntity;
import org.apache.http.impl.client.DefaultHttpClient;
import org.apache.http.impl.conn.BasicClientConnectionManager;
import org.apache.http.message.BasicHeader;
import org.apache.http.protocol.HTTP;

import java.io.BufferedReader;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.security.cert.CertificateException;
import java.security.cert.X509Certificate;
import java.util.ArrayList;
import java.util.List;

/**
* Sample Java class to invoke REST APIs on Release pipeline
*
*/
public class PipelineApiHelper {
```

```
private String username;
private String password;
private String tenant;
private String serverHost;

private Gson gson = new Gson();

private static String FETCH_TOKEN_URL_TEMPLATE = "https://%s/identity/api/tokens";
private static String TRIGGER_EXECUTION_URL_TEMPLATE = "https://%s/release-management-
service/api/release-pipelines/%s/executions";
private static String FETCH_PIPELINE_INFO_URL_TEMPLATE = "https://%s/release-management-
service/api/release-pipelines?name=%s";

private static ClientConnectionManager connectionManager;

/**
* Sample helper class to fetch pipeline info and trigger an execution
*
* @param serverhost Code Stream server host name
* @param username login credentials
* @param password login credentials
* @param tenant tenant id for the user
*/
public PipelineApiHelper(String serverhost, String username, String password, String tenant) {
this.username = username;
this.password = password;
this.tenant = tenant;
this.serverHost = serverhost;
}

/**
* Trigger a pipeline execution
*
* @param pipelineName name of the pipeline to execute
* @param pipelineParamsJson JSON string for the pipeline execution
* pipeline param JSON is the input required for the pipeline execution.
* for a single pipeline parameter 'token', the JSON input would look like:
* Eg: {"description":"test run","pipelineParams":[{"name":"token","type":"STRING","value":"4321"}]}
*/
public void triggerPipelineExecution(String pipelineName, String pipelineParamsJson) {
//A SSO token is required to make any calls to the Code Stream server. Token can be obtained easily by
passing the credentials as follows
final String token = fetchToken();
List<Header> headers = getCommonHeaders();
headers.add( new BasicHeader("Authorization", "Bearer " + token));
try {
//fetch the pipeline id
InputStream pipelineRespStream = getRequest(String.format(FETCH_PIPELINE_INFO_URL_TEMPLATE,
serverHost, pipelineName),
headers);
BufferedReader bufferedReader = new BufferedReader(new InputStreamReader(pipelineRespStream));
//response contains the pipeline info (containing the id)
PipelineInfoWrapper pInfo = gson.fromJson(bufferedReader, PipelineInfoWrapper.class);

//trigger the pipeline execution with the pipeline id and pipeline params
```

```
InputStream is = postRequest(String.format(TRIGGER_EXECUTION_URL_TEMPLATE, serverHost,
pInfo.getContent().get(0).getId()),
headers,
pipelineParamsJson);
bufferedReader.close();
bufferedReader = new BufferedReader(new InputStreamReader(is));

String response;
System.out.println("Response:");
while ((response = bufferedReader.readLine()) != null) {
System.out.print(response);
}
System.out.println("==============================================================");
} catch (Exception ex) {
System.out.println("Error triggering pipeline: " + ex.getMessage());
}
System.out.println("Pipeline execution triggered !");
}

/**
* Fetch SSO Token from Identity server
* @return token
*/
private String fetchToken() {
//POST username,password,tenant to fetch token API
TokenRequest tokenRequest = new TokenRequest();
tokenRequest.setUsername(username);
tokenRequest.setPassword(password);
tokenRequest.setTenant(tenant);
String token = null;

try {
InputStream is = postRequest(String.format(FETCH_TOKEN_URL_TEMPLATE, serverHost),
getCommonHeaders(),
gson.toJson(tokenRequest));
BufferedReader bufferedReader = new BufferedReader(new InputStreamReader(is));
TokenResponse tokenResponse = gson.fromJson(bufferedReader, TokenResponse.class);
if (tokenResponse != null) {
token = tokenResponse.getId();
}

} catch (Exception ex) {
System.out.println("Error fetching token: " + ex.getMessage());
}
return token;
}

