# Table of Contents

Lab Overview - HOL-1821-04-CMP - vRealize Code Stream - DevOps Solutions ............... 2  
Lab Guidance .................................................................................................................. 3  
Module 1 - Introduction to vRealize Code Stream (30 minutes) ..................................... 10  
  Introduction..................................................................................................................... 11  
  Introducing vRealize Code Stream ............................................................................... 12  
  Architecture Concepts................................................................................................. 14  
  Introduction to Lab Scenario ......................................................................................... 16  
  Building a Simple Application Release Pipeline ......................................................... 18  
  Executing a Simple Application Release Pipeline ..................................................... 31  
  Conclusion....................................................................................................................... 37  
Module 2 - Using vRealize Code Stream for Application Release Automation (60 minutes) ......................................................................................................................... 38  
  Introduction to Lab Scenario ......................................................................................... 39  
  Power on Brownfield Server ......................................................................................... 41  
  Building a "Staging" Stage for QA .............................................................................. 44  
  Build a Gating Rule........................................................................................................ 54  
  Build an Approval Policy and Move to Production ....................................................... 61  
  Executing the Production Application Release Pipeline ........................................... 67  
  Review Enhancements to the Pipeline ......................................................................... 76  
  Execute the Enhanced Pipeline ................................................................................... 84  
  Conclusion....................................................................................................................... 94  
Module 3 - vRealize Code Stream for IT DevOps (45 minutes) ....................................... 96  
  Introduction to vRealize Code Stream for IT DevOps .................................................. 97  
  Introduction to Lab Scenario ......................................................................................... 99  
  Capture a vRealize Automation Blueprint from Rainpole Development Tenant .... 103  
  Migrate the Captured Blueprint to the Rainpole vRealize Automation Production Tenant ......................................................................................................................... 116  
  Modify JSON Data to Customize Execution ................................................................. 130  
  Conclusion....................................................................................................................... 146
Lab Guidance

Note: It will take more than 90 minutes to complete this lab. You should expect to only finish 2-3 of the modules during your time. The modules are independent of each other so you can start at the beginning of any module and proceed from there. You can use the Table of Contents to access any module of your choosing.

The Table of Contents can be accessed in the upper right-hand corner of the Lab Manual.

You will explore how to use vRealize Code Stream to orchestrate DevOps release processes and continuous application delivery. The same concepts will then be applied to the Software-Defined Data Center to move infrastructure components as code between environments.

Lab Module List:

- **Module 1 - Introduction to vRealize Code Stream** (45 minutes) (Basic)
  Explore how vRealize Code Stream leverages your existing software development tools. Review how to implement application release automation for custom developed applications and IT infrastructure.

- **Module 2 - Using vRealize Code Stream for Application Release Automation** (60 minutes) (Advanced)
  Create an application release pipeline with vRealize Code Stream, vRealize Automation, Jenkins, and Artifactory. As a release manager, you will automate the build, test, and deployment of a sample eCommerce application.

- **Module 3 - vRealize Code Stream for IT DevOps** (45 minutes) (Advanced)
  Walk through migrating vRealize Automation infrastructure constructs between tenants using the vRealize Code Stream Management Pack for IT DevOps.

Lab Captains:

- **Module 1 - Ed Bontempo, Senior Systems Engineer, USA**
- **Module 2 - Ed Bontempo, Senior Systems Engineer, USA**
- **Module 3 - Ed Bontempo, Senior Systems Engineer, USA**

This lab manual can be downloaded from the Hands-on Labs Document site found here:

http://docs.hol.vmware.com

This lab may be available in other languages. To set your language preference and have a localized manual deployed with your lab, you may utilize this document to help guide you through the process:

Introduction to the Scenario

Rainpole Systems is an electronics manufacturer located in Palo Alto, CA. Rainpole designs and manufactures electronic devices for use in everything from aircraft instrumentation to home automation. Given the diversity of their product set, Rainpole needs to develop cloud-based IT services to support the increasing demands of the business units.

Rainpole Systems is planning to use vRealize Automation and vRealize Code Stream to automate the build, testing, and deployment of their custom developed eCommerce application. They also plan to leverage the vRealize Code Stream Management Pack for IT DevOps to promote vRealize Automation application blueprints from development to production to minimize the chance of errors impacting their production cloud environment.

Rainpole Systems is currently leveraging a number of tools in their software development lifecycle including Jenkins, Gitlab, and Selenium. These systems have been loosely tied together through custom scripts, but they lack an overall view of where particular builds are in the application release process. Furthermore, they lack governance as code is promoted from development to test and ultimately to production. They are exploring leveraging vRealize Code Stream to streamline these processes and improve their overall software development lifecycle.

You will take on the roles of a Rainpole Cloud Administrator, a Rainpole Developer, and a Development Manager in this exciting lab exercise in order to experience how VMware and vRealize Code Stream can help make these goals a reality.
Location of the Main Console

1. The area in the RED box contains the Main Console. The Lab Manual is on the tab to the Right of the Main Console.
2. A particular lab may have additional consoles found on separate tabs in the upper left. You will be directed to open another specific console if needed.
3. Your lab starts with 90 minutes on the timer. The lab can not be saved. All your work must be done during the lab session. But you can click the **EXTEND** to increase your time. If you are at a VMware event, you can extend your lab time twice, for up to 30 minutes. Each click gives you an additional 15 minutes. Outside of VMware events, you can extend your lab time up to 9 hours and 30 minutes. Each click gives you an additional hour.

Alternate Methods of Keyboard Data Entry

During this module, you will input text into the Main Console. Besides directly typing it in, there are two very helpful methods of entering data which make it easier to enter complex data.
Click and Drag Lab Manual Content Into Console Active Window

You can also click and drag text and Command Line Interface (CLI) commands directly from the Lab Manual into the active window in the Main Console.

Accessing the Online International Keyboard

You can also use the Online International Keyboard found in the Main Console.

1. Click on the Keyboard Icon found on the Windows Quick Launch Task Bar.
Click once in active console window

In this example, you will use the Online Keyboard to enter the "@" sign used in email addresses. The "@" sign is Shift-2 on US keyboard layouts.

1. Click once in the active console window.
2. Click on the Shift key.

Click on the @ key

1. Click on the "@" key.

Notice the @ sign entered in the active console window.
**Activation Prompt or Watermark**

When you first start your lab, you may notice a watermark on the desktop indicating that Windows is not activated.

One of the major benefits of virtualization is that virtual machines can be moved and run on any platform. The Hands-on Labs utilizes this benefit and we are able to run the labs out of multiple datacenters. However, these datacenters may not have identical processors, which triggers a Microsoft activation check through the Internet.

Rest assured, VMware and the Hands-on Labs are in full compliance with Microsoft licensing requirements. The lab that you are using is a self-contained pod and does not have full access to the Internet, which is required for Windows to verify the activation. Without full access to the Internet, this automated process fails and you see this watermark.

This cosmetic issue has no effect on your lab.

**Look at the lower right portion of the screen**

![Image of lab status](image)
Please check to see that your lab is finished all the startup routines and is ready for you to start. If you see anything other than "Ready", please wait a few minutes. If after 5 minutes your lab has not changed to "Ready", please ask for assistance.
Module 1 - Introduction to vRealize Code Stream (30 minutes)
Introduction

This module will demonstrate the power of vRealize Code Stream to rapidly build application release pipelines. With application release pipelines, you can model and automate existing processes to accelerate the delivery of custom application code from the developer's laptop all the way through to production. You will begin with an overview of vRealize Code Stream, how it works, and what problems it solves. Finally, you will create a simple application release pipeline with a test stage.

This HOL-1821-04-CMP lab is broken up into 3 individual modules with varying lengths. Use the "Modules" list to determine which use case(s) you want to complete. You may choose to complete any or all modules, keeping in mind that you have 90 minutes available per session.

If you have not already reviewed it, the Lab Introduction contains details about Rainpole Systems (our example company) as well as important information on text entry, multiple language support, and on-screen keyboard configuration. If desired, you can click here to view this information again.

