# Table of Contents

Lab Overview - HOL-1921-03-CMP - vRealize Automation 7 - Advanced Extensibility.............................................. 3
- Lab Guidance ....................................................................................................................................................... 4

Module 1 - Introduction to Extensibility (30 minutes) ......................................................................................... 11
- Introduction ...................................................................................................................................................... 12
- Introduction to Extensibility ........................................................................................................................... 13
- Introducing the Event Broker ......................................................................................................................... 19
- Exploring Event Subscriptions ...................................................................................................................... 26
- Find and Import Blueprints from the VMware Solutions Exchange ......................................................... 47
- Conclusion ...................................................................................................................................................... 53

Module 2 - Infoblox IPAM Integration with vRealize Automation (45 Minutes) ............................................. 55
- Introduction ...................................................................................................................................................... 56
- Reviewing the Infoblox Configuration .......................................................................................................... 57
- Reservation Preparation ................................................................................................................................. 60
- Creating a Blueprint to use Infoblox ............................................................................................................. 63
- Configuring Infoblox Integration with an NSX On-Demand Load Balancer ........................................... 69
- Infoblox Integration with an NSX On-Demand Routed Network ............................................................... 82
- Infoblox Integration with an NSX On-Demand NAT Network ................................................................... 98
- Conclusion ...................................................................................................................................................... 110

Module 3 - Integrating vRealize Automation with ServiceNow (45 Minutes) .................................................. 113
- Introduction ...................................................................................................................................................... 114
- Install and Configure MID Server ............................................................................................................... 115
- Configure ADFS and ServiceNow SSO ........................................................................................................ 116
- Configure vRealize Automation for ADFS .................................................................................................. 117
- Install and Configure vRealize Automation Plugin for ServiceNow ......................................................... 119
- Conclusion ...................................................................................................................................................... 120

Module 4 - Puppet Integration with the Configuration Management Framework (45 Minutes) ................... 122
- Introduction ...................................................................................................................................................... 123
- Puppet Installation and Prerequisites .......................................................................................................... 124
- Creating the Configuration Management Framework Endpoint ............................................................. 126
- Create a Puppet Managed Blueprint .......................................................................................................... 133
- Deploy and Test the New Blueprint ............................................................................................................. 142
- Conclusion ...................................................................................................................................................... 156

Module 5 - Creating and Managing Custom Resources with vRealize Automation (45 Minutes) ................. 158
- Introduction ...................................................................................................................................................... 159
- Create a Custom Resource From a vRealize Orchestrator Workflow ....................................................... 160
- Create an XaaS Blueprint from a Custom Resource Workflow ................................................................ 164
- Create Resource Actions to Manage the Custom Resource ................................................................... 174
- Entitle the XaaS Blueprint and Actions ...................................................................................................... 189
- Deploy and Manage the New Custom Resource ......................................................................................... 200
- Conclusion ...................................................................................................................................................... 206
Module 6 - Modifying Blueprint Request Forms with Custom Forms Designer (60 Minutes)..........................................................................................................................208
Introduction.........................................................................................................................209
Enabling Custom Forms on a Blueprint ...........................................................................210
Implementing Blueprint Constraints ...............................................................................222
Working with External Sources .........................................................................................227
Implementing External Validations ....................................................................................230
Extending Custom Form Design with Cascading Style Sheets (CSS)...............................237
Restoring Our Configuration Changes ..............................................................................246
Conclusion.........................................................................................................................249
Lab Overview - HOL-1921-03-CMP - vRealize Automation 7 - Advanced Extensibility
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.

Lab Module List:

- **Module 1 - Introduction to Extensibility** (30 minutes) Explore the limitless flexibility that extensibility brings to vRealize Automation, and learn about different ways you can implement and leverage it to streamline and automate complex processes.

- **Module 2 - Infoblox IPAM Integration with vRealize Automation** (45 minutes) Discover IPAM integration between VMware NSX, Infoblox NIOS and vRealize Automation. Learn how the On-Demand network in NSX consumes Infoblox IPAM for workloads deployed through vRealize Automation.

- **Module 3 - Integrating vRealize Automation with ServiceNow** (45 minutes) Examine the new ServiceNow integration capabilities in vRealize Automation 7. Learn how to provision and perform day 2 operations on resources in the ServiceNow portal.

- **Module 4 - Puppet Integration with the Configuration Management Framework** (45 minutes) Learn how vRealize Automation 7.4’s Configuration Management Framework integrates with Puppet to install and configure software on a newly provisioned machine.

- **Module 5 - Creating and Managing Custom Resources with vRealize Automation** (45 minutes) Learn how to use vRealize Orchestrator Custom Resources to extend vRealize Automation’s ability to manage existing objects throughout your business. From user accounts to coffee makers, Custom Resources are a powerful way to bring automation to all aspects of your business.

- **Module 6 - Modifying Blueprint Request Forms with Custom Forms Designer** (45 minutes) Learn how to extend and customize existing virtual machine blueprints to leverage the new Custom Forms Designer introduced in vRealize Automation 7.4. Make your request forms more dynamic and responsive with ClarityUI from VMware. Covering topics such as retrieving external values with vRealize Orchestrator, Customizing Look and Feel with CSS, and external validation.

**Lab Captains:**

- **Cody De Arkland, Senior Technical Marketing Architect, USA**

This lab manual can be downloaded from the Hands-on Labs Document site found here:
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 to begin their journey towards providing cloud-based IT services. The IT department has been tasked with finding a way to cut the time needed to provision individual virtual machines for development and quality assurance teams, but also to help the network teams speed up the provisioning of network infrastructure using NSX. In addition, the developers are interested in exploring their options for provisioning applications, both through their existing investment in external configuration management systems and through native vRealize Automation capabilities.

The line of business owners inside of Rainpole Systems are also keenly interested in maintaining their visibility into the lifecycle of the infrastructure that is provisioned through their IT Service Management tools.

As the Rainpole IT Department adopts this new model of cloud computing, they will also need a way to troubleshoot and diagnose any problems that may arise in the new infrastructure.

You will take on the roles of a Rainpole Cloud Administrator, a Rainpole Developer and a Development Manager in this exciting lab exercise to experience how VMware and vRealize Automation 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
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 you lab has not changed to "Ready", please ask for assistance.
Module 1 - Introduction to Extensibility (30 minutes)
Introduction

Learn how to deploy a WordPress application using Puppet and vRealize Automation's Configuration Management Framework. This HOL-1921-03-CMP lab is broken up into 5 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 you have 90 minutes available per session.

If you have not already reviewed it, the 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 to review this information again.

In this module you will act as the cloud admin and a developer for Rainpole to configure and deploy a WordPress application.

You will need about 30 minutes to complete this module.
Introduction to Extensibility

Automation isn't a new technology - it's a new way of thinking about existing processes and combining them with existing technologies. The goal of automation is to allow these existing investments to be leveraged while enforcing a policy-based model of lifecycle and governance to ensure they can be consumed in a repeatable fashion. VMware's vRealize Automation does just that - provide a framework for IT organizations to connect into their existing investments in infrastructure, asset management, business process management and others while maintaining a governance model.

What is Extensibility?

Extensibility is the ability to extend the functionality of a system without affecting the structure or code of the system.

In vRealize Automation, this means to extend the functionality of the product to integrate with existing processes and infrastructure. For example, anyone can take a VM template and deploy consistent VM's from that template. What is missing from this process is the ability to integrate with existing systems, such as CMDB, ITSM, and IP Address Management (IPAM) applications.

How is this Different from Customization?

As we discussed earlier, extensibility extends the functionality of a system within a provided framework, and without affecting the structure or code of the system. Customizations are generally built outside of a standard framework, and may cause issues with supportability and upgradability.

vRealize Automation is unique in that some of the features that would require customization in other products can be achieved through configuration changes or event subscriptions, leveraging vRealize Orchestrator and its many available plug-ins for both VMware and third party tools.

Extensibility Options in vRealize Automation 7

vRealize Automation 7 has the following options for extending functionality beyond simple virtual machine deployment:

- Event Broker
- XaaS blueprints and actions
Event Broker

One of the most exciting features in vRealize Automation 7 is the ability to use the Event Broker to monitor the vRealize Automation Message Queue for events, each of which can drive extensibility actions. These drivers, or Subscriptions, can be monitored for one or many matches to hundreds of different characteristics, allowing for unparalleled flexibility when driving custom actions. These Subscriptions can also be specified as Blocking Events, which would allow an external system to take action before the vRealize Automation request or event can continue.

We will explore the Event Broker in greater detail in the modules that follow in this lab.
Anything as a Service (XaaS)

XaaS, formerly known as Advanced Service Designer, brings new features and capabilities to vRealize Automation 7.

With XaaS, anything that can be automated through a vRealize Orchestrator workflow can be presented and consumed as service, opening a world of possibilities to extend vRealize Automation - the sky's the limit! The most powerful new feature is the ability to leverage XaaS in the Converged Blueprint Designer. Now you can simply drag and drop XaaS blueprints from the library directly onto the Design Canvas to create complex blueprints in minutes.

Custom Properties

Properties are used to control certain aspects of the provisioning process. Some properties are determined by standard settings that you must specify for all machines. For example, memory and disk size values are required for all blueprints. You can specify additional properties individually or in property groups within blueprints and in reservations. Use custom properties to add values or override existing or default values for the following information:

- Machine operating system
• Virtualization platform
• Build settings, such as disk size
• Integration with external systems

Custom properties can be marked as required properties. When a property is specified as required, the user or system must supply a value for that property when the machine is requested. One example of this may be to mandate that an accounting code be entered prior to requesting a machine.

Custom properties can be defined individually or in property groups that can be applied to various objects in the system. Properly authorized users can specify custom properties for blueprints, endpoints, business groups, and reservations. Within blueprints, properties can be added to the blueprint itself or to the individual components within a blueprint. Blueprint-level custom properties take precedence over custom properties that are configured at the component level.

The picture above shows a few properties that might be defined for a machine deployment. This is for demonstration purposes only. You will define some properties later in this module.

1. Values for the property can be defined at creation.
2. Values for the property can be defined at the time of request by the user.

Custom Property Precedence

When the same property exists in more than one source, vRealize Automation follows a specific order of precedence when applying properties to the machine.

You can add custom properties that apply to provisioned machines to the following elements:

• **Reservation**, to apply the custom properties to all machines provisioned from that reservation.
• **Business Group**, to apply the custom properties to all machines provisioned by business group members.
• **Blueprint**, to apply the custom properties to all machines provisioned from the blueprint.
• **Property Groups**, which can be included in a blueprint, to apply all the custom properties in the group to all machines provisioned from the blueprint. **NOTE:** A blueprint can contain one or more property groups.

• **Machine Request** to apply the custom properties to the machine being provisioned.

• **Approval Policy**, if advanced approval support is enabled, to require approvers to provide values for the machine being approved.

The following list shows the order of precedence for custom properties. A Property value specified in a source that appears later in the list override values for the same property specified in sources that appear earlier in the list.