/**
* Construct the common headers required for the API calls
*
* @return
*/
private List<Header> getCommonHeaders() {
List<Header> headers = new ArrayList<Header>(){{
add(new BasicHeader(HTTP.CONTENT_TYPE, "application/json"));
add(new BasicHeader("Accept", "application/json"));
}};

return headers;
}

/**
* Helper method for Http POST request
* @param url
* @param headers
* @param requestBody
* @return
* @throws Exception
*/
private InputStream postRequest(String url, List<Header> headers, String requestBody) throws Exception
{
HttpClient httpClient = new DefaultHttpClient(getConnManager());
HttpPost post = new HttpPost(url);

post.setEntity(new StringEntity(requestBody));
Header[] headersArray = headers.toArray(new Header[headers.size()]);
post.setHeaders(headersArray);
HttpResponse response = httpClient.execute(post);
if (null != response && response.getStatusLine().getStatusCode()/100 == 2) {
return response.getEntity().getContent();
} else {
return null;
}
}

/**
* Helper method for http GET request
* @param url
* @param headers
* @return
* @throws Exception
*/
private InputStream getRequest(String url, List<Header> headers) throws Exception {
HttpClient httpClient = new DefaultHttpClient(getConnManager());
HttpGet get = new HttpGet(url);

Header[] headersArray = headers.toArray(new Header[headers.size()]);
get.setHeaders(headersArray);
HttpResponse response = httpClient.execute(get);
if (null != response && response.getStatusLine().getStatusCode()/100 == 2) {
return response.getEntity().getContent();
} else {
return null;
}
}

/**
* Construct a connection manager
* Note: This sample method ignores the SSL certificates. Ignoring them may not be something that you
intend.
```

```
* @return
* @throws Exception
*/
private static ClientConnectionManager getConnManager() throws Exception {
if (connectionManager == null) {
SSLSocketFactory sslSocketFactory = new SSLSocketFactory(new TrustStrategy() {
public boolean isTrusted(
final X509Certificate[] chain, String authType) throws CertificateException {
return true;
}
});
Scheme httpsScheme = new Scheme("https", 443, sslSocketFactory);
SchemeRegistry schemeRegistry = new SchemeRegistry();
schemeRegistry.register(httpsScheme);

connectionManager = new BasicClientConnectionManager(schemeRegistry);
}
return connectionManager;
}

public static void main(String[] args) {
PipelineApiHelper helper = new PipelineApiHelper("test.com", "test@test.com", "password", "sample");
helper.triggerPipelineExecution("PipelineName", "{\"id\":\"\",\"description\":\"test
run\",\"pipelineParams\":[{\"name\":\"token\",\"type\":\"STRING\",\"value\":\"4321\"}]}");
}
}

class TokenResponse {
String expires;
String id;
String tenant;

public String getId() {
return id;
}

public String getExpires() {
return expires;
}

public String getTenant() {
return tenant;
}

public void setExpires(String expires) {
this.expires = expires;
}

public void setId(String id) {
this.id = id;
}

public void setTenant(String tenant) {
this.tenant = tenant;
}
```

```
}

class TokenRequest {
String username;
String password;
String tenant;

public String getUsername() {
return username;
}

public void setUsername(String username) {
this.username = username;
}

public String getPassword() {
return password;
}

public void setPassword(String password) {
this.password = password;
}

public String getTenant() {
return tenant;
}

public void setTenant(String tenant) {
this.tenant = tenant;
}
}

class PipelineInfoWrapper {
List<PipelineInfo> content;

public List<PipelineInfo> getContent() {
return content;
}

public void setContent(List<PipelineInfo> content) {
this.content = content;
}
}

class PipelineInfo {
String id;

public String getId() {
return id;
}
```

```
public void setId(String id) {
this.id = id;
}
}
```

Collecting Logs to Troubleshoot Failures

8

vRealize Code Stream creates logs to aid in troubleshooting. You can use the log to find and correct some problems on your own.

If a technical support representative requests more logs, you can retrieve them from the file system of the vRealize Automation appliance.

Collect Logs from the vRealize Automation Appliance

You can access the `catalina.out` log file from the vRealize Automation appliance.

Prerequisites

Verify that you have credentials for logging in to the Linux-based virtual machine with root privileges.

Procedure

1. Log in to the virtual machine.
2. Navigate to the `/var/lib/vcac/server/logs` directory.
3. Retrieve the logs from the `catalina.out` file and troubleshoot the problem.