You will need about 30 minutes to complete this module.
Introducing vRealize Code Stream

vRealize Code Stream Introduction

What is vRealize Code Stream? Think of Code Stream as a code-release-specific orchestration engine. It allows users to define a visual representation/template of a release pipeline, tying in to all of the existing various SDLC mechanisms that a customer is using, while providing stage-gate checks to verify if a release should continue its path from one stage to the next.

In a release delivery process, after development checks in their enhancements or new features, several additional groups need to test, stage, and accept the release prior to putting it into production. Today this is largely accomplished using siloed processes with a collection of tools that may or may not be consistent across groups. Not only does the process to deploy and test at each stage take too long, but coordination and consistency at each stage also suffers. vRealize Code Stream allows application release teams to model any kind of release process for any kind of application - from simple, single-node, on-premises applications to complex, multi-tiered, cloud-based or hybrid next-gen applications. Pipeline models configure the workflow tasks and governance policies used to build, deploy, and test the software at each stage in the delivery process, as well as the gating rules between stages.

As an introduction, watch this brief video on vRealize Code Stream.
Overview and Benefits

VMware helps enterprise IT to overcome cloud management challenges with the following differentiators:

- **Achieving Fastest Time-to-Cloud Value**: Extensive out-of-the-box capabilities, extensible architecture, and ability to leverage existing investments enables you to create scalable private, public, and desktop cloud services in just days.
- **Rapid Ecosystem Integration**: A full spectrum of extensibility options that empowers IT personnel to enable, adapt, and extend their cloud to work within their existing IT infrastructure and processes, thereby eliminating expensive service engagements while reducing risk.
- **Embracing Consumerization of IT**: Unifies disparate systems and processes into a highly personalized, self-service experience governed by user and organizational awareness, ensuring that the exact needs of the user are met in the context of the business.
- **Boosting IT Efficiency**: VMware's advanced self-service and automation capabilities lead to a significant reduction in operational expenditures (OpEx,) while organizationally aware governance enables a multi-tenant infrastructure, improving hardware utilization and eliminating capital expenditures (CapEx.)
- **Increasing Business Agility**: Enterprise IT is empowered to quickly enable cloud services so that the business can react quickly to changing market demands and capture market share.
- **Hybrid Cloud Automation**: Agnostic provisioning of hardware platforms, operating systems, hypervisors, management tools, and public clouds; vRealize Automation allows customers to rapidly stand up cloud services to deliver quick time to value from your IT investments.
Architecture Concepts

vRealize Code Stream is integrated with vRealize Automation and can work with vCenter Server, Chef, Puppet, or scripts. vRealize Code Stream can also integrate with continuous integration frameworks and testing frameworks, and a variety of other development toolsets.

Architecture

vRealize Code Stream includes three primary modules: a pipeline execution engine, a release dashboard, and reporting. Code Stream also integrates with a number of DevOps-related components, including but not limited to:

- **Repository and Source Code Management** tools, including jFrog Artifactory, git, and Nexus
- **Continuous Integration (CI)** tools, including Jenkins, Bamboo, and Microsoft Team Foundation Server
- **Testing Frameworks** such as Selenium
- **Issue Tracking** tools, including Jira and Bugzilla
- **Change Management and ITSM** toolsets, including BMC Remedy and ServiceNow
- **Provisioning and Configuration Management** tools, including vRealize Automation, Pivotal Cloud Foundry, and Puppet
This integration is provided by vRealize Code Stream's open source Xenon engine, and is available out of the box, via REST API integration, or through vRealize Orchestrator plugins.

**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 to a category.

**Gating Rule:** A set of rules that must be completed before the software changes are promoted to the next stage, starting the next set of tasks. The gating rules include testing rules and compliance rules. Gating rules that are associated with a pipeline are specific to an organization and to applications.

**Instance:** A vRealize Orchestrator plugin 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 release into production. For example: development, test, user acceptance test (UAT,) load test, staging, and production.

**Provider:** Plugin 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.
Introduction to Lab Scenario

As you work through the lab's use cases, you'll be assuming roles within a fictional company known as Rainpole Systems, and addressing their business and IT challenges.

Rainpole is an IT company that is under pressure to build private clouds, which means creating a data center architecture that can deliver flexible, scalable computing while maintaining governance and control over their environment. Rainpole's IT Automation team decided to reach out to VMware to help them with their journey to private cloud.

vRealize Automation empowers IT to accelerate the delivery of personalized, business-relevant IT services while improving overall efficiency, by delivering infrastructure, applications, and custom services through a unified IT service catalog. In addition to the need to drastically reduce the time to provision services for their customers, Rainpole also had a need to have a more agile development process that will enable more frequent and reliable software releases. vRealize Code Stream fits in to this category by allowing for the modeling of any release process for any kind of software, while integrating with existing continuous integration tools and artifact management.

Over the next few months, Rainpole's IT Automation team will be starting their private cloud journey with vRealize Automation and vRealize Code Stream.

Rainpole's Fundamental IT Challenge

- Quickly implement a tool which will allow IT to maintain control over their data center resources while allowing engineering to quickly create and destroy environments for development and testing using shared, heterogeneous clouds.
- Provide a personalized self-service experience to the different teams within engineering development (as a starting point, within QE, or Quality Engineering, and potentially other parts of the organization to be added in the future.)
- Decrease service delivery times by automating the end-to-end delivery and management of resources to meet the needs of the agile development environment. Provide an approval mechanism when needed to ensure compliance is met.

In order to support self-service for the engineering and QE teams while leveraging existing infrastructure from both internal and external resources and retaining the necessary controls, the Rainpole IT leadership team has decided to implement VMware's next generation cloud management solution. The Rainpole development team will kick off the implementation as the first group of users, with QE scheduled to follow on shortly after in a second wave.

If you have not already reviewed it, the Lab Introduction contains useful details about Rainpole Systems (our example company,) as well as important information on text entry, multiple language support, and on-screen keyboard configuration. If desired, you can click here to review this information again.
In this module, you will act as the Rainpole Cloud Administrator to explore building and executing a simple application release pipeline within vRealize Code Stream.
Building a Simple Application Release Pipeline

In this exercise, you will build a simple Application Release Pipeline with vRealize Code Stream.

Open Chrome Browser from Windows Quick Launch Task Bar

First you will log in to vRealize Code Stream to begin building a new application release pipeline.

1. Click on the **Chrome icon** on the Windows Quick Launch Task Bar.
Log In to vRealize Code Stream

1. Log in as rpadmin with the password VMware1!
2. Click Sign In

Passwords for all users will be the same: VMware1!
Examine the MyCommerce Test Blueprint

Before we create a pipeline, we will first examine the vRealize Automation blueprint used in this exercise.

1. Click the **Design** tab, and the Blueprints section will load by default
2. To view the **MyCommerce Test** blueprint, click on the name (blue text) to open the Blueprint Design Canvas
View the blueprint on the Design Canvas and make note of its components. When you deploy a test system with vRealize Code Stream you will select this blueprint, which deploys a CentOS server, performs systems configuration (disabling the firewall,) installs Apache Web Server, and connects the machine to an existing network.

1. Note the Apache and System_Configuration software components, deployed to the vSphere Machine
2. Note the blueprint's connection to the DefaultExternalNetwork
3. Click Cancel to exit the design canvas and begin creating your new pipeline
Navigate to Code Stream and Add a New Pipeline

1. Click the Code Stream tab
2. Click the Pipelines sub-tab
3. Finally, click the Add button in the toolbar to begin creating a new pipeline

Build a New Pipeline with a Test Stage

1. Enter the Name: HOL - Simple Pipeline
2. Enter the Description: Simple pipeline created in Module 1
3. Click the Stages button to add stages to the pipeline.
The first thing you'll want to do once you're in Code Stream is to start building out your own pipeline. Pipelines can be modeled first without specifying any configurations. This can give you a good way to lay out the entire process before entering any details.