- Property Group
- Blueprint
- Business Group
- Compute resource
- Reservations
- Endpoint
- Runtime

As you can see, a runtime property takes precedence over other properties. A runtime property meets the following conditions:

- The property is marked as Prompt User, which specifies that the user must supply a value for it when requesting a machine. This requires that the requestor customize individual characteristics of each machine, or gives them the option of doing so when a default value is provided for the property.

### Custom Property Types

The following vRealize Automation custom property types are available:

- **Internal** - The specified value is maintained in the database only. For example, the email address of the manager who approved a machine request is recorded in the VirtualMachine.Admin.Approver property but the property has no effect on the machine.

- **Read-only** - The specified value is implemented on the machine and cannot be changed. For example, VirtualMachine.Admin.UUID specifies the UUID of the machine, which cannot be changed.

- **External** - A machine’s external properties are determined when the virtualization platform creates the machine or during the WinPE phase of the build process. To set these properties, their values must be provided to the proxy agent, which passes them on to the virtualization platform, or to the guest agent, which implements them in the WinPE phase.

- **Not Updated** - The specified value is implemented on the machine but is never updated. For example, if the property VirtualMachine.Admin.AddOwnerToAdmins is set to true, the owner of the machine is added to its local administrators group.
If the owner is later removed from this group, the property is not updated to false.

- **Updated** - The specified value is implemented on the machine and is updated through data collection. For example, if the compute resource of a machine is changed, a proxy agent updates the value of the machine’s `VirtualMachine.Admin.Hostname` property.

**Learn More**

To learn more about custom properties and to explore the built-in custom properties, please scan the QR code above or go to the following link.

Custom Properties Reference for vRealize Automation 7.3

Introducing the Event Broker

Lifecycle extensibility has always been a critical feature of vRealize Automation. In past versions of the product this has only been available for IaaS lifecycle states, but not for Advanced Services Designer provisioning (now known as XaaS) or the Approval process.

Enter the vRealize Automation 7 Event Broker!

In this module, we will review:

• What Event Broker is all about
• Typical use cases which can be addressed using the Event Broker
• Walk through examples of a couple of the most common use cases, including integration with external systems for ITSM ticketing and CMDB functions

What is the Event Broker?

Event Broker is a centrally managed, simplified subscription based model for extending the complete provisioning lifecycle for Converged Blueprints in vRealize Automation 7, AND several other key services such as Approvals. Through the vRealize Automation Administration UI, you can now configure criteria-based subscriptions for an extended list of Event Topics and conditions, and assign a vRealize Orchestrator workflow to be called.

This is no longer done blueprint by blueprint, but instead can be based on broader conditions configured as part of the subscription process (such as by event lifecycle state name and phase, machine type, blueprint name, custom property, and many other possibilities). It also means that a single event could trigger multiple vRealize Orchestrator workflows without having to create a "wrapper" workflow!
Past versions of vRealize Automation allowed administrators to use the IaaS SDK and/or vRealize Orchestrator workflow stubs to customize the machine provisioning lifecycle. These were critical for common use cases including custom host naming, IPAM integrations, creating Active Directory (AD) accounts, adding machine objects to custom AD OUs, joining machines to domains, updating CMDB with new or updated machine details, and reversing all of this when a machine was decommissioned. While these types of extensions were critical in most vRealize Automation implementations, there were some use cases that could not be addressed in vRealize Automation 6.x, such as extensibility for managed objects created though the Advanced Service Designer (known as XaaS in vRealize Automation 7) and for customizing the Approval process.

Within this module, we will be exploring two key use cases:

- Approvals - calling out to external system for approvals, rather than using the vRealize Automation built in Approvals
- CMDB - creating and updating an ITSM CMDB during the provisioning and destroy of a virtual machine
Event Topics

With the new Event Broker, the integration points have expanded beyond the original six IaaS workflow states to Event Topics which can be broadly applied based on a variety of Conditions, rather than assigned to each blueprint. There are now nine broad Event Topics, shown above, which describe the type of event message sent to a specific vRealize Automation Service.

Event topics, described below, allow events to be triggered by other vRealize Automation services and sent to the Event Broker. Machine Lifecycle and Machine Provisioning are certainly the two you will use most frequently!

1. **Blueprint component completed** - A blueprint component that is part of a composite blueprint finishes provisioning. The component is any blueprint that is part of a composite blueprint.
2. **Blueprint component requested** - A blueprint component that is part of a composite blueprint is requested. The component is any blueprint that is part of a composite blueprint.
3. **Blueprint configuration** - A blueprint is created, updated, or deleted.
4. **Business group configuration** - A business group is created, updated, or deleted.
5. **Catalog item request completed** - A composite blueprint finished provisioning. This event topic includes all blueprint components. It does not include standalone XaaS blueprints.
6. **Catalog item requested** - A composite blueprint is requested. This event topic does not include XaaS blueprints.
7. **Component action completed** - An action run on a deployed blueprint component when a deployment action was requested.

8. **Component action requested** - An action to run on a deployed blueprint component is requested when a deployment action was requested.

9. **Deployment action completed** - An action on a deployed blueprint finished running, including running all the component actions.

10. **Deployment action requested** - An action on a deployed blueprint is requested.

11. **EventLog default event** - A standard entry is added to the event log.

12. **IPAM IP lifecycle event completion** - An IP allocation or deallocation request is finished.

13. **Machine lifecycle** - A provided IaaS action is run on a provisioned machine.

14. **Machine provisioning** - An IaaS machine is in the process of being provisioned.

15. **Orchestration server configuration** - A vRealize Orchestrator server configuration is created, updated, deleted, or modified to use a different default instance.

16. **Post Approval** - A post-approval policy level is configured to use the event subscription option.

17. **Pre Approval** - A pre-approval policy level is configured to use the event subscription option.

18. **Resource reclamation completion event** - A resource lease expired and the resources are reclaimed.

As you can see, these are very broad topics! We will review how to make our event triggers more granular using conditions as we dig a little deeper in just a bit.

**NOTE**: All subscriptions are Tenant specific!

### Blocking and Non-Blocking Events

<table>
<thead>
<tr>
<th></th>
<th>Blocking</th>
<th>Non-Blocking</th>
</tr>
</thead>
<tbody>
<tr>
<td>Workflow Fails</td>
<td>May Fail the Request (depends on Phase and Topic)</td>
<td>Request succeeds</td>
</tr>
<tr>
<td>Workflow Returns Values</td>
<td>Return values processed</td>
<td>Return values ignored</td>
</tr>
<tr>
<td>Concurrency</td>
<td>Workflows run in sequence, per request. Workflows run in parallel for multiple requests.</td>
<td>Workflows run in parallel</td>
</tr>
<tr>
<td>Potential use</td>
<td>Ability to manipulate provisioning parameters based on external input; also to chain execution of workflows in specific order.</td>
<td>Events fire but do not change provisioning parameters</td>
</tr>
</tbody>
</table>

There are two special types of Event Topics you must be aware of before we start trying to set up subscriptions in vRealize Automation - Blocking and Replyable Event Topics - and how they compare to standard, non-blocking event topics.
• **Non-Blocking Event Topics** - allow only non-blocking subscriptions; triggered asynchronously and you cannot rely on the order that the subscriptions are triggered or the order in which the vRealize Orchestrator workflows run.

• **Blocking Event Topics** - supported by some, but not all, events; subscriptions marked as blocking will prevent other subscriptions which also meet the event criteria from being triggered until this first workflow is finished. If you have multiple blocking workflow subscriptions for the same event topic (and criteria), you can prioritize the subscriptions to set the execution order. They are executed in priority order with highest priority being 0 (zero). After all blocking subscriptions are processed, the message is sent to all non-blocking subscriptions at the same time. Because the blocking workflow subscriptions run synchronously, the changed payload includes updated attributes when the subsequent workflows are notified (called).

• **Replyable Topics** - supported by some, but not all, events; replyable topics are inherently Blocking Events, but have special characteristics. The service that registered the replyable event topic can accept a reply event that provides the workflow output, usually as a result of an interaction with the system or user. The reply output parameters must meet the criteria defined in the reply schema so that the vRealize Automation service that published the original replyable event can process it. For example, pre-approval and post-approval workflow subscriptions are replyable; if you create a workflow that sends an approval request to an external system, the reply, approved or rejected, is processed by vRealize Automation to complete the request or mark it as rejected.

### Pre and Post Event State Phase

<table>
<thead>
<tr>
<th>PRE/POST</th>
<th>Blocking</th>
<th>Non-Blocking</th>
<th>Blocking</th>
<th>Non-Blocking</th>
</tr>
</thead>
<tbody>
<tr>
<td>Workflow fails</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Machine Provisioned, BuildingMachine and RegisterMachine can fail Request; All others Succeed</td>
<td>Request Succeeds</td>
<td>Request Succeeds</td>
<td>Request Succeeds</td>
<td></td>
</tr>
<tr>
<td>Workflow returns values</td>
<td>Return values processed</td>
<td>Return values ignored</td>
<td>Return values processed</td>
<td>Return values ignored</td>
</tr>
</tbody>
</table>

The Event State Phase you use when setting up your Subscription conditions will impact how the workflow return values are processed.

• **Pre State Phase** - For Blocking Events (including replyable events), it is possible for the workflow output (return) values to impact the Request status as well as to alter the values; Non-Blocking Events do not have this capability.

• **Post State Phase** - For Non-Blocking Events, subscriptions calling workflows during the Post State will not impact the overall Request status, and return values will be ignored. For Blocking Events, generally failing workflow will not fail the Request. This can, however, be configured through custom properties on the
blueprint to change the Request state. Return values will be processed for Blocking Events.

Error/timeout behavior is configurable for each state by the following custom properties on the machine:

- Extensibility.Lifecycle.Error.Event.{Workflow}.{State} - the value of the property is the name of the event to be triggered on the workflow in case of error/timeout
- Extensibility.Lifecycle.Error.State.{Workflow}.{State} - the value of the property is the name of the state in which the workflow will forcibly transition to in case of error/timeout

### Using Workflow Output Parameters

<table>
<thead>
<tr>
<th>Topic</th>
<th>Workflow Output Parameter</th>
<th>Used for</th>
</tr>
</thead>
<tbody>
<tr>
<td>MachineProvisioned</td>
<td>virtualMachineAddOrUpdateProperties</td>
<td>Manipulate VM properties for configuring Network cards, host names, IP address, etc.</td>
</tr>
<tr>
<td>BuildingMachine</td>
<td>virtualMachineDeleteProperties</td>
<td>Remove properties of the virtual machine</td>
</tr>
<tr>
<td>RegisterMachine</td>
<td>workflowNextState</td>
<td>Force the virtual machine into an other workflow state than the normal following state (use with caution, may have undesired affects)</td>
</tr>
<tr>
<td></td>
<td>virtualMachineEvent</td>
<td>Fires an event on the machine and is the recommended way to interact with the workflow</td>
</tr>
<tr>
<td>Pre-Approval</td>
<td>approved</td>
<td>Used for Approval Event Topics only, to return approval status from external systems</td>
</tr>
<tr>
<td>Post-Approval</td>
<td>FieldNames/fieldValues</td>
<td>Used Approval Event Topics, to return a set of properties (Name-Value pairs) – similar to Request properties which can be over-ridden by an Approver in native Approvals</td>
</tr>
</tbody>
</table>

Workflow output parameters can be used, as the previous table indicates, to allow Blocking subscriptions to alter the Request. Above are some examples of ways that a Request can be modified. The use cases we will review in this module include:

- Creating Configuration Item in an external CMDB - uses non-blocking event as no modifications to request or created virtual machine are needed
- Using external approval system.
Event State Timeouts

There is a global default configuration for extensibility workflows to timeout after 30 minutes. This global default can be adjusted as needed, under Infrastructure > Administration > Global Settings.