1. Enter **HOL - Simple Pipeline** for the name
2. Enter **Simple pipeline created in Module 1** for the description
3. Click **Stages >** to begin modeling the pipeline

### Add a Pipeline Stage

To add a stage in vRealize Code Stream:

1. Click on **Add Stage**, which will create a basic stage named **Stage0**
2. Double-click **Stage0** and enter **Test** to rename it
3. Click **Add Task** to begin creation of the first task in the pipeline
Add a Provisioning Task

Enter the following information for the task:

1. Select **vRealize Automation 7** for the Provider drop-down
2. For Name, enter **Provision Machine**
3. Click **OK**

Configure Provisioning Task

1. Select the gear icon (not shown) next to the Provision Machine task and then select **Configure** from the drop-down

You will be prompted to save the pipeline configuration (not shown) - click **Yes** when prompted.
Task vRA Properties

Before configuring the task, you should resize the window so it fits better in the screen. Click in the upper left corner and drag down and to the right to resize (not shown here.) You should be able to see the Save and Cancel buttons at the bottom of the Task Configuration window, as well as the scrollbar on the right.

1. In the task editor, click the drop-down and select vRealize Automation 7 for the Endpoint
2. Click the drop-down next to Blueprint and select MyCommerce Test. Note this is the blueprint you examined earlier in this exercise.
1. Scroll the window down by dragging the slider on the right side
2. On the left side of the window, click **MyCommerce**
3. Note the **properties** that are available for the virtual machine. The options for CPUs, Memory, etc are defined for the vRealize Automation blueprint and are available for selection (if allowed) here
4. Enter **HOL-1821 Code Stream** for the Description
5. Click **Save** to continue to be taken back to the Stages screen
Add a Deploy Website Task

1. Still using the Test stage, select the **Add Task** icon to create a new task

---

**Deploy Task**

1. Select **Script** for the Provider drop-down
2. For the Name, enter **Install Website**
3. Select **OK**
Configure Deploy Task

1. Select the gear icon next to the Install Website task and then select Configure from the drop-down

You will be prompted to save the pipeline configuration (not shown) - click Yes when prompted.

Configure Custom Script Task - General

1. In the General tab, enter the following information:
For Host, select **Read From Pipeline or Task Property**. This is because you want to leverage the VM that gets created as a result of the Provision task you just created.

- In the Machine Type Property Path (the field directly under the User Defined Hosts radio button,) type `${Test.Provision Machine.machines}` (Note, there is a **space** between the words **Provision** and **Machine**, and this field is case sensitive.)
- Select **All Hosts** from the drop-down next to the variable
- Username: **root**
- Password and Confirm Password: **VMware1!**
- Script Type: **Bash**
- Execute Script file: `/root/mycommerce/install.sh`

2. Select the **Advanced** tab

**Note:** You can paste the variables or begin typing `${` and make the selection from the list presented. The next variable will be presented once you click on the existing variable.

### Configure Custom Script Task - Advanced

![Configure Custom Script Task - Advanced](image)

Under **Script Parameters** - **Other**, enter the following parameters:

1. Select **Add**
2. Enter the following parameter names and values:
   - Parameter Name: **JENKINS_SERVER**
- Parameter Value: jenkins-01a.corp.local
- Parameter Name: JENKINS_JOB_NAME
- Parameter Value: MyCommerce-Build (Note: The parameter value is case sensitive)

3. Click **Save** to be taken back to the Stages screen

**Completed Test Stage**

1. Once both tasks are configured and saved, click **Activate** to activate the pipeline template
Executing a Simple Application Release Pipeline

In the previous exercise, you created a simple application pipeline and activated it. It is now time to execute the pipeline to validate that everything is working as expected.

Execute the Pipeline

1. Click to highlight the pipeline **HOL - Simple Pipeline**
2. Click **Execute**

Execute the Pipeline, Continued

1. Click **OK**
1. Click **OK**

**Enable Auto-Refresh**

1. Click the **Pipeline Executions** tab
2. Click **Enable Auto-refresh** to automatically update the pipeline status (it will likely take several minutes for the pipeline to complete)
3. Click the **small arrow** to the left of the pipeline execution to see the details

**Successful Pipeline Execution**
Once the pipeline Test stage and associated tasks turn green, it has executed successfully.

**Deployment Verification**

To verify the VM and code deployment worked, you will need the IP address of the VM that was created as part of the pipeline execution.

1. Within vRealize Automation, select the **Items** tab
2. Click the **Deployments** tab
3. Click the arrow next to the most recent deployment in the list that has a name starting with **MyCommerce Test** to expand the components
4. Document the **IP Address** in the IP Address column. (Note: it may be necessary to hover the mouse cursor over the IP address, or to expand the IP Address column, in order to see it in its entirety.)
Once you have the IP address, you can verify that your pipeline execution was a success.

1. Create a **new tab** in Chrome
2. Browse to the IP of the newly created VM in the new lab, which will show the MyCommerce.com homepage (Note: the IP address in your lab may differ from this screenshot)
Exercise Clean-Up

Once you have verified successful completion of the pipeline, you will now clean up the lab. Return to the vRealize Automation tab in Chrome (not shown) and:

1. Click to highlight the recent **MyCommerce Test** deployment (Note: do not click on the name itself, or you will be taken to the Item Details screen rather than selecting the deployment)
2. Click the arrow next to **Actions**
3. Click **Destroy**
ConfirmDestroy

1. Click **Submit** to destroy the virtual machine and **OK** (not shown) to confirm.

You have successfully concluded this exercise.
Conclusion

In this module, you have begun to explore the capabilities of vRealize Code Stream. You learned about Application Release Automation and modeled a simple pipeline to quickly deploy a test system using vRealize Automation. After modeling this pipeline, you executed it and verified the results. This was a very simple example and we encourage you to explore Module 2 in this lab to see more complex pipelines and use cases.

You've finished Module 1

Congratulations on completing Module 1.

If you are looking for additional information on vRealize Code Stream, try one of these:

- Go to https://www.vmware.com/products/vrealize-code-stream.html
- Or use your smart device to scan the QRC Code.

Proceed to any module below which interests you most.

- **Module 2 - Using vRealize Code Stream for Application Release Automation** (60 minutes) (Advanced) Create an application release pipeline with vRealize Code Stream, vRealize Automation, Jenkins, and Artifactory. As a release manager, you will automate the build, test, and deployment of a sample eCommerce application.

How to End Lab

To end your lab click on the **END** button.
Module 2 - Using vRealize Code Stream for Application Release Automation (60 minutes)
Introduction to Lab Scenario

As you work through the lab's use cases, you'll be assuming roles within a fictional company, Rainpole Systems, and addressing their business and IT challenges.

Rainpole is an IT company that is under pressure to build private clouds, which means creating a data center architecture that can deliver a flexible, scalable computing environment while maintaining governance and control over their environment. The Rainpole IT Automation team decided to reach out to VMware to help them with their journey to the private cloud. vRealize Automation empowers IT to accelerate the delivery of personalized, business-relevant IT services while improving overall efficiency by delivering infrastructure, applications, and custom services through a unified IT service catalog. In addition for a need to drastically cut off the time to provision services for their customers, Rainpole also had a need to have a more agile development process that will enable more frequent and reliable software releases. vRealize Code Stream fits into this category by allowing modeling of any release process for any kind of software, while integrating with existing continuous integration tools and artifact management.

Over the next few months, the Rainpole IT Automation team will be starting their private cloud journey with vRealize Automation and vRealize Code Stream.

Rainpole's Fundamental IT Challenge

• Quickly implement a tool which will allow IT to maintain control over their data center resources while allowing engineering to quickly create and destroy environments for development and testing using shared, heterogenous clouds.
• Provide a personalized self-service experience to the different teams within engineering development (as a starting point, with QE and potentially other parts of the organization to be added in the future.)
• Decrease service delivery times by automating the end-to-end delivery and management of resources to meet the needs of the agile development environment. Provide an approval mechanism when needed to ensure compliance is met.

In order to support self-service for the engineering and QE teams while leveraging existing infrastructure from both internal and external resources and retaining the necessary controls, the Rainpole IT leadership team has decided to implement VMware's next generation cloud management solution. The Rainpole development team will kick off the implementation as the first group of users, with QE scheduled to follow on shortly after in a second wave.

If you have not already reviewed it, the Lab Introduction contains details about Rainpole Systems (our example company,) as well as important information on text entry,
multiple language support, and on-screen keyboard configuration. If desired, you can click here to review this information again.

This module will build on the simple application release pipeline created in Module 1. If you did not take Module 1, a starter pipeline with the work from Module 1 is already created so there is no requirement to previously complete it.

In this module, you will act as the Rainpole Administrator and the Rainpole Cloud Administrator to explore vRealize Code Stream.
Power on Brownfield Server

Before we begin the exercises in this module, you will need to power on a server used by the subsequent pipelines. This server will act as a "brownfield" staging and production environment for the MyCommerce.com application.

Open Chrome Browser

1. Click the Chrome icon in the Windows Quick Launch Task Bar. If the browser window is already open following completion of the previous module, open a new tab in the browser instead (not shown.)

Navigate to the vSphere Client and Log In
1. In Chrome, click the **Region A** folder to expand the list of bookmarks
2. Click **HTML 5 Client** in the drop-down
3. Click the checkbox labeled **Use Windows session authentication** to populate the User name and Password fields
4. Click **Login**

**Locate the Brownfield VM**

1. Locate the VM brownfield-01a in the inventory (pictured above,) and click on it.
Power On Brownfield VM

1. Click the **Actions** link to expand the menu of available Virtual Machine actions
2. Move the cursor over the **Power** section to expand the sub-menu
3. Click **Power On**

Verify Power On and Exit

1. Verify that the brownfield-01a virtual machine icon has changed to include a green arrow, indicating it is powered on. Once verified, you can log out of the vCenter client.
2. Click **Administrator@CORP.LOCAL** to expand the menu
3. Click **Logout** and close the browser tab (not shown)
Building a "Staging" Stage for QA

In this exercise, you will expand a simple application release pipeline to include a "Staging" stage.

Navigate to vRealize Automation

First you will log in to vRealize Automation to begin building a new application release pipeline.

1. Click on the vRealize Automation on the Chrome bookmarks bar

Log in to vRealize Automation
1. Log in as `rpadmin` with the password `VMware1!`
2. Click **Sign in**

Passwords for all users will be the same: `VMware1!`

**Navigate to Code Stream and Edit a Pipeline**

1. Click the **Code Stream** tab
2. Click the **Pipelines** sub-tab
3. Click the pipeline named **Base - Standard Pipeline**
4. Finally, click the **Edit** button in the toolbar to begin editing the pipeline
1. Click on **Stages >** to navigate to the Stages view
Adding a "Staging" Stage

To add a "Staging" stage in vRealize Code Stream:

1. Click on Add Stage, which will create a basic stage named Stage0
2. Double-click Stage0 and enter Staging

Clone an Existing Task
Next, you're going to create an Install Website task for the "Staging" stage. To save time, you can clone the existing task from the Test stage.

1. Select the gear icon (not shown) next to the Install Website task and select **Clone** from the drop-down menu.
2. Click **OK** (not shown) on the Clone Task warning that passwords will be reset to null.

### Change the Name of the Install Website Task

1. Drag the cloned **Install Website-1** task to the Staging stage
2. Select the gear icon next to the Install Website-1 task and then click **Configure** in the drop-down
3. You will be prompted to save the pipeline configuration (not shown) - click **Yes** when prompted.

### Change the Name of the Install Website Task

1. Click in the Name field of the Task Configuration for Script and change the name to **Install Website**
1. Scroll down the Task Configuration window by dragging the slider on the right side.
2. In the general section, enter the following information:
   - For Host, select **User Defined Hosts**. This is because you will use an existing machine that has already been provisioned to demonstrate the ability to leverage "brownfield" hosts in a pipeline.
   - In the open text box below User Defined Hosts, type **brownfield-01a.corp.local** and press Enter.
   - Username: verify it is still set to **root**.
   - Password and Confirm Password: **VMware1!**.
   - Script type: verify it is still set to **Bash**.
   - Script File: verify it is still set to **/root/mycommerce/install.sh**.
3. Click the **Advanced** tab.
Configure Install Website Task - Advanced

1. Under Script Parameters - Other, verify the following parameters:
   - Name: JENKINS_SERVER
     Value: jenkins-01a.corp.local
   - Name: JENKINS_JOB_NAME
     Value: MyCommerce-Build

2. Click Save

Add a Delete Test Machine Task
Because you no longer need the test machine you provisioned in the test stage, you can create a task to delete the test machine. A vRealize Orchestrator workflow has already been created to call the vRealize Automation API to destroy the virtual machine. You will now add this task as another step in the Staging stage.

1. Click **Add Task** on the Staging stage

**Add a Delete Test Machine Task, Continued**

Enter the following information.

1. Select **vRealize Orchestrator** for the Provider drop-down (Note: you may need to scroll down in the drop-down menu to see vRealize Orchestrator)
2. For Name, enter **Delete Test Machine**
3. Select **OK**
Configure Delete Test Machine Task

1. Select the gear icon next to the Delete Test Machine task, and then select **Configure** in the drop-down

You will be prompted to save the pipeline configuration (not shown) - click **Yes** when prompted.
Configure Custom Script Task - General

1. For Workflow, select **Destroy vRA Machine** from the drop-down

2. In the Input Parameters section, for machineName enter `$\{\text{Test.Provision Test Machine.machines}\}` (Note, there is a space between the words Provision and Test and Machine, and that this field is case sensitive.)

3. Click **Save**

**Note:** You can paste the variables above or start typing `$\{` and make the selection from the list presented. The next variable will be presented once you click on the existing variable.

This completes the Staging stage for now. Remain on the **Edit Pipeline** screen for the next exercise.
Build a Gating Rule

Gating rules allow you to control when code is promoted between pipeline stages.

You can create gating rules based upon approvals or upon the results of automated tests performed with testing frameworks like Selenium. In this exercise, you will create a gating rule to promote code from the Test stage to the Staging stage based on the results of a Selenium test executed by Jenkins.

Return to the Pipeline Properties Screen

1. On the Edit Pipeline screen from the previous exercise, click < Previous
Add a Property Value to the Pipeline

1. Click + Add to add a new property
2. Enter the following for the newly created property:
   - Name: teststring
   - Description: Test String for Selenium Tests
   - Value: MyCommerce.com - Deal of the Day
   - Note: All entries in Step 2 are case sensitive

3. Click Stages > to continue
Create a Selenium Test Task in the Test Stage

1. In the Test stage, click + Add Task

Adding Selenium Test Task

Enter the following information:

1. Select Jenkins for the Provider drop-down
2. For Name, enter Selenium Test
3. Click OK
Configure Selenium Test Task

Select the gear icon next to the Selenium Test task, and then select **Configure** in the drop-down. You will be prompted to save the pipeline configuration (not shown) - click **Yes** when prompted.

(Note: it may be necessary to scroll down in the Test stage to see the Selenium Test task)
Selenium Test Jenkins Properties

1. Scroll down the Task Configuration window by dragging the slider on the right side
2. For Endpoint, select Jenkins from the drop-down
3. For Job, select MyCommerce-Test from the drop-down
4. In the Parameters section, for testurl enter $\{\text{Test.Provision Test Machine.machines[0].value.hostIp[0]}\}$
5. In the Parameters section, for teststring enter $\{\text{pipeline.teststring}\}$
6. Click Save

Note: You can paste the variables or start typing $\{\}$ and make the selection from the list presented. The next variable will be presented once you click on the existing variable, though the last two in the string will need to be manually entered. The next step will explain more about the syntax used above.

Note About JSON Syntax

When specifying the testurl parameter in the previous step, you may have noticed how the parameter completion didn't work once you got to the "[0]" part. This is because we
need to pass the IP of the host to our Selenium job, which is verifying that the website was actually deployed and is running.

These kinds of expressions are possible in vRealize Code Stream, since the values are passed in JSON format. For variable binding in Code Stream, you can autocomplete up to any variable in the pipeline, but the expression after (in our case, after ".machines"), is a pure JSON path expression.

The string we are parsing contains a "hostIp" parameter, which we need to pass to the Selenium test. We know that it contains this because we first captured the output of `${Test.Provision Machine.machines}`.

For more information about this syntax, refer to https://github.com/jayway/jsonpath outside of this environment.