This can be overridden per Subscription, as you will see when we start setting them up!
Exploring Event Subscriptions

We will explore the topic of Event Subscriptions in vRealize Automation 7 in a bit more detail. Let's start by getting logged in as the cloud administrator. Other users will not have access to the Events menu in the UI.

It's worth noting that the following pre-requisites are already set up for you:

- vRealize Automation and the embedded vRealize Orchestrator are already installed

Open Chrome Browser from Windows Quick Launch Task Bar

1. Click on the Chrome Icon on the Windows Quick Launch Task Bar.

Log in to vRealize Automation

1. Enter cloudadmin for the username
2. Enter **VMware1!** for the password
3. Click **Sign In**

## Navigate to Event Subscriptions

![Diagram showing navigation steps]

We will start by locating the Subscriptions configuration:

1. Click on the **Administration** tab. Choose **Events** from the left hand navigation (not pictured).
2. Select **Subscriptions** to show table of existing subscriptions
3. Click the **New** button

### Zoom Out Browser

![Diagram showing browser zoom options]
Due to limitations of the lab environment's screen resolution we will use the zoom feature of the browser to zoom out. Note that in some cases this may cause rendering issues where text fields are unable to be seen. You may have to adjust back to 100% at times to be able to see these fields.

1. Select the configuration menu in Chrome
2. Click on the - twice to zoom out to 80%

**Select Event Topic**

![Select Event Topic Diagram]

The first step of setting up a new subscription is to choose the Event Topic you will use. As was mentioned in the introduction section, Machine Provisioning and Machine Lifecycle are the two you will likely use the most.

1. Click on the different Event Topics in the list, for this example choose **Machine Provisioning**
2. You will see details about that topic displayed on the right side
3. Note that the Schema view is particularly helpful
4. Click **Next** button
Selecting Conditions

Because Event Subscriptions are tenant-wide, planning ahead and choosing proper conditions for each subscription is critical! By default, the Run for all Events radio button is selected. If you leave this, you will be firing your workflow MANY times! This is because the event will trigger for each combination of Lifecycle State and Phase (Pre and Post). The number of Lifecycle States will vary between different Event Topics, but all will have a Pre and Post Phase.

1. For demonstration purposes, we will leave Run for all events selected
2. Click Next (not pictured)

Select a Workflow
Now we need to choose a vRealize Orchestrator workflow to execute for this Event Subscription.

1. Open the Orchestrator tree until you have **HOL-1921** folder open
2. Select the **Orchestrator > HOL-1921 > EB Get Custom Properties Example** workflow
3. On the right side, you can review the details of the workflow, including the Input and Output parameters
4. Click **Next**

### Save Subscription

1. Leave the default **Name** for your new subscription
2. Enter **Sample EB Subscription** in the description box
3. Here you can see the **Blocking** checkbox, which we will leave unchecked for this example; the type of Event topic will determine if this checkbox is available or not
4. Click the **Finish** button to save (not pictured)

### Publish Subscription

1. The new Subscription will be in a Draft state by default and must be Published to become active.
2. Select the row for the new Subscription (but don’t click on the Name or you will re-open it in edit mode)
3. Click the **Publish** button

### Launch vRealize Orchestrator

We need to have the vRealize Orchestrator Client open before we actually do our testing, otherwise we may miss watching it.

1. Double-click on the **vRealize Orchestrator Client** icon on the Windows Desktop.

### Log in to vRealize Orchestrator

1. Ensure the user name is set to **administrator@vsphere.local** and enter **VMware1!** for the password
2. Click **Login** (This could take 30 seconds)
Navigate to the Workflow List

1. Choose the Workflow tab
2. Expand the folders until you can see the workflows under administrator@vra-01a.corp.local > HOL > 1921-03-CMP folder

We will come back to vRealize Orchestrator to check on the workflow execution as we do our testing.

Open New Incognito Window

1. Right-click on Chrome in the Task bar

To reduce the login / logout cycles as we test our new Event Subscriptions, we will open a New incognito window in Chrome to allow us to log in as a second user.

1. Right-click on Chrome in the Task bar
2. Select **New incognito window** to open new window

**Launch vRealize Automation**

1. Choose the **vRealize Automation** shortcut to launch
2. Ensure **corp.local** domain is selected
3. Click **Next**
Log in as DevUser

1. Enter **devuser** for the username
2. Enter **VMware1!** for the password
3. Click **Sign In**

Request item from Catalog

To test out our Event Subscription, we will request a simple, single Virtual Machine.

1. Click on the **Catalog** tab
2. Locate the **CentOS** Catalog Item and click **Request**

## Submit Request

![Image of Submit Request interface]

1. Enter a Request **Description. NOTE:** If you have reduced the zoom on Google Chrome to 80%, you may not be able to see the dialogue box outlines to select due to rendering issues in Google Chrome. You can either reset your view to 100% or click in the area to the right of Description to type.

2. Click **Submit**

3. Click the **OK** button on the submission confirmation dialog (not pictured)

Leave this browser instance open as we will be returning in future steps to do more testing!

### Switch to the Orchestrator Client

1. Switch to the Orchestrator Client in the task bar
It's time to review what is happening in vRealize Orchestrator as our VM Request processes. Return to the vRealize Orchestrator client which you logged into a few steps back by choosing it from the Task bar. You should already be in the workflow list.

1. Expand the **EB Get Custom Properties Example**. Note that it may take some time for the list of executions to show up here.
2. You should see a long list of executions for this workflow - this is because you selected "Run for all Conditions" when we set up the Subscription. You can see what happens when you forget to set up conditions. Click on the first one in the list.
3. In the lower-right pane, click on the **Logs** tab to see the output from the workflow. All this workflow does is write all of the parameters for the event instance to the log for the workflow execution - nothing fancy yet, but can be very helpful as you start learning about the Event Broker!
4. You can drag the divider up so you can see the complete log output
5. When you review the log output in detail, there is some info that we want to pay particular attention to as we may want to use them later:
   - **BlueprintName** - name of the blueprint that was requested (will be the same for all of the workflow executions).
- **State** - there will be 2 or 3 log workflow executions per State (for each Phase); this is one we will want to use as one of our subscription conditions usually!

- **Phase** - will be PRE or POST, or in some cases EVENT; another one that will be frequently used as a condition when we set up our subscriptions.

- **Name** - vRA generated name of the VM being provisioned (your VM name may be different to the one shown)

- **Properties** - in this test it will be null, but we can actually send specific custom property values to our workflow by adding a property to the Blueprint. We will do this next!

It's worth taking a couple of minutes to click through the different workflow executions and note the different States and Phases! Understanding the different States and Phases and the sequence in which they run will help you immensely!

### States & Phases for Machine Provisioning Event Topic

This is a compiled complete list of all of the State and Phase values for the Machine Provisioning Event Topic, in sequential order (the same order as what you would see when reviewing each of the workflow instance logs in previous step). You can click on the image to enlarge it in your manual view.
Adding Properties to the Event Payload

Custom Properties and Property Groups are used in a variety of ways and for many types of integrations within vRealize Automation Blueprints. These are often absolutely necessary for extensibility. With the Event Payload, we can pick and choose specific Properties to pass, or we can send some or all, on a Blueprint by Blueprint basis.

1. Make sure you choose the first browser you had open, where you are logged in as **CloudAdmin**
2. Choose the **Design** tab
3. Edit the **CentOS** Blueprint by clicking on the name to open it in the Design Canvas. Note that your list of Blueprints may be different if you have completed other lab modules already
Open the Custom Properties tab for the Machine

We need to add a new custom property to the CentOS virtual machine:

1. Choose the CentOS VM on the canvas
2. Select the Properties tab
3. Select the Custom Properties tab
4. Click the +New button to add a new property

New Custom Property

We want to focus on the event that is triggered immediately after the VM build is complete.
1. Enter the property **Name** as:
   Extensibility.Lifecycle.Properties.VMPSMasterWorkflow32.Requested

Remember that you can copy this long string from above and paste it into the property name using the **Send Text** button above the desktop.

What you provide as a **Value** to this property will affect which actual properties are passed to the Events. You could, for example, use "*hostname"", which would include all properties that contain "hostname", or you could use "__*" to include all hidden vRA properties. For this test, we will want to pass everything.

2. Enter property **Value** as: *
3. Click **OK**
4. Click **Save** and then **Finish** at bottom of the canvas to save and close the blueprint (not pictured)

**Add Conditions to the Event Subscription**

To save time, for our next test, we are going to narrow down the vRealize Orchestrator workflow calls only to a specific Event, State and Phase by editing the Subscription we created initially.

1. Click on the **Administration** tab. If necessary, Choose **Events** option on the left hand navigation menu (not pictured)
2. Select **Subscriptions** from the menu
3. Click on the Name of the Subscription: **EB Get Custom Properties**
Edit Subscription Conditions

We are going to enter a few Conditions to really limit this Event Subscription down to a single Event State and Phase. The Condition builder allows us to choose "All, Any or None" complex criteria. For this example we will:

1. Choose the **Conditions** tab for the subscription
2. Select the **Run based on conditions** radio button
3. Click on the **Clause...** dropdown
4. Select the **All of the following** radio button to start a new set of conditions

Choose Lifecycle State Condition
We want to trigger this event subscription now only for a specific State.

1. Click the **Clause**... dropdown
2. Expand the **Data** option
3. Expand the **Lifecycle state** option
4. Select **Lifecycle state name**

If you take a moment to scroll through this list, you can see that there are many attributes you can use for configuring these conditions!

**Set the Lifecycle State Name**

![Set the Lifecycle State Name](image)

To finish setting up the Lifecycle State criteria, we need to select a value:

1. Choose the condition of **Equals**
2. Click the value dropdown to show the list
3. Click the dropdown to choose a **Constant** value
4. Scroll down (quite a ways) until to locate the **VMPSMasterWorkflow32.Requested** state and select it. **Note** the very long list of possible Event States there are now to choose from! This list includes states for all types of Event Topics.
5. Choose the **+Add expression** link to add additional criteria to our condition

Did you notice just how many options there are as you were scrolling through the long list of Lifecycle States? Many more than the original six (6) we had in past versions using the workflow stubs!
Add Remaining Criteria

Remember in our "trace" of the Events that were fired earlier that there were multiple Phases for each State? In order to have only a single step in our VM request lifecycle trigger a vRealize Orchestration workflow, we must further narrow down our condition by adding more criteria. If we left it as-is, we would still have our workflow called three (3) times. Finish adding these criteria, just as you did for the Lifecycle State name, so the workflow only gets called for the POST Phase when Virtual Machines are requested.