**Add a Gating Rule**

1. Between the Test and Staging stages, click on the Gating Rule icon
Configure the Gating Rule

1. Select the radio button next to **If outcome of a vRealize Orchestrator workflow is successful** (Note: it may be necessary to drag the Gating Rule Configuration window up following this step, in order to view the rest of the window)
2. For Select Workflow, select **Test Acceptance Threshold** from the drop-down
3. In the Input Parameters section, for Threshold Percentage enter **100**
4. In the Input Parameters section, for Test Result enter **${Test.Selenium Test.testResult}** (Note: there is a space between the words Selenium and Test, and this field is case sensitive.)
5. Click **Save**

**Note:** You can paste the variables or start typing ${ and make the selection from the list presented. The next variable will be presented once you click on the existing variable.

This completes the exercise. Next you will build a production stage.
Build an Approval Policy and Move to Production

Next, you will create a new stage called Production with an approval policy that requires the Rainpole Cloud Admin to approve code moves to production.

To save on lab resources, you will re-deploy to the brownfield server and pretent that it's a server in a production environment.

Adding a Production Stage

To add a Production stage in vRealize Code Stream:

1. Click on + Add Stage to create a basic stage named Stage0
2. Double click Stage0 and enter Production
3. Select the gear icon (not shown) next to the Install Website task in the Staging stage, and then select Clone in the drop-down

Click OK (not shown) on the Clone Task warning that passwords will be reset to null.
Configure the Install Website Task

1. Drag the cloned **Install Website-1** task to the Production stage
2. Select the gear icon (not shown) next to the Install Website-1 task and then select **Configure** in the drop-down

You will be prompted to save the pipeline configuration (not shown) - click **Yes** when prompted.

Change the Name of the Install Website Task

1. Click on the Name field of Task Configuration and enter **Install Website**
Configure Install Website Task - General

1. Scroll down the Task Configuration window by dragging the slider on the right side.
2. In the General section, enter the following information:
   - For Host, verify that User Defined Hosts is selected.
   - In the open textbox, verify that brownfield-01a.corp.local is entered.
   - Username: verify that it is still set to root.
   - Password and Confirm Password: VMware1!
   - Script type: verify that it is still set to Bash.
   - Script File: verify that it is still set to /root/mycommerce/install.sh.
3. Click the Advanced tab.

Note: The Production and Staging stages utilize the same brownfield server. This would not normally be done in a pipeline, but was done here due to limited lab resources.
Configure Install Website Task - Advanced

1. Under Script Parameters - Other, verify the following parameters:
   - Name: **JENKINS_SERVER**
     - Value: jenkins-01a.corp.local
   - Name: **JENKINS_JOB_NAME**
     - Value: MyCommerce-Build (Note: this parameter value is case sensitive)

2. Click **Save**
Add a Gating Rule

1. Between the Staging and Production stages, click on the Gating Rules icon.

Configure the Gating Rule

1. Select a condition to proceed to next stage: **If outcome of a vRealize Orchestrator workflow is successful**
2. Select Workflow: **Approval**
3. Input Parameters:
   - **Approval Group (DN)**: **vsphere.local/approvalgroup**
   - **Approval Message**: The execution of $(releasePipelineName)
4. Save
1. Select the radio button next to **If outcome of a vRealize Orchestrator workflow is successful** (Note: it may be necessary to click on the Gating Rule Configuration window title and drag it up in order to view the entire window)
2. For Select Workflow, select **Approval** from the drop-down
3. In the Input Parameters section, for Approval Group (DN) enter **vsphere.local\approvalgroup**
4. Click **Save**

The base lab already has a custom group created in vRealize Automation named approvalgroup with cloudadmin@corp.local as its only member.

### Activate Pipeline

1. Click **Activate** to activate the pipeline. The pipeline is now ready to be executed.
Executing the Production Application Release Pipeline

In the previous exercise, you created a production application release pipeline and activated it. It is now time to execute the pipeline to validate that everything is working as expected.

Execute the Pipeline

1. Click to highlight the pipeline Base - Standard Pipeline
2. Click Execute

Execute the Pipeline, Continued
1. Click **OK**

**Enable Auto-Refresh**

Click the Pipeline Executions tab (not shown.) From there:

1. Click **Enable Auto-refresh** to automatically update the status (Note: Auto-refresh may already be enabled if you've completed the previous module)
2. Click the **small arrow** on the left of the pipeline execution to see the details
Pipeline Pause for Approval

1. The pipeline will successfully complete up to the **gating rule** between Staging and Production. This pause is for the approval policy that you defined in the gating rule.

**NOTE:** The pipeline is paused when a yellow 'pause' symbol appears next to the **IN_PROGRESS** status. You will also notice a vertical yellow bar next to the gating rule. Continue to the next step when the pipeline pauses. (This process may take 10 or 15 minutes.)

Log Out of vRealize Automation

Now let's log out of vRealize Code Stream in order to approve the request to move code into production.

1. Click **Logout**
Return to Login Page

1. Click Go back to login page

Log in to vRealize Automation as Cloud Admin

1. Log in with username "cloudadmin" and password "password".
2. Click on "Sign in".
1. Log in as **cloudadmin** with the password **VMware1!**
2. Click **Sign in**

**Navigate to the Inbox to Approve the Request**

1. Click the **Inbox** tab
2. Click on the **Manual User Action** tab
3. Click on the request number (example: number 3) to open the approval tab
Approve the Request

Notice the details of the request in the Approver Message field. When done, proceed to approving the request:

1. Enter **This code is ready for production** in the Comments field
2. Click **Submit** to approve

Log Out of vRealize Automation

Now let's log out of vRealize Automation in order to return to the pipeline execution already in progress.

1. Click **Logout**
Return to Login Page

1. Click **Go back to login page**

Log In to vRealize Automation as Rainpole Admin

1. Log in as **rpadmin** with the password **VMware1!**
2. Click **Sign in**

**Successful Pipeline Execution**

You will now return to the Pipeline Execution sub-tab within Code Stream, and verify that the pipeline completed successfully.

1. Click on the **Code Stream** tab
2. Click the **Pipeline Executions** sub-tab
3. Expand the pipeline
4. Validate that the Status of the pipeline has changed to **Execution finished successfully.**
Website Verification

You can now verify that your pipeline execution was a success.

1. Create a **new tab** in Chrome
2. Browse to **http://brownfield-01a.corp.local** in the new tab, which will show the MyCommerce.com homepage

You have completed this exercise. Next, you will review an enhanced version of this pipeline.
Review Enhancements to the Pipeline

In this exercise, you will review a pre-created pipeline with a number of enhancements to the workflow, including parallel execution, conditional execution, and artifact management. This enhanced pipeline will build on the work you completed in previous exercises.

Navigate to Code Stream and Edit the Enhanced Pipeline

Note: If you still have the MyCommerce website tab open, click on the vRealize Automation tab in Chrome first.

1. Click the Code Stream tab
2. Click the Pipelines sub-tab
3. Click the pipeline named Complete - Enhanced Pipeline
4. Finally, click the Edit button in the toolbar to begin editing the pipeline
Pipeline Properties

1. Three new properties have been added to the pipeline which will be used by tasks later in this lesson:
   - **EMERGENCY_RELEASE** is set to a default value of `false`. This property will be used later to signify an emergency bugfix release of the software, and will bypass the Install Website task in the Staging stage.
   - **BUILD_NAME** is set to a default value of MyCommerce-Build. This property will be used in the artifact management tasks to resolve the correct binary artifacts.
   - **BUILD_NUMBER** does not have a default value. This property will also be used in artifact management tasks to resolve the correct binary artifacts.

2. Click the **Stages >** button to continue
Parallel Tasks in Test Stage

vRealize Code Stream has the capability to execute tasks in parallel in order to accelerate the speed of execution.

1. In the Test stage, notice the **Provision Machine** and **Resolve Artifact** tasks are side by side. This signifies that they will execute in parallel. To make a task run in parallel, simply drag the task next to the task you would like it to run in parallel with.

View Configuration of the Resolve Artifact Task

1. Select the gear icon (not shown) next to the **Resolve Artifact** task, and then click **Configure** in the drop-down
This task will resolve an artifact from a repository in Artifactory by utilizing the pipeline properties defined previously.