1. Under the **Cause**... dropdown, select **State Phase** (found under **Data > Lifecycle State**) (Note: You may need to scroll right on the selection box to scroll down to Lifecycle State)
2. Select **Equals** for the operator
3. Choose a Constant value of **POST**
4. Add another expression (not pictured)
5. Select **Machine type** (under **Data > Machine**) Note: you may have to scroll to the right on the bottom scroll bar to navigate down on the right.
6. Select **Equals**
7. Choose a Constant value of **Virtual Machine**

Your final set of conditions should look like the screenshot above. When you are done, click the **Finish** button in the lower right corner of the screen to save and close the Workflow Subscription page.
Test One More Time

Now, let's test out our updates to the Subscription and the Properties we have made! Click on the second Chrome instance you should still have open from the Task bar. If it was closed, you can open a new private window again.

1. Ensure you are logged in as devuser (remember, the password is VMware1! in case you find yourself logged out)
2. Select the Catalog tab
3. Request the CentOS item again
4. Enter a Request Description and Submit the request (not pictured)
5. Click the OK button on the Request submission confirmation dialog (not pictured)
Check Results in vRealize Orchestrator

1. Click on the last workflow execution
2. In the Logs, notice that **State** and **Phase** match our Subscription criteria
3. All of **Properties** for this Request are now listed (scroll down to see them all) - they were all passed to our workflow and are available to use as part of our custom logic in the workflow!

Now that we've covered the basics, we will move on to our real use cases!
Clean Up Provisioned Workloads

As a final step in this module, please return to vRealize Automation in your second browser window, where you are logged in as devuser. This will free up additional lab resources so you can continue on to other modules. If you don't plan to continue to another module, you can skip this step and simply End your lab.

1. Click on the **Items** tab
2. Select a row with the first deployment (don't click directly on the Name, or you'll be taken into the view details page)
3. Click the **Actions** dropdown at the top of the grid
4. Choose **Destroy** and Submit to confirm
5. Repeat the steps above for **ALL** deployments.
6. Close this browser window

The deployments will disappear from the list after the Destroy Request completes. You can use the Refresh button at the bottom of the list to see them disappear.
Find and Import Blueprints from the VMware Solutions Exchange

Another way to extend the functionality of vRealize Automation is to import Blueprints from the VMware Solution Exchange. Solution Exchange is the place to go to find Blueprints and plugins developed by partners and other community members.

Let's get going and import a new XaaS Blueprint from the Solution Exchange

VMware Solution Exchange

The Solution Exchange provides a search function to help you find what you are looking for. In this case we searched for and found the vRealize Automation Services Pattern Blueprint.
For this lab, the Blueprint has already been downloaded to the \c:\HOL\HOL-1921-03-CMP folder.

Go to [https://marketplace.vmware.com/vsx/](https://marketplace.vmware.com/vsx/) to explore all of the content that is available via the Solution Exchange.

**Launch the CloudClient**

1. Double-click the CloudClient from the desktop

**Log in to vRealize Automation Using the CloudClient**

1. Type the following command to login to the vRealize Automation instance

   ```
vra login userpass --user cloudadmin@corp.local --password VMware1! --server vra-01a.corp.local --tenant vsphere.local
   ```
Import the New Blueprint

1. Type the following command to import the new Three Tier Service Blueprint

   vra content import --path C:\hol\HOL-1921-03-CMP\Three-TierServicesPattern.zip --resolution OVERWRITE

2. Type `exit` to close the CloudClient (Not Shown)

Open Google Chrome

Now that the Blueprint has been imported, let's check out what has been created in vRealize Automation as a result.

1. Double-click on the **Google Chrome** icon on the desktop
2. Select vRealize Automation tab if it does not launch automatically
Log in to vRealize Automation

1. Enter **cloudadmin** as the username
2. Enter **VMware1!** for the password
3. Click **Sign in**

Open the Newly Imported Blueprint

1. Select the **Design** tab
2. Click on the **Three-Tier Services Pattern** Blueprint to open the blueprint
Examine the Blueprint

1. Click the - icon until the entire Blueprint is visible on the Design Canvas

Note that all of the components needed to build this application have been imported, including software components and their dependencies. After importing the blueprint, the base vSphere machines would need to be modified to select the proper template for your environment. Since you haven't done that customization yet, this blueprint isn't quite ready to deploy. To learn more about blueprint creation and configuration, consider checking out HOL-1921-SDC-02!

2. After you are done examining the blueprint, click **Cancel** to close the Blueprint
Another resource for downloading Blueprints is VMWARE {code}. The code site allows community members to post and share vRealize Automation Blueprints as well as workflows and other content for VMware solutions.

Visit https://code.vmware.com to learn more.
Conclusion

In this module you learned about the various options for extensibility in vRealize Automation. You also learned how to create a basic subscription with the Event Broker. Finally, we closed off with learning how to leverage the CloudClient to import content from the VMware Solutions Exchange.

You've finished Module 1

If you are looking for additional information on deploying vRealize Automation, try one of these:

- Or use your smart device to scan the QR Code above

Proceed to any module below which interests you most, or move to the next step to end your lab.

- **Module 2 - Infoblox IPAM Integration with vRealize Automation** (45 minutes)
  Discover IPAM integration between VMware NSX, Infoblox NIOS and vRealize Automation. Learn how the On-Demand network in NSX consumes Infoblox IPAM for workloads deployed through vRealize Automation.
- **Module 3 - Integrating vRealize Automation with ServiceNow** (45 minutes)
  Examine the new ServiceNow integration capabilities in vRealize Automation 7. Learn how to provision and perform day 2 operations on resources in the ServiceNow portal.
- **Module 4 - Puppet Integration with the Configuration Management Framework** (45 minutes)
  Learn how vRealize Automation 7.4’s Configuration Management Framework integrates with Puppet to install and configure software on a newly provisioned machine.
- **Module 5 - Creating and Managing Custom Resources with vRealize Automation** (45 minutes)
  Learn how to use vRealize Orchestrator Custom Resources to extend vRealize Automation's ability to manage existing objects throughout your business. From user accounts to coffee makers, Custom Resources are a powerful way to bring automation to all aspects of your business.
- **Module 6 - Modifying Blueprint Request Forms with Custom Forms Designer** (45 minutes)
  Learn how to extend and customize existing virtual machine blueprints to leverage the new Custom Forms Designer introduced in vRealize Automation 7.4. Make your request forms more dynamic and responsive with ClarityUI from VMware. Covering topics such as retrieving external values with vRealize Orchestrator, Customizing Look and Feel with CSS, and external validation.
How to End Lab

To end your lab click on the **END** button.
Module 2 - Infoblox IPAM Integration with vRealize Automation (45 Minutes)
Introduction

Learn how to use the new vRealize Automation Infoblox IPAM integration, including the new ability to leverage NSX on-demand routed networks, load balancers, and NAT networks.

This HOL-1921-03-CMP lab is broken up into 5 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 you have 90 minutes available per session.

If you have not already reviewed it, the 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 to review this information again.

You will need about 45 minutes to complete this module.
Reviewing the Infoblox Configuration

Before getting into the nuts and bolts of the new integration enhancements, let's take a quick walkthrough in the base configuration of the Infoblox NIOS environment to see which constructs will be consumed.

We will use two types of Infoblox networks to showcase the integration.

- IPv4 Networks
- Network Containers and Network Leafs

A network container is a parent network that contains other network containers and leaf networks. A leaf network is a network that does not contain other networks. Infoblox Grid Manager uses network containers to group IPv4 and IPv6 networks. After you select an IPv4 network container from the IPAM tab, Grid Manager displays it in the Net Map (network map) panel, by default. Net Map provides a high-level view of your IPv4 network address space.

Open the Chrome Browser

Double-click on Google Chrome browser icon from the desktop.
Open the Infoblox Grid Manager

1. Open the **HOL Admin** bookmark folder
2. Select **Infoblox Grid Manager** from the bookmark list

Log in to the Infoblox Grid Manager

1. Enter **admin** for the Username
2. Enter **VMware1!** for the Password
3. Click **Login**
View the Infoblox Networks

1. Click the **Data Management** tab
2. Click the **IPAM** sub-tab
3. The 172.16.100.0/24 network is an Infoblox IPv4 network. In this environment, this network is mapped to the NSX transit network.
4. The 172.16.161.0/24 network is a network container. This will be used to hold our leaf networks that are created when using an NSX on-demand routed network in vRealize Automation.
5. The 192.168.110.0/24 network is an Infoblox IPv4 network. In this environment, this network is mapped to the default external VM network.
Reservation Preparation

In order to utilize the networks managed by Infoblox, the reservation will need to be modified. The reservation needs to be modified by mapping the NSX network to the Infoblox Network Profile.

Log In to vRealize Automation

1. Open a new tab in the browser
2. Open the vRealize Automation bookmark
3. Enter cloudadmin as the username
4. Enter VMware1! for the password
5. Click Sign In
Navigate to Reservations

1. Select the **Infrastructure** tab
2. Select **Reservations** from the menu on the left

Edit the Development Reservation

1. Select **Reservations** from the menu on the left
2. Click on **Development Reservation** to open the reservation to edit the network mapping
Edit Network

1. Select the **Network** tab
2. Add another check box for the **vxw-dvs-40-virtualwire-1-sid-5000-Transit Network**
3. Select **Infoblox External Network Profile** for the **Transit Network**

This will associate the 172.16.100.0/24 network viewed earlier with the Distributed Virtual Switch created by the NSX Transit Network

4. Select **RegionA0-Global-TZ** from the **Transport Zone** drop-down
5. Scroll down to view additional options
6. Select the Checkbox next to **NSX-DLR** to enable the DLR for this reservation
7. Select **Infoblox External Network Profile** for the **NSX-DLR**

This will allow Infoblox to assign IP Addresses to any new interfaces on the NSX-DLR created by an on-demand routed NSX network.

7. Click **OK**
Creating a Blueprint to use Infoblox

Edit the Infoblox CentOS Blueprint

We will now configure the Infoblox CentOS blueprint to leverage the network profile we enabled in the previous step.