1. Scroll down the Task Configuration window by dragging the slider on the right side
2. Note the Input Properties that are used to resolve the correct artifact. Two of these properties, artifactID and version, are based on properties defined at the pipeline level.
3. Click **Cancel** to return to the Pipeline view
View Configuration of the Install Website Task

1. Click the gear icon (not shown) next to the Install Website task in the Test stage, and then click **Configure** in the drop-down

**View Install Website Properties**

1. Click on the **Advanced** tab
2. Select **Remote Script File**
3. Enter the script path: `root/mycommerce/artfactory-install.sh`
The Install Website task has been updated to take advantage of Artifactory for binary retrieval:

1. Scroll down the Task Configuration window by dragging the slider on the right side
2. The Script File value has been updated to `/root/mycommerce/artifactory-install.sh` in order to take advantage of Artifactory integration
3. Click Advanced at the top of the window to view the details on the properties sent to this script

**View Install Website Advanced Properties**

![Task Configuration for Script](image)

The advanced properties are used to specify the location and build information for the artifacts being retrieved.

1. Scroll down the Task Configuration window by dragging the slider on the right side
2. In Script Parameters Other, the TargzFile parameter is set based on the output from the Resolve Artifact task covered previously.
3. Click Cancel to return to the Pipeline view
View Install Website Task in the Staging Stage

1. Select the gear icon (not shown) next to the Install Website task in the Staging stage, and then click Configure in the drop-down

View Install Website Properties

The Install Website task in the Staging stage is identical to the one in the Test stage, with one exception: this task has conditional logic applied which allows it to be bypassed in the event the EMERGENCY_RELEASE property is set to true. This functionality can be useful to speed up the deployment process for bug or security fixes.
1. For Execute Task, verify that the radio button for **On Conditions** is selected
2. In the textbox below the radio button, you will see the algorithm to determine whether or not the task will be executed. In this case, the task will be executed as long as the value of `EMERGENCY_RELEASE` is not equal to `True, True, or true`.
3. Click **Cancel** to return to the Pipeline view

This concludes the exercise. You will now execute this pipeline to see it in action.
Execute the Enhanced Pipeline

In the previous exercise, you reviewed an enhanced application release pipeline. This pipeline is used to test and implement changes to the MyCommerce.com application.

Rainpole's developers have identified a change needed for the MyCommerce.com application, and they have a new build of the application ready to implement. However, this new build must be tested and approved before it can be implemented in production, and it must be done in a way that will satisfy all requirements of Rainpole's SDLC process.

In this exercise, you will execute the enhanced vRealize Code Stream pipeline to implement this specific change. Since it is critical that this change is implemented quickly, you will also leverage conditional task execution to skip a portion of the pipeline that is not required in the event of emergency changes.

**Note:** Execution of this pipeline may take 10-15 minutes to complete.

Execute the Pipeline

1. Click to highlight the pipeline **Complete - Enhanced Pipeline**
2. Click **Execute**
Executing the Pipeline, Continued

1. Change the value for the EMERGENCY_RELEASE property to **TRUE** to flag this specific pipeline execution as an emergency release
2. Enter **6** for the BUILD_NUMBER property, to specify the exact build to implement and test
3. Click **OK** to start the application release pipeline
Enable Auto-Refresh

Following the previous step, click on the Pipeline Executions sub-tab (not shown.) From there:

1. Click **Enable Auto-refresh** to automatically update the status. (It may take 10-15 minutes for this pipeline execution to complete.)
2. Click the **small arrow** to the left of the pipeline execution to see the details
3. As the pipeline executes, note that **Provision Machine** and **Resolve Artifact** execute concurrently in the Test stage
Pipeline Pause for Approval

1. The pipeline will successfully complete through the gating rule between Staging and Production. This pause is for the approval policy you defined on the gating rule. The status of the pipeline will remain at **PAUSED** and will have a **yellow pause sign** next to it.

2. Note that the Install Website task in the Staging stage was skipped (gray bar at the top of the task, and **Skipped** for the timeframe) because we set the **EMERGENCY_RELEASE** value to **TRUE** when executing the pipeline.

Log Out of vRealize Automation

Now let's log out of vRealize Automation in order to approve the request to move the code into production.

1. Click **Logout**
Return to Login Page

1. Click **Go back to login page**

Log In to vRealize Automation as Cloud Admin

1. Log in as **cloudadmin** with the password **VMware1!**
2. Click **Sign in**

**Navigate to Inbox to Approve the Request**

1. Click on the **Inbox** tab
2. Click on the **Manual User Action** tab
3. Click on the **request number** (example: the number 1) to open the approval item
Approve the Request

1. Notice the details of the request in the Approver Message field
2. Enter **This code is ready for production** in the Comments field
3. Click **Submit** to approve

Log Out of vRealize Automation

Now let's log out of vRealize Automation to check on the status of the pipeline.

1. Click **Logout**
Return to Login Page

Click **Go back to login page**

Log In to vRealize Automation as Rainpole Administrator

1. Log in as **rpadmin** with the password **VMware1!**
2. Click **Sign in**

### Successful Pipeline Execution

You will now return to the pipeline execution tab and validate that the pipeline has completed successfully.

1. Click on the **Code Stream** tab
2. Click the **Pipeline Executions** sub-tab
3. Validate that the Detailed Status of the pipeline has changed to **Execution finished successfully**
Successful Pipeline Execution, Continued

Now let's verify the results of the pipeline execution:

1. Create a **new tab** in Chrome
2. Browse to `http://brownfield-01a.corp.local` in the new tab, which will show the MyCommerce.com homepage (Note: it may be necessary to hold **Shift** and press the **Reset** button in the browser in order to see the change to the application)

Congratulations! You have completed this exercise, and the emergency change to MyCommerce.com has been successfully tested and released into production via the vRealize Code Stream enhanced pipeline.

You may now close the browser window.
Conclusion

In this module, you have explored building complex application release pipeline with vRealize Code Stream. You learned about approvals, artifact management tasks, automated testing, conditional execution, and viewing pipeline status. vRealize Code Stream is truly the glue that binds together your Software Development Lifecycle (SDLC) processes to increase agility and ensure that processes are repeatable.

You've finished Module 2

Congratulations on completing Module 2.

If you are looking for additional information on integrating software development lifecycle tools with vRealize Code Stream, try one of these:

- View the [vRealize Code Stream 2.3 Information Center](https://vcs.vmware.com/)
- Or use your smart device to scan the QRC Code.

Proceed to any module below which interests you most.

- **Module 1 - Introduction to vRealize Code Stream** (45 minutes) (Basic)
  Explore how vRealize Code Stream leverages your existing software development tools. Review how to implement application release automation for custom developed applications and IT infrastructure.
- **Module 3 - vRealize Code Stream for IT DevOps** (45 minutes) (Advanced)
  Walk through migrating vRealize Automation infrastructure constructs between tenants using the vRealize Code Stream Management Pack for IT DevOps.
How to End Lab

To end your lab click on the **END** button.
Module 3 - vRealize Code Stream for IT DevOps (45 minutes)
Introduction to vRealize Code Stream for IT DevOps

VMware's concept of the Software-Defined Data Center (SDDC) abstracts every service in the data center from its underlying platform and defines services as software. Software-defined content can be treated just like code in any software application, meaning that modern software development methodologies can be adopted, and that practices such as agile development and DevOps can be applied to the configurations and content within the SDDC. Although this approach lends itself to modern development practices and rapid service delivery, the reality is that it can still be challenging to manage an entire data center's worth of software-defined services.

The vRealize Code Stream Management Pack for IT DevOps eases the burden of managing software-defined content by providing administrators with the following capabilities:

- Manage Software-Defined Data Center (SDDC) content as code, automating the release process.
- Apply version control to all SDDC content.
- The ability to provide Continuous Integration/Continuous Delivery (CI/CD) through automated testing and release.
- Capture software-defined content automatically in a consistent format from multiple environments.
- Store content automatically on a centralized storage location.
- Partition the storage to allow multiple developers to work on the same content separately using branches.
- Tag content to allow concurrent release to multiple environments.
- Roll back to the last good releaseable version across any content type.
- Leverage vRealize Automation XaaS forms, API access, and access control through entitlements of catalog items.
- Use vRealize Code Stream to extend pre/post extensibility using custom pipelines.
- (Optional) Require approval before deploying content to a test or release environment.
- Audit and version control logging.