1. Select the **Design** Tab
2. Select the **Description** area for the **Infoblox CentOS** blueprint
3. Select **Edit**

Select the Network Object

1. Select the existing network on the canvas named **DefaultExternalNetwork**
Select the Network Profile

1. Select the ellipsis next to the existing Default External Network Profile

Choose the Infoblox External Network Profile

1. Select the Infoblox External Network Profile from the Network Profile list
2. Select OK
Explore the Blueprint

1. Select the CentOS component to open the details pane

Modify the Build Information

1. Select the Build Information tab
2. Select the Drop Down on the Clone from snapshot menu item
3. Select application authoring snapshot

Select the Network Tab
1. Select the **Network** tab
2. Select **New**

### Select the Network Profile

1. Select the **Drop Down** menu next to the **0 Network**
2. Select **InfobloxExternalNetworkProfile**
3. Select **OK**

### Navigate to Merged Properties

1. Select the **Properties** tab
2. Note the the Infoblox-Default Property Group has already been assigned to this Blueprint
3. Click **View Merged Properties**
View Merged Properties

1. Note the properties that will be assigned to this Blueprint. This property group was created when the Infoblox Setup Workflow was run from the Infoblox vRealize Orchestrator Plug-in. Please see the Infoblox documentation for more information about the settings contained here.

2. Click **Close** to continue

Return to the Design Canvas

1. Click the **Blank Area** to the right of the **CentOS** component on the Design Canvas to hide the lower panel.
Observing our Blueprint

Your blueprint should now look like the following image

Our InfobloxExternalNetwork is selected and bound to the CentOS blueprint. This will allow our Blueprint to consume this specific Network, via the network profile.

Select Finish

1. Select Save
Configuring Infoblox Integration with an NSX On-Demand Load Balancer

As configured, the Infoblox CentOS Blueprint will deploy a CentOS machine with an Infoblox assigned IP address. One of the advantages of vRealize Automation's integration with NSX is the ability to dynamically create network topologies that support complex applications. Let's configure a load-balanced blueprint with Infoblox assigned IP addresses.

Add a Load Balancer

1. Click on a blank space to minimize the detail pane
2. Select Network & Security from the Categories menu
3. Drag the On-Demand Load Balancer to the Design Canvas
Configure the Load Balancer

1. Select **CentOS** from the Member drop-down
2. Select **InfobloxExternalNetworkProfile (NIC 0)** from the Member network drop-down to map the load balancer to the NIC on the CentOS machine
3. Select **InfobloxExternalNetworkProfile** from the VIP network drop-down to ensure the VIP is assigned by Infoblox and placed on the correct network
4. Click **New**

Configure the Virtual Server

1. **Protocol:** HTTP
2. **Port:** 80

Settings:
- Use default settings for all other values
- Customize

[OK] [Cancel]
The load balancer needs to be configured with the protocol and other information to properly load-balance the application.

1. Note the default setting will load-balance the CentOS machines over port 80
2. Click OK to save this virtual server

**Set the Virtual Machine Properties**

A load balancer doesn't do any good when there is only one machine deployed. The CentOS machine properties need to be modified to allow for more than one instance.

1. Click on the CentOS machine on the Design Canvas to bring up the details pane
2. Click on the General tab
3. Increase the Maximum Instances to 3
4. Click Finish to save the blueprint
Open a New Browser Window

With the Blueprint configured, it is time to deploy a load-balanced application with IP address managed by Infoblox.

1. Right-click on the Chrome Browser in the task bar
2. Select **New incognito window** to open a new browser window and log in with a different user
Navigate to vRealize Automation

1. Open the **vRealize Automation** bookmark
2. Ensure **corp.local** is selected as the domain
3. Click **Next**

**Note:** Because of the nature of caching, you may be sent directly to the login screen for vRealize Automation. This is acceptable, continue to the next step!
Log In

1. Enter `devuser` as the username
2. Enter `VMware1!` as the password
3. Click Sign In

Request the Blueprint

1. Select the Catalog tab
2. Click Request on the Infoblox CentOS Blueprint
Increase the Instance Count

To make the load-balancer useful, more than one instance of the machine will be deployed.

1. Click on CentOS to modify the details of the machine
2. Increase the **Instances** to 2
3. Click **Submit**
4. Click **OK** on the request confirmation screen (not shown)

Open the Request
1. Select the Requests tab
2. Click the request number to open the request status

Monitor the Request

![Request Information]

1. Click Execution Information to view the details of the request
2. Refresh the status until the first two lines show successful

Switch Browser Windows

![Browser Windows]

While the request proceeds, we can look at the changes that been made in Infoblox as a result of the request.

1. Select the first instance of the Chrome Browser
Navigate to Infoblox Cloud VMs

1. Select the **Infoblox Grid Manager** browser tab
2. If present, this warning can be ignored
3. Select the **Cloud** tab
4. Select the **VMs** sub-tab
5. Note the IP addresses that were allocated to this deployment.
   1. One IP address for the VIP on the NSX edge
   2. One IP address for each of the VM's

Switch Browser Windows

1. Select the second browser window to return to the submitted request
Refresh the Request Status

1. Refresh the status periodically until all of the status rows show **Successful** (Note: this could take a few minutes)
2. Click OK
View the New Items

1. Select the **Items** tab
2. Expand the deployment
3. Note the VIP IP address that was assigned by Infoblox. We will use this IP to test the new load-balancer.

This deployment created an NSX Edge with a VIP for load-balancing and two new VM's.

Test the Application

1. Open a new browser tab
2. Type the IP address of the VIP from the previous step in the address bar
3. Note the IP address on the web page
Refresh the Browser

1. Refresh the browser a few times
2. Note that the IP address and host name change as the load balancer round-robins the requests

Congratulations, you have just deployed an on-demand NSX load-balancer with Infoblox IP address management

3. Close the browser tab

Destroy the Deployment

To save resources, destroy the deployment.

1. Select the space next to the name of the deployment.
2. Click **Destroy**
Submit the Request

1. Click **Submit**
2. Click **OK** on the confirmation screen (not shown)
Infoblox Integration with an NSX On-Demand Routed Network

In the previous steps we deployed a blueprint that created an on-demand load balancer. Next, we will use NSX to create an on-demand routed network with IP addresses provided by an Infoblox Network Container

Switch Browser Windows

1. Switch to the first browser window

Open the Blueprint

1. Select the **vRealize Automation** browser tab
2. Click on the **Infoblox CentOS** blueprint to open it for editing
Select the Load Balancer

1. Select the **On-Demand_Load_Balancer** from the design canvas

Minimize the Details Pane

1. Select the **down arrow** to minimize the details pane

Delete the Load Balancer

1. Click the **red X** to remove the load balancer
2. Click **Yes** on the confirmation screen (not shown)
Select the External Network

1. Select the **InfobloxExternalNetwork** from the design canvas

Minimize the Details Pane

1. Select the **down arrow** to minimize the details pane

Delete the Network

1. Click the **red X** to remove the InfobloxExternalNetwork
2. Click **Yes** on the confirmation screen (not shown)
Decrease the Maximum Instances

For a routed network deployment, we only need one VM in the routed network. Decrease the maximum number of instances.

1. Select the CentOS machine from the Design Canvas
2. Decrease the Maximum Instances to 1
3. Click on an empty space on the design canvas to minimize the details pane
Add the On-Demand Routed Network

1. Select **Network & Security** from the Categories menu
2. Drag the **On-Demand Routed Network** to the Design Canvas

Select the Parent Network Profile

The Parent Network Profile determines which network the machine will be placed on. The profile also determines which IPAM solution provides the IP addresses.

1. Click the ... to open the Network Profile Selection dialogue box
Select the Network Profile

1. Select the **Infoblox Routed Network Profile**
2. Note that the IPAM Endpoint of this profile is Infoblox NIOS
3. Click **OK**
1. This Network profile is designed for NSX On-demand Routed Networks served by Infoblox IPAM. This network profile will create up to 32 networks
2. Click on the **CentOS** machine to add it to the network
Add the Network Adapter

The machine needs to have a new network adapter that is assigned to the InfobloxRoutedNetworkProfile

1. Select the **Network** tab
2. Click the **+New** button
3. Select the **InfobloxRoutedNetworkProfile** from the drop-down
4. Click **OK** to save the network configuration
5. Click **Finish** to save the changes to the Blueprint

Switch Browser Windows

Now that the blueprint has been modified to deploy to a on-demand routed network, we will deploy the blueprint to test.

1. Select the second browser window from the task bar
Request the Blueprint

1. Select the Catalog tab
2. Click Request on the Infoblox CentOS blueprint to start the request
Submit the Request

1. Click Submit
2. Click OK on the request confirmation screen (not shown)

View the Request
1. Select the **Requests** tab
2. Click the **request number** to open the request. Note that your request numbers may be different if you have completed other labs prior to this one.

**Monitor the Request**

1. Click **Execution Information** to view the details of the request
2. Refresh the status until the first 3 lines show **Successful**

Now that the allocations are complete, we will switch over to the Infoblox Grid Manager to view IP allocations

**Switch Browser Windows**

1. Select the first browser window
Navigate to the IPAM Networks View

1. Switch to the **Infoblox Grid Manager** browser tab
2. Select the **Data Management** tab
3. Open the **172.16.161.0/24** network container
View the Leaf Network

1. Hover over the block to view the leaf network's details

This Leaf Network was created by our deployment of the NSX on-demand routed network. This network has 8 IP addresses total as configured by the network profile's 255.255.255.248 subnet mask. The next deployment would create another network adjacent to this one. When creating on-demand routed networks, it is important to adjust your networks to ensure you have enough room in the network for each machine while maintaining enough networks to support your number of deployments.

Switch Browser Windows

1. Select the second browser window from the task bar
Monitor the Request

1. **Refresh** the status of the request until all of the items are deployed successfully (Note: this may take several minutes to complete)
2. Click **OK**

Review the Deployment

1. Select the **Items** tab
2. Expand the deployment to see the details of the components
3. Make note of the IP Address assigned to the machine

Test the Deployment

1. Open a new browser tab
2. Enter the IP address from the previous step in to the address bar
3. Note that the IP addresses match in this case because the machine was deployed to a routed network and not a NAT or Load-balanced network
4. Close the browser tab

The first address in the leaf network is assigned to the new DLR interface by Infoblox. In this case, 172.16.161.1 is the new gateway for these virtual machines.

Destroy the Deployment
1. After closing the previous screen, you should return to your devuser vRealize Automation tab
2. Select the space next to the name of the deployment (clicking the name of the deployment will open a different screen)
3. Select **Actions**
4. Click **Destroy**

**Submit the Request**

1. Click **Submit**
2. Click **OK** on the request confirmation page (not shown)
Infoblox Integration with an NSX On-Demand NAT Network

Previously we created a blueprint that will deploy a machine with a routed network. vRealize Automation now allows for the creation of on-demand NAT networks with IP addresses managed by Infoblox.

Switch Browser Windows

1. Select the first browser window from the task bar

Edit the Blueprint

1. Select the vRealize Automation browser tab
2. Click on the Infoblox CentOS blueprint to open it for editing
The blueprint will need to be modified to remove the routed network and attach a new on-demand NAT network.