Watch the following brief video on the vRealize Code Stream Management Pack for IT DevOps for more information.
vRealize Code Stream Management Pack for IT DevOps

Video
Introduction to Lab Scenario

In this module, you will walk through migrating vRealize Automation infrastructure constructs between Rainpole's Development and Production tenants using the vRealize Code Stream Management Pack for IT DevOps.

vRealize Code Stream Management Pack for IT DevOps

The Rainpole Cloud Administrators have decided to set up a Development tenant within vRealize Automation in order to develop their blueprints and software components. In order to manage the release of their developed blueprints and components, they have installed the vRealize Code Stream Management Pack for IT DevOps, allowing them to manage infrastructure as code utilizing a DevOps lifecycle.

If you have not already reviewed it, the Lab Introduction contains details about Rainpole Systems (our example company) as well as important information on text entry, multiple language support, and on-screen keyboard configuration. If desired, you can click here to review this information again.

In this module, you will act as both the Rainpole Cloud Administrator and as a Development User in order to explore the vRealize Code Stream Management Pack for IT DevOps.
Prepare Environment

In order to prepare the lab environment for these exercises, it is necessary to log in to vRealize Orchestrator and run a specific workflow. To log in to vRealize Orchestrator, click on the vRealize Orchestrator Client shortcut on the desktop.

Log In to vRealize Orchestrator

1. Log in to vRealize Orchestrator as administrator@vsphere.local (this is the default value) with password VMware1!
2. Click Login
1. Select the **Workflows** tab
2. Navigate to **Library -> Content Management -> Configuration -> Helpers** (not shown)
3. Scroll down to locate the **Reset Queue Schedule Policy** workflow, and click on it to select it
4. Click the **Start workflow...** icon to run the workflow
Log Out of vRealize Orchestrator

1. Verify successful completion of the Reset Queue Schedule Policy workflow. The most recent run will show a green checkmark indicating success.
2. Log out of vRealize Orchestrator by clicking the X to close the window, and click Exit in the confirmation window (not shown)
Capture a vRealize Automation Blueprint from Rainpole Development Tenant

Here you will prepare a blueprint for deployment into the Rainpole Production tenant by first capturing it from the Development tenant, assigning a version to it, and storing it within the repository.

Open Chrome Browser from Windows Quick Launch Task Bar

First you will log in to the Rainpole Development tenant to view the new blueprint to be captured.

1. Click on the Chrome icon on the Windows Quick Launch Task Bar. If the browser window is already open following completion of the previous module, open a new tab in the browser instead (not shown.)

Browse to the Rainpole vRealize Automation Development Tenant

1. Click on the Rainpole Dev Tenant bookmark in the Chrome bookmarks bar.
Log in to the Rainpole Development Tenant

1. Ensure that the domain **corp.local** is selected from the drop-down menu
2. Click **Next**
Log in to the Rainpole Development Tenant - Continued

1. Log in as **rpadmin** with the password **VMware1!**
2. Click **Sign In**
View the Blueprint in the Catalog

1. Click Catalog
2. View the Hadoop service, but do not click on it. We will be capturing this blueprint in this exercise.

Examine the Blueprint that you will Capture

1. Click on the Design tab, and the Blueprints section will be displayed by default
2. To view the Hadoop blueprint, click on the name (blue text) to open the blueprint design canvas
View the Blueprint Design Canvas

View the blueprint on the **Design Canvas** and make note of its components. You will capture this blueprint as code and store it in the repository, ready for deployment into production.

1. Note the **Hadoop-Master** software component, deployed to the vSphere Machine object
2. Note the blueprint's connection to the **ExternalDefaultNetwork**
3. Click **Cancel**

Log Out of the Rainpole Dev Tenant

1. Click **Logout**
Browse to the Rainpole vRealize Automation Production Tenant

You will now log in to the Rainpole production tenant, and request the service to capture the Hadoop blueprint from the Rainpole Development tenant.

1. Click on the vRealize Automation bookmark on the Chrome bookmarks bar

Log In to the vRealize Automation Production Tenant

1. Ensure that the domain `corp.local` is selected from the dropdown menu
2. Click `Next`
Log In to the Rainpole vRealize Automation Production Tenant

You will log in as the Rainpole Administrator, an administrator who has been granted rights to the vRealize Code Stream for IT DevOps Management Pack.

1. Log in as `rpadmin` with the password `VMware1!`
2. Click `Sign in`
Navigate to the Catalog and run the Single Package Request

You will now submit a Single Package Request as you will only be capturing a single package in this instance. Note that Group Package Requests can also be submitted from this screen which can be used to capture, test, and release a group of content packages, such as multiple blueprints.

1. Click on the **Catalog** tab
2. Select the **Packages** service category
3. Find the **Single Package Request** service, and click **Request**
1. Once you've read the description of this request, click **Next** to continue
Fill In the Request Details (Part 2)

1. Select the checkbox for **Capture content from endpoint**
2. Click **Next**
Fill In the Request Details (Part 3)

You will now specify the type of package to capture. In this exercise you will be capturing a vRealize Automation Composite Blueprint, the source endpoint to capture the package from (the Rainpole Development tenant,) and finally the content you will be capturing (the Hadoop blueprint you viewed earlier.)

1. From the Select package type drop-down menu, select **Automation-CompositeBlueprint**
2. From the Select source endpoint drop-down menu, select **Rainpole vRA Development Tenant**
3. From the Select content package drop-down menu, select the blueprint **Hadoop - [id=Hadoop]**
4. Ensure the **Include dependencies:** checkbox is selected
5. Click **Next**
Submit the Request

1. Enter **Production ready** in the Enter a version comment textbox
2. Check the **Mark new version as release (optional)** checkbox
3. Click **Submit**

Complete the Request

Click **OK**
Monitor the Pipeline Execution

Now to verify the pipeline execution:

1. Click the **Code Stream** tab
2. Click the **Pipeline Executions** sub-tab
3. Monitor the **Status**
4. Click **Refresh** until the Status entry reads "**Execution finished successfully.**"

Congratulations! You have successfully captured a package containing a vRealize Automation composite blueprint from the Rainpole Development tenant. Continue to the next section to deploy the captured package into the Rainpole Production Tenant.
Migrate the Captured Blueprint to the Rainpole vRealize Automation Production Tenant

You will now take your previously captured blueprint and deploy it to the Rainpole vRealize Automation Production tenant.

View the Existing Blueprints

First, let’s check that the Hadoop blueprint we captured from the Rainpole Development tenant previously does not exist in the Rainpole Production tenant:

1. Click the Design tab
2. Click Blueprints to ensure that you are viewing the list of available blueprints
3. Ensure that the Hadoop blueprint does not exist in the list of blueprints
Let's also verify that the Hadoop blueprint software component dependencies do not exist in the Rainpole Production tenant:

1. Click **Software Components**
2. Verify that the **Hadoop** software component does not exist in the list of available components
Run the Single Package Request

Now you will release the Hadoop blueprint, which you captured and stored in the repository earlier, into the Rainpole Production tenant:

1. Click the **Catalog** tab
2. Select the **Packages** service
3. Find the **Single Package Request** service, and click **Request**
Fill In the Request Details (Part 1)

1. Click **Next**
1. Click the checkbox for **Release content to production**
2. Click **Next**
Fill In the Request Details (Part 3)

Now you will identify the package that will be released into the Production tenant:

1. From the Select package type drop-down menu, select **Automation-CompositeBlueprint**
2. From the Select content package drop-down menu, select **Hadoop - [id=Hadoop,includeDependencies=true]**
3. From the Version (optional) drop-down menu, select 1
4. Click **Next**
Fill Out the Release Details

Now you will specify where to release the content package:

1. Select **Rainpole vRA Production Tenant** in the Select release endpoints in deployment order textbox, and click the arrow to move it to the textbox on the right (final results shown above)
2. In the Enter a release comment textbox, type **Version 1.0 blueprint**
3. Click **Next**
Submit the Request

1. Click Submit

Complete the Request

• Click OK
Monitor the Pipeline Execution

Now let's check the pipeline execution:

1. Click the **Code Stream** tab
2. Click the **Pipeline Executions** sub-tab
3. Monitor the status of the pipeline
4. Click **Refresh** until the Status column reads **Execution finished successfully**.
View the Migrated Blueprint

Now let's verify that the Hadoop blueprint you captured earlier has been successfully released into the Rainpole Production tenant:

1. Click the **Design** tab
2. Click **Blueprints** to ensure that you are viewing the blueprints
3. Click on the new **Hadoop** blueprint that you just migrated to view the Design Canvas
View the Blueprint Design Canvas

1. You can see that the blueprint and its components were all migrated into the Rainpole Production tenant
2. Click **Cancel**
View Software Components

View the software components to ensure that the Hadoop blueprint dependency (the Hadoop-Master software component) was also captured and released into the Rainpole Production tenant:

1. Click **Software Components**
2. See that the **Hadoop-Master** software component was migrated as it was a dependency for the Hadoop blueprint

Log Out of vRealize Automation

Now let's check if the Service category and the Hadoop Service were created in the catalog. You will need to log out and log in as a Rainpole Development user to view the service.