**Select the Routed Network**

1. Select the `InfobloxRoutedNetwork`

**Minimize the Details Pane**

1. Select the **down arrow** to minimize the details pane

**Delete the Routed Network**

1. Click the **red X** to delete the network
2. Click **Yes** on the confirmation dialogue box (not shown)
Add the On-Demand NAT Network

1. Select the **Network & Security** item from the Categories menu
2. Drag the **On-Demand NAT Network** to the design canvas

Select the Parent Network Profile

1. Select the **Parent network profile** selector
2. Select **Infoblox One:Many NAT Network** (not shown)

Note that this network will create a single routable IP address assigned by Infoblox and a /24 network created dynamically in Infoblox
1. **Click OK**

### Add the Network Adapter

1. Select the **CentOS** machine on the Design Canvas
2. Click the **Network** tab
3. Click **+New** (shown greyed out)
4. Select **InfobloxOneManyNetwork** from the Network drop-down to assign the new network adapter to the NAT Network
5. Click **OK**
6. Click on an empty spot on the design canvas to minimize the details pane
Select the NAT Network

1. Select the **InfobloxOneManyNetwork** on the design canvas to bring up the details pane

Configure the NAT Rules

1. Select the **InfobloxOneManyNetwork** on the design canvas to bring up the details pane

2. Click on the **NAT Rules** tab

3. Click on the **New** button

4. Fill in the details for the NAT rule:
   - **Name**: http
   - **Component**: CentOS (NIC 0)
   - **Source Port**: 80
   - **Destination Port**: 80
   - **Protocol**: TCP

5. Click on the **OK** button

6. Click on the **Finish** button
In a One to Many NAT network there is a single public IP address that translates traffic to multiple internal IP addresses. Traffic is designated for a destination by ports. In this case we will redirect all port 80 traffic to the CentOS machine.

1. Select the **NAT Rules** tab
2. Click +**New** (shown greyed out)
3. Configure the following settings for the rule:
   - Name: http
   - Component: CentOS (NIC 0)
   - Source Port: 80
   - Destination Port: 80
   - Protocol: TCP
4. Click **OK**
5. Click **Finish** to save the changes to the blueprint

**Switch Browser Windows**

Now that the blueprint is configured to use an on-demand NAT network, we will deploy the blueprint and view the changes made in Infoblox.

1. Select the second browser window

**Request the Blueprint**

1. Select the **Catalog** tab
2. Click **Request** on the Infoblox CentOS blueprint

Submit the Request

For this request the default selections are OK.

1. Click **Submit**
2. Click **OK** on the request confirmation (not shown)

Open the Request
1. Select the **Requests** tab
2. Click on the **request** to open the details. Again, note that your request numbers may be different if you have completed other labs prior to this one

**View the Request Details**

![Request Details](image)

1. Click **Execution Information** to view the request details
2. **Refresh** the status until the first four items show successful (This may take a few minutes)

**Switch Browser Windows**

Now that the network components have been deployed, we will go in to the Infoblox Grid Manager to view the newly created network views

1. Select the first browser window from the task bar
Switch to the Infoblox Browser Tab

Review On-Demand NAT Network

Switch Browser Windows

Now we can go back and check the status of our deployment and test the application

1. Select the second browser window

View the Request Status

1. **Refresh** the status until all items show **Successful** (this could take a few minutes to complete)
View the Deployment

1. Select the **Items** tab
2. Expand the deployment
3. Note that the IP address given to the machine is a NAT’d address assigned by Infoblox
4. Click on **Edge** to open the details about the NSX Edge
View the NSX Edge Details

The Edge hold the NAT rules for the application. This is where we will find the public IP address for our application.

1. Click and drag the column separator to make the Source Address column readable
2. Make a note of the Source Address. We will use this again
3. Click Close

Test the Application
When we created the blueprint, we created a NAT rule for port 80. Let's test the application to make sure everything is working as expected.

1. Open a new browser tab
2. Type the IP address from the previous step in to the address bar
3. Note the IP address of the machine is a NAT'd address
4. Close the browser tab

Optional: Destroy the Deployment

If you plan on completing another module, destroy your deployment to save resources

1. Click on the deployment (Note: if you click on the deployment name it will open an different window)
2. Select **Actions**
3. Click **Destroy**
4. Click Submit on the new request screen (Not shown)
5. Click **OK** on the request confirmation screen (Not shown)
Conclusion

In this module you learned about prerequisites necessary to configure Infoblox integration with vRealize Automation. You also learned how to configure and deploy a blueprint with on-demand NAT, Routed, and Load Balanced NSX networks with Infoblox IPAM managed IP networks.
You've finished Module 2

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

- Visit the vRealize Automation Documentation Center http://bit.ly/2m3IflX
- Or use your smart device to scan the QR Code above

Proceed to any module below which interests you most, or move to the next step to end your lab.

- **(30 minutes)** Explore the limitless flexibility that extensibility brings to vRealize Automation, and learn about different ways you can implement and leverage it to streamline and automate complex processes.
- **(45 minutes)** Examine the new ServiceNow integration capabilities in vRealize Automation 7. Learn how to provision and perform day 2 operations on resources in the ServiceNow portal.
- **(45 minutes)** Learn how vRealize Automation 7.4’s Configuration Management Framework integrates with Puppet to install and configure software on a newly provisioned machine.
- **(45 minutes)** Learn how to use vRealize Orchestrator Custom Resources to extend vRealize Automation’s ability to manage existing objects throughout your business. From user accounts to coffee makers, Custom Resources are a powerful way to bring automation to all aspects of your business.
- **(45 minutes)** Learn how to extend and customize existing virtual machine blueprints to leverage the new Custom Forms Designer introduced in vRealize Automation 7.4. Make your request forms more dynamic and responsive with ClarityUI from VMware. Covering topics such as retrieving external values with vRealize Orchestrator, Customizing Look and Feel with CSS, and external validation.

How to End Lab
To end your lab click on the END button.
Module 3 - Integrating vRealize Automation with ServiceNow (45 Minutes)
Introduction

Learn how to configure and consume vRealize Automation catalog items via the ServiceNow integration.

This HOL-1821-03-CMP lab is broken up into 5 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 you have 90 minutes available per session.

If you have not already reviewed it, the 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 to review this information again.

You will need about 45 minutes to complete this module.
Install and Configure MID Server

In this lesson you will watch a video that outlines how to install and configure the Management, Instrumentation, and Discovery (MID) server for ServiceNow. The MID Server facilitates communication between the ServiceNow platform and external applications, such as vRealize Automation. The MID Server allows for secure communication, data collection, and probing.

Video: Installation and Configuration of the MID Server
Configure ADFS and ServiceNow SSO

Once the MID Server has been set up, you will now begin to configure Active Directory Federation Services (ADFS) for ServiceNow SSO to handle SAML Token authentication between vRealize Automation and ServiceNow. This service will enable your users to log in to ServiceNow and have their credentials passed automatically to vRealize Automation when a catalog request is made. This will avoid the users having to log in to multiple systems to request and manage their catalog items.
Configure vRealize Automation for ADFS

This video will review the steps required to configure vRealize Automation to use ADFS for SAML Token authentication with ServiceNow. This is necessary to allow for the user session in ServiceNow to translate to vRealize Automation and grant them access to their catalog items and actions.
Configuring vRA to use ADFS

**Note:** The XML template for the Claim Rule transformation step at 5:50 in the video can be found on page 8 of the VMware Identity Manager Integration with Active Directory Federation Services document.
Install and Configure vRealize Automation Plugin for ServiceNow

Now that the MID Server, and ADFS has been configured for both ServiceNow and vRealize Automation, you can now move forward with installing and configuring the vRealize Automation plugin for ServiceNow.

Setting up the vRealize Automation Plugin for ServiceNow

The ServiceNow plugin for vRealize Automation can be found on the VMware Solution Exchange as the VMware vRealize Automation Plug-In for ITSM.
Conclusion

Congratulations on completing Module 3 - Integrating vRealize Automation with ServiceNow. Now that you have configured the vRealize Automation plugin for ServiceNow, your users can request, deploy, and manage catalog items in the ServiceNow portal. ServiceNow can now act as the front end for your users who wish to consume vRealize Automation catalog items and not have to log into a separate tool to do so.

You've finished Module 3

If you are looking for additional information on deploying vRealize Automation, try one of these:

- Or use your smart device to scan the QR Code above

Proceed to any module below which interests you most, or move to the next step to end your lab.

- (30 minutes) Explore the limitless flexibility that extensibility brings to vRealize Automation, and learn about different ways you can implement and leverage it to streamline and automate complex processes.
- (45 minutes) Discover IPAM integration between VMware NSX, Infoblox NIOS and vRealize Automation. Learn how the On-Demand network in NSX consumes Infoblox IPAM for workloads deployed through vRealize Automation.
- (45 minutes) Learn how vRealize Automation 7.4’s Configuration Management Framework integrates with Puppet to install and configure software on a newly provisioned machine.
- (45 minutes) Learn how to use vRealize Orchestrator Custom Resources to extend vRealize Automation’s ability to manage existing objects throughout your business. From user accounts to coffee makers, Custom Resources are a powerful way to bring automation to all aspects of your business.
• (45 minutes) Learn how to extend and customize existing virtual machine blueprints to leverage the new Custom Forms Designer introduced in vRealize Automation 7.4. Make your request forms more dynamic and responsive with ClarityUI from VMware. Covering topics such as retrieving external values with vRealize Orchestrator, Customizing Look and Feel with CSS, and external validation.
Module 4 - Puppet Integration with the Configuration Management Framework (45 Minutes)
Introduction

Learn how to deploy a WordPress application using Puppet and vRealize Automation's Configuration Management Framework. This HOL-1921-03-CMP lab is broken up into 5 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 you have 90 minutes available per session.

If you have not already reviewed it, the 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 to review this information again.

In this module you will act as the cloud admin and a developer for Rainpole to configure and deploy a WordPress application.

You will need about 45 minutes to complete this module.
Puppet Installation and Prerequisites

vRealize Automation's Configuration Management Framework can take advantage of third-party configuration management tools that you may already leverage in your environment. In this case you will be using Puppet Enterprise, which has been installed and configured in the environment.

In this module the following items have been configured:

- Puppet Enterprise installed and configured with several roles
- Puppet plug-in installed in vRealize Orchestrator from the Orchestrator Control Center
- Puppet endpoint added to vRealize Automation

Puppet Roles

In Puppet, the roles and profiles method allows an administrator to configure an endpoint with all of the software and configuration elements needed for immediate use by the developer or end-user. There are three levels used in this method to define a system configuration.

- Component Modules: These are modules that manage a single technology (For example, WordPress, Apache, MySQL)
- Profiles: This class uses multiple components modules to configure a complete stack.
- Roles: This class uses multiple profiles to build the stack

WordPress Role

In this lab, WordPress will be installed with all of elements needed on a single machine. The following is a high-level overview of the wordpress_server role that is pre-defined in Puppet Enterprise

- Role::wordpress_server
  - include apache
  - include mysql::server
  - include mysql::client
  - include apache::mod::php
  - include stdlib
  - include mysql::bindings::php
  - include profile::firewall
    - class profile::firewall
      - ensure => 'stopped'
  - include profile::wordpress
The outline shows that the role of wordpress_server includes several modules with their default configurations. The role also includes two profiles, firewall and wordpress. These profiles have some configuration elements that turn off the firewall, change the install directory for WordPress, change the install source for WordPress, and define the version of WordPress to be installed.

### Puppet Plug-in for vRealize Automation

The Puppet Plug-in for vRealize Automation is installed through the vRealize Orchestrator Control Center. The plug-in installs the workflows shown in the image. The notable workflows used in this lab are the following:

- **Add a Puppet Enterprise Master**: Adds a Puppet Enterprise Master as an endpoint for Orchestrator. This workflow is called by vRealize Automation when adding a Puppet endpoint.
- **Install PE Agent with Role**: Installs the Puppet Enterprise Agent on the new endpoint and configures the node with the role chosen in the deployment.
- **Purge PE Agent Node**: The workflow removes the node and certificate from the Puppet Enterprise console.
Creating the Configuration Management Framework Endpoint

The Configuration Management Framework Endpoint allows vRealize Automation to communicate with the Configuration Management server. vRealize Automation will allow the administrator to define multiple Configuration Management servers to cater to different parts of the business.

Open Chrome Browser from Windows Quick Launch Task Bar

1. Click on the Chrome Icon on the Windows Quick Launch Task Bar.

Log In to vRealize Automation

1. Enter **cloudadmin** as the username
2. Enter **VMware1!** as the password
3. Click **Sign in**
Navigate to vRO Configuration

1. Select the **Administration** tab
2. Select **vRO Configuration** from the menu
Create a new Puppet Endpoint

Let's create a new vRealize Automation Configuration Management Framework Endpoint for Puppet

1. Select **Endpoints** from the menu on the left
2. Click **+New**
Select Puppet as the Plug-in Type

When the Puppet Plug-in is installed in vRealize Orchestrator, the Puppet plug-in type is enabled in vRealize Automation

1. Click the **drop-down arrow** to expand the Plug-in list
2. Select **Puppet** from the Plug-in list
3. Click **Next**
Enter a Name for the Puppet Endpoint

1. Enter **HOL-Puppet** in the Name field
2. Click **Next**
Enter the Puppet Endpoint Details

1. Enter **HOL-Puppet** as the Display Name
2. Enter **puppet-01a.corp.local** for the Hostname
3. Enter **root** as the SSH and RBAC Username
4. Enter **VMware1!** for the password
5. Ensure **No** is selected in the sudo command selection
6. Click **Finish**

**Note:** This action may take a few moments to complete as it is communicating with the Puppet Master to validate the initial configuration.

**Note:** A user named "root" was created in the Puppet Enterprise Console. Puppet's implementation of the framework requires that a user with the same name exist in both SSH and the Puppet Console. If you create a user with limited access to the SSH host, then you would enable the sudo command selection option to elevate commands run on the host.
Endpoint Created

Note the newly created endpoint along side the existing puppet endpoint. The Configuration Management Framework allows for more than one Configuration Management endpoint.

The next step is to create a blueprint that will utilize the Puppet Endpoint to configure the WordPress application on a CentOS template.
Create a Puppet Managed Blueprint

In this section, you will create a Blueprint that will take advantage of the Puppet Endpoint to configure the node after deployment.

Navigate to the Blueprint List

1. Select the Design tab
2. Click on the description field of the CentOS Blueprint to select the Blueprint
3. Click Copy
Create a Copy of the Blueprint

1. Enter **Puppet - CentOS** for the Blueprint name
2. Enter a description for your new Blueprint
3. Click **OK**
Configure the Blueprint

A snapshot has been created with a custom YUM repository (a software download location for CentOS and Redhat Linux based operating systems) containing all of the packages needed to deploy your version of Apache, MySQL, and WordPress. We need to select the appropriate snapshot.

1. Select the CentOS machine on the Design Canvas
2. Select the Build Information tab
3. Select the application authoring snapshot from the Clone from snapshot drop-down
4. Click the down arrow to collapse the details pane and expose the Design Canvas again
Add Puppet to the Machine

Now that the correct snapshot is selected, we need to add the Puppet agent to the machine.

1. Select **Configuration Management** from the Categories menu
2. Drag **Puppet** to the machine

**Note:** A configuration screen will display automatically after dragging the item to the blueprint. If you click off of the configuration screen, your Puppet object will display the red and white exclamation point, as shown above. You can reenter the screen by clicking on the Puppet_1 object again.
Configure Puppet

This Blueprint will be used for our developers to provision their environments.

1. Type Dev_Puppet in the ID field. (Note: The ID cannot contain spaces)
2. Select the Server tab

Configure the Puppet Server

1. Select HOL-Puppet as the Puppet master
2. Since this will only be used by developers, select dev from the Puppet environment drop-down
3. Check the Set in request form box to allow the user to select the appropriate Puppet role for their task.

Note: Domain based auto-signing has been configured on the Puppet Enterprise server so the Shared secret field is not necessary in this case.
Configure the Managed Node

The Puppet endpoint will install the Puppet Agent on the new machine. In order to complete this task, we need to provide the appropriate credentials for the template.

1. Select the **Managed Node** tab
2. Enter **root** for the username
3. Enter **VMware1!** for the Password
4. Click **Finish**

Publish the Blueprint
When a Blueprint is first created, it is in a Draft state. We need to publish the Blueprint to make it available to be added to a catalog.

1. Select the Puppet - CentOS Blueprint by clicking on its description
2. Click Publish

Navigate to Catalog Management

Now that the Blueprint has been created, it needs to be added to a catalog for users to deploy.

1. Select the Administration tab
2. Click Administration on the left menu if necessary (Not Shown)
3. Click the Catalog Management menu item
Select the Blueprint

1. Select **Catalog Items** from the menu on the left
2. Click **Puppet - CentOS** to open the catalog item for editing
Add the Blueprint to a Service

1. Scroll down
2. Select **Infrastructure** from the Service drop-down to add this to the Infrastructure service in the end-user catalog
3. Click **OK**
Deploy and Test the New Blueprint

Now that the Blueprint has been configured, let's deploy a WordPress application as a developer.

Log Out of the Console

1. Click **Logout**

Log Out (continued)

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

1. Enter **devuser** as the username
2. Enter **VMware1!** as the password
3. Click **Sign in**

Request the Blueprint

1. Select the **Catalog** tab
2. Click **Request** on the Puppet - CentOS Blueprint

**Zoom Out Google Chrome**

1. Select the 3 vertical dots on the top right of Google Chrome
2. Reduce zoom to 80% to allow full view of Catalog

**Configure the Request**

When we configured the Blueprint, we selected the box to allow the user to choose the Puppet role appropriate to their work.

1. Select **Dev_Puppet** from the menu
2. Click ... to bring up the Role Selection menu (Note: You may have to scroll right to see the button)
Select the Role

1. Select role::wordpress_server from the menu
2. Click OK

Submit the Request

1. Click Submit to start the request
2. Select OK on Request Confirmation (not shown)
Open the Request

1. Select the Requests Tab
2. Click the request number to open the request status

Monitor the Request
We will monitor the status of the request until all components are successful. Note: This could take several to many minutes depending on the load of the environment. Deployments can at times take up to 15 minutes to complete in some cases.

1. Click **Execution Information** to view the status of all components
2. Click the **refresh** button to refresh the status
3. When all components show successful, click **OK**

**View the Deployment**

![Image of deployment status](image)

After the deployment is successful, we will go to the Items tab and view the deployment to find the IP address. We will use this IP address to test the new WordPress deployment

1. Select the **Items** tab
2. Expand the **Puppet - CentOS** deployment
3. Note the Name and IP Address of the new virtual machine.

**Open a New Browser Window**

1. Right-click Chrome in the taskbar
2. Select **New window** from the menu to launch a new browser window.

**Test the New Deployment**

1. Enter the IP address from the previous step in the address bar and hit **Enter**.

You now have an instance of WordPress configured to your standards for your developer. Let's take a look at how this looks in the Puppet Enterprise console.
Open the Puppet Enterprise Console

1. In the same browser window, open the HOL Admin bookmark folder
2. Select Puppet Console from the list

Log In to the Puppet Console

1. **User name:** admin
2. **Password:** 
3. **Log in**
1. Enter **admin** as the User name
2. Enter **VMware1!** for the password
3. Click **Log in**

**View the Puppet Inventory**

![Puppet Inventory Screenshot]

The Configuration Management Framework added the machine that we deployed to the inventory of the Puppet Enterprise server and assigned the appropriate role classes.

1. Select **Nodes** from the menu
2. Click the machine that you deployed earlier to view its properties.

**Note:** Your actual machine number may differ from the screenshot above
View the Assigned Classes

1. Click the Configuration tab, which will display the node configuration including assigned Classes
2. Note the role::wordpress_server that was selected during deployment has been assigned to this node.

Return to vRealize Automation

Please leave the Puppet Console browser open.

1. Return to vRealize Automation by clicking on the vRealize Automation browser window in the taskbar
Destroy the Deployment

When a deployment containing a Configuration Management item is destroyed, the framework kicks off a workflow to remove it from the Configuration Management service.

1. Select the area next to **Puppet - CentOS** to select the deployment
2. Click the **Actions** menu
3. Select **Destroy**
4. Click **Submit** (Not shown)
Return Google Chrome Zoom To 100%

1. Select 3 Vertical dots on the top right of Google Chrome
2. Select + symbol until back to 100%

Navigate to the Requests Tab

1. Select the Requests tab to view the destroy request status
View the Destroy Request

1. Select the Destroy Request number to open the details

Monitor the Status of the Request

1. Click Execution Information to view the detailed status of the request
2. Click on the refresh button to refresh the status until Dev_Puppet shows successful (Note: Even though the above screenshot indicates Successful already, yours could take several minutes)

Return to the Puppet Console
1. Return to the Puppet Console by clicking on the browser window with the previous hostname in the taskbar

**View the Updated Puppet Inventory**

1. Select **Nodes** from the menu to refresh the inventory
2. Note that machine has been removed from inventory by the framework
Conclusion

In this module you learned about prerequisites necessary to configure the Configuration Management Framework with Puppet Enterprise, created and configured a Puppet Endpoint, designed a Blueprint which utilized Puppet Enterprise to deploy WordPress, and deployed and tested that Blueprint.

You've finished Module 4

If you are looking for other resources around deploying vRealize Automation, try one of these:

- Visit the vRealize Automation Documentation Center http://bit.ly/2m3IflX
- Or use your smart device to scan the QR Code above

To learn more about the Puppet Enterprise Plug-in for vRealize Automation visit the following site:

https://docs.puppet.com/pe/latest/vro_intro.html

Proceed to any module below which interests you most, or move to the next step to end your lab.

- (30 minutes) Explore the limitless flexibility that extensibility brings to vRealize Automation, and learn about different ways you can implement and leverage it to streamline and automate complex processes.
- (45 minutes) Discover IPAM integration between VMware NSX, Infoblox NIOS and vRealize Automation. Learn how the On-Demand network in NSX consumes Infoblox IPAM for workloads deployed through vRealize Automation.
- (45 minutes) Examine the new ServiceNow integration capabilities in vRealize Automation 7. Learn how to provision and perform day 2 operations on resources in the ServiceNow portal.
- (45 minutes) Learn how to use vRealize Orchestrator Custom Resources to extend vRealize Automation's ability to manage existing objects throughout your
business. From user accounts to coffee makers, Custom Resources are a powerful way to bring automation to all aspects of your business.