1. Click **Logout**
1. Click **Go back to login page**

**Log In to the Rainpole Production Tenant as Devuser**

1. Log in as **devuser** with the password **VMware1!**
2. Click **Sign in**

**Navigate to the Catalog and View the Hadoop Blueprint**

1. Click the **Catalog** tab
2. Click on the **Big Data** service category
3. The **Hadoop** blueprint is part of the Big Data service catalog, as all dependencies were captured (including the blueprint icon)

Congratulations! You have successfully migrated a composite blueprint and all of its dependent components from the Rainpole vRA Development tenant to the Rainpole Production tenant.
Modify JSON Data to Customize Execution

In this exercise, you will request a single package capture with the mock request flag checked. You will then utilize CloudClient to export the request to a JSON file format, modify it by setting the mock request flag to false, and using the Houdini plugin for CloudClient to submit the request capturing a new version of the Hadoop blueprint.

Log Out of vRealize Automation

First, you will need to log out and log back in as the Rainpole Administrator.

1. Click **Logout**

Return to Login Page

1. Click **Go back to login page**
Log In to the Rainpole vRealize Production Tenant

1. Log in as **rpadmin** with the password **VMware1!**
2. Click **Sign in**
Navigate to the Catalog

You will now run through a new Single Package Request. However, this time you will mark it as a mock request so that the request is not actually deployed. You will export this request in a JSON file format later.

1. Click the **Catalog** tab
2. Select the **Packages** service category
3. Find the **Single Package Request** service and click **Request**
Fill In the Request Details (Part 1)

1. Click **Next**
1. Select the checkbox for **Capture content from endpoint**
2. Click **Next**
1. From the Select package type drop-down menu, select **Automation-CompositeBlueprint**
2. From the Select source endpoint drop-down menu, select **Rainpole vRA Development Tenant**
3. From the Select content package drop-down menu, select **Hadoop - [id=Hadoop]**
4. Ensure the **Include Dependencies:** checkbox is selected
5. Click **Next**
Submit the Request

1. Enter **Generating a request from the vRA UI** in the Enter a version comment textbox
2. Click the **Mark new version as release (optional)** checkbox
3. Click the **Email request data** checkbox
4. Enter **rpadmin@rainpole.com** for the email address
5. Enter the email address **rpadmin@rainpole.com** in the Email addresses (comma separated): field
6. Click the **Mock request** checkbox, as we do not want to actually run the request
7. Click **Submit**

Complete the Request

- Click **OK**
1. Click **OK**

Next, you will identify this request using CloudClient, and export it in a JSON format so that you can edit it.

**Open CloudClient**

Double-click the **cloudclient** icon on the Desktop (minimize the Chrome browser window if necessary, so that you can view the desktop.)

**Edit Window Properties**

First, we'll increase the size of the CloudClient window so that you can read the CLI more easily:

1. Click the **Icon** in the top left of the window
2. Select **Properties** from the menu
Increase Window Width

1. Set Width in the Window Size section to 125
2. Click OK
Log In to the CloudClient Interface

1. Enter the following command to log in to CloudClient:

```
vr \login userpass --server vra-01a.corp.local --user rpadmin@corp.local --password VMware1! --tenant vsphere.local
```

Identify the Last Request ID

In order to export our most recent request into a JSON file format, we must first identify the request ID:

In order to export our most recent request into a JSON file format, we must first identify the request ID:
1. Run the command: `vra request list`
2. Right click in the CloudClient window and select **Mark**
3. Highlight the **ID** for the highest numbered request in the list and right click again to copy it (Note: this may or may not be the last request in the list)

### Export the Request to a JSON Formatted File

Now, you will run the command to export the request into a JSON file format:

```
vra request detail --id <paste the request id copied earlier here> --export c:\request.json --format JSON
```

Remember that you can paste the request ID by right-clicking in the command window and selecting **Paste**

### Open File Explorer

1. From the Windows Quick Launch Bar, open **File Explorer**
Open the request.json File Using Notepad++

Now you will begin to modify the previous request by editing the JSON file you just created.

1. Navigate to **Local Disk (C:)**
2. Right-click on the **request.json** file and select **Edit with Notepad++**
Search for the Line to Edit

You will modify the previous request by editing the JSON file to remove the Mock Request key which was set during request.

1. From the menu bar, click on the image of the **Binoculars** to open the Find window
2. In the Find what field, type **provider-mock**
3. Click **Find Next**
4. Click **Close**
Edit the JSON File

1. In the `provider-mock` section, change the value from true to `false`
2. Click **Save** to save the modified file
3. **Close** the Notepad++ window
4. Return to the CloudClient window

Display the Content Packages

Now let's check the current version of the Hadoop blueprint stored within the repository.

1. Run the command: **houdini content list**

The list shows the captured Hadoop blueprint and dependencies package that was captured in the first lesson of this module.

2. Note the current version and latest release version are set to **1**.
Run the Modified JSON Request

1. Run the following command: `houdini request --file c:\request.json`

This command will submit the modified JSON request.

Verify the Status of the Request

1. Run the command: `vra request list`
You will see that the request is now **IN_PROGRESS**. Run the command again until the state entry changes to **SUCCESSFUL**. This may take some time depending on your lab environment, so please be patient.

### Check the Content Package Version

1. Run the command: **houdini content list**
2. The list shows that the captured Hadoop Blueprint and dependencies package that you captured earlier has now been recaptured and its version has been incremented. This was achieved by setting the mock request flag to false by editing the JSON request.

You have completed this exercise. This is only a small example of leveraging the "mock request" functionality within the vRealize Code Stream Management Pack for IT DevOps to change package configuration at request time by modifying data in JSON format.
Conclusion

In this module, you have begun to explore the capabilities of the vRealize Code Stream Management Pack for IT DevOps. From the Rainpole Development Tenant, you captured a vRealize Automation blueprint and its dependencies, adding it to the repository with version control. You then took that blueprint and its dependencies, and deployed it to the Rainpole Production Tenant. Following this deployment, you confirmed that the blueprint and dependencies were added to the catalog and that the required software components came along for the ride.

You then explored the use of the plugin for CloudClient by exporting a request, modifying the JSON file, and using it to submit a new request.

You've finished Module 3

Congratulations on completing Module 2.

I hope you have enjoyed this lab. If you want to learn more about the vRealize Code Stream Management Pack for IT DevOps, please try:

- Or use your smart device to scan the QRC Code

Proceed to any module below which interests you most.

- **Module 1 - Introduction to vRealize Code Stream** (45 minutes) (Basic) Explore how vRealize Code Stream leverages your existing software development tools. Review how to implement application release automation for custom developed applications and IT infrastructure.
- **Module 2 - Using vRealize Code Stream for Application Release Automation** (60 minutes) (Advanced) Create an application release pipeline with vRealize Code Stream, vRealize Automation, Jenkins, and Artifactory. As a release manager, you will automate the build, test, and deployment of a sample eCommerce application.
How to End Lab

To end your lab click on the **END** button.
Conclusion

Thank you for participating in the VMware Hands-on Labs. Be sure to visit http://hol.vmware.com/ to continue your lab experience online.

Lab SKU: HOL-1821-04-CMP

Version: 20171005-174543