- (45 minutes) Learn how to extend and customize existing virtual machine blueprints to leverage the new Custom Forms Designer introduced in vRealize Automation 7.4. Make your request forms more dynamic and responsive with ClarityUI from VMware. Covering topics such as retrieving external values with vRealize Orchestrator, Customizing Look and Feel with CSS, and external validation.
Introduction

Learn how to use vRealize Orchestrator Custom Resources to extend vRealize Automation's ability to manage existing objects throughout your business. From user accounts to coffee makers, Custom Resources are a powerful way to bring automation to all aspects of your business.

This HOL-1821-03-CMP lab is broken up into 5 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 you have 90 minutes available per session.

If you have not already reviewed it, the 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 to review this information again.

In this module you will act as the cloud admin and a developer for Rainpole to configure an Active Directory User custom resource and deploy that resource.

You will need about 45 minutes to complete this module.
Create a Custom Resource From a vRealize Orchestrator Workflow

Custom Resources utilize vRealize Orchestrator's object types to create manageable items in vRealize Automation. To get started, a Custom Resource needs to be created.

Open Chrome Browser from Windows Quick Launch Task Bar

1. Click on the Chrome icon in the task bar to open the browser

Sign In to vRealize Automation

1. Enter **cloudadmin** as the username
2. Enter **VMware1!** as the password
3. Click **Sign in**
Navigate to the XaaS Menu

1. Select the **Design** tab
2. Select the **XaaS** menu item on the left

Create a New Custom Resource

Custom Resources allow us to map vRealize Orchestrator objects to a manageable object in vRealize Automation.

Create a new Custom Resource to map the AD:User object type to vRealize Automation

1. Select **Custom Resources** from the menu on the left
2. Click **+New**
Define the Orchestrator Type

1. Type **AD** in the box and select **AD:User** from the selection box that appears
2. Enter **AD User** for the Name
3. (Optional) Add a description to the Description field
4. Click **Next**
**View the Schema**

In the previous step, we selected the AD:User Orchestrator type. This screen shows the schema information from that object.

1. Scroll down on the Form page to view the fields that are available for the AD:User type
2. Click **Finish**
Create an XaaS Blueprint from a Custom Resource Workflow

Now that we have created a custom resource, we need to create a XaaS Blueprint and some resource actions to manage it.

Create a new XaaS Blueprint

Now that a Custom Resource has been created, we can create a new Blueprint to consume that resource.

1. Select the XaaS Blueprints menu item
2. Click the +New button to create a new Blueprint
Navigate to the Active Directory Folder

1. Expand the folder tree by clicking on the arrow next to the folders
The create a user with a password in an organizational unit is the workflow we will use to create users for our developers to test with.

1. Scroll down until you see **User**
2. Expand the **User** folder
3. Select the **Create a user with a password in an organizational unit** workflow
4. Click **Next**
Configure the General Options

The name and description fields are populated from the Orchestrator workflow selected in the previous step. The default values are OK for this run.

1. This Blueprint will be assigned directly to a service. **Uncheck** the option to Make available as a component in the design canvas.
2. Click **Next**
Select the Domain Name Field

The developer accounts need to be created in the corp.local domain, so the domain name field will be modified to read only with corp.local as the value.

1. Scroll down
2. Select The domain name in Win2000 form to edit the field
Make the Field Read Only

1. Select the **Constraints** tab
2. Click the **Read only** drop-down box
3. Select **Constant**
4. Select **Yes** from the value drop-down box
Set the Domain Value

1. Click the **Value** drop-down box
2. Select **Constant**
3. Enter `corp.local`
4. Click on the **Value** drop-down box (#1) again to collapse the drop-down (Not Shown)
Apply the Changes

1. Click **Apply** to save the changes
2. Click **Next**
Select the Custom Resource

The new XaaS Blueprint will provision to the Custom Resource that was defined in previous steps.

1. Select **newUser [AD User]** to map the Blueprint to the AD User Custom Resource
2. Click **Finish**

Publish the Blueprint

The XaaS Blueprints interface is shown with options for creating and managing XaaS blueprints.
1. Select the new Blueprint by clicking on the description
2. Click Publish
Create Resource Actions to Manage the Custom Resource

Now that a Blueprint has been created, Resource Actions need to be created to manage the lifecycle of the Active Directory User.

Create a new Resource Action

1. Select Resource Actions from the menu on the left
2. Click +New

The first resource action will enable a disabled user account.

1. Select **Resource Actions** from the menu on the left
2. Click **+New**
Select the Workflow

1. Expand Orchestrator->Library->Microsoft->Active Directory->User
2. Select Enable a User
3. Note the Input Parameter is a type of AD:User, the same type as the Custom Resource created earlier
4. Click Next
Map the Input Resource

1. The default values have already been set. Note the mapping between the Custom Resource and the Input Parameter for the workflow.
2. Click **Next**
Configure Details

1. The Name is inherited from the workflow. Change this to **Enable this user**
   
   The Enable this user action should only be available if the user is disabled

2. Select **Available based on conditions** for the Target criteria
3. Select the **Clause...** drop-down box
Set the Clause Property

1. Select the **Is enabled** parameter to determine if the account is enabled or not

Enter the Criteria

In this case, the action should only be available if the user is disabled.

1. Click the criteria drop-down box and select **Equals**
2. Click the value and type **false** in the text box
3. Click **Next**
The enable a user workflow only has one input which has been satisfied by the Custom Resource defined earlier. For other workflows, a description or other fields may be necessary and could be entered here.

1. Click Finish (Shown greyed out)
2. Click Yes to acknowledge that there will not be a form for this action

### Publish the Action

The new action must be published so it will be available for use.
1. Click on the description of the new action (You may have to scroll down to find it)
2. Click **Publish**
3. Click **New** to start creating the disable this user action

Select the Workflow

1. Expand **Orchestrator->Library->Microsoft->Active Directory->User**
2. Select **Disable a User**
3. Note the Input Parameter is a type of AD:User, the same type as the Custom Resource created earlier
4. Click **Next**
Map the Input Resource

1. The default values have already been set. Note the mapping between the Custom Resource and the Input Parameter for the workflow.
2. Click **Next**
Configure Details

1. The Name is inherited from the workflow. Change this to **Disable this user**

   The Disable this user action should only be available if the user is disabled

2. Select **Available based on conditions** for the Target criteria

3. Select the **Clause...** drop-down box

Set the Clause Property

1. Select the **Is enabled** parameter to determine if the account is enabled or not
In this case, the action should only be available if the user is disabled.

1. Click the criteria drop-down box and select **Equals**
2. Click the value and type **true** in the text box
3. Click **Next**
Form Options

The disable this user workflow only has one input which has been satisfied by the Custom Resource defined earlier. For other workflows, a description or other fields may be necessary and could be entered here.

1. Click Finish (Shown greyed out)
2. Click Yes to acknowledge that there will not be a form for this action

Publish the Action

The new action must be published so it will be available for use.
1. Click on the description of the new action
2. Click **Publish**
3. Click **+New** to start creating the destroy this user action

### Select the Workflow

1. Expand **Orchestrator->Library->Microsoft->Active Directory->User**
2. Select **Destroy a User**
3. Note the Input Parameter is a type of AD:User, the same type as the Custom Resource created earlier
4. Click **Next**
Map the Input Resource

1. The default values have already been set. Note the mapping between the Custom Resource and the Input Parameter for the workflow.
2. Click **Next**
1. The Name is inherited from the workflow. Change this to **Destroy this user**

   This workflow will delete the user, so the Disposal Type should be selected

2. Select **Disposal** as the Type
3. Click **Next**
Form Options

The enable a user workflow only has one input which has been satisfied by the Custom Resource defined earlier. For other workflows, a description or other fields may be necessary and could be entered here.

1. Click **Finish** (Shown greyed out)
2. Click **Yes** to acknowledge that there will not be a form for this action

**Publish the Action**

The new action must be published so it will be available for use.

1. Click on the description of the new action (You may need to scroll down to find it)
2. Click **Publish**
Entitle the XaaS Blueprint and Actions

Now that a Blueprint and Resource Actions have been created, the Blueprint needs to be entitled.

Navigate to Catalog Management

1. Select the **Administration** tab
2. Click **Catalog Management** from the menu on the left

Navigate to Actions

Before entitling the actions, we should add some icons that identify them better in the interface.
1. Select **Actions** from the menu on the left
2. Click on **Disable this user** to open the properties

**Change the Icon**

![Icon Selection Image]

1. Click **Browse...** to select an icon

**Select the Icon File**

![Icon File Selection Image]

1. Type or browse to `c:\hol\HOL-1921-03-CMP\minus.png` in the File name box.

Hint: You can use the drag and drop feature to copy the text to the console.
2. Click Open
3. Click Finish (Not Shown)

**Modify the Enable This User Action**

![Enable this user action]

1. Select the **Enable this user** action to edit the properties

**Change the Icon**

![Change the Icon]

1. Click **Browse...** to select an icon
Select the Icon File

1. Type c:\hol\HOL-1921-03-CMP\plus.png in the File name box.
2. Click Open
3. Click Finish (Not Shown)
Create a New Service

To make the new Blueprint easier to find, a new service will be created.

1. Select Services from the menu on the left
2. Click +New to create a new service

Configure the New Service

1. Enter User Management in the Name field
2. Ensure Active is selected in the Status drop-down
3. Click Browse... to change the icon
Select the Icon File

1. Enter `c:\hol\HOL-1921-03-CMP\user-group-icon.png` in the box
2. Click **Open**
3. Click **OK** (Not Shown)
Manage the Catalog Items

Now that the new service is created, the XaaS Blueprint needs to be added to the Service

2. Click Manage Catalog Items

Add the XaaS Blueprint to the Service

1. Click the + button
Select the XaaS Blueprint

1. Select the **Create a user with a password...** XaaS Blueprint that was created earlier
2. Click **OK**
3. Click **Close** (Not shown)
Edit the Development Entitlement

With the service created, the Development Entitlement needs to be edited to include the new service.

1. Select **Entitlements** from the menu on the left
2. Click on the **Development** entitlement to edit

Add the User Management Service

1. Select the **Items & Approvals** tab
2. Click the + button
Select the Service

1. Check the box next to **User Management**
2. Click **OK**

Entitle the New Actions
Entitling the service in the previous steps will allow the user access to the service. The actions need to be entitled as well to ensure the developers can manage the user accounts they create.

1. Click on the + button

**Select the New Actions**

![Add Actions Window]

1. Select the three actions you created earlier (You may have to scroll down to find all three, you also maybe more options than shown in the screenshot above)
2. Click **OK**
3. Click **Finish** (Not Shown)
Deploy and Manage the New Custom Resource

Now that the XaaS Blueprint and the associated actions are configured, it is time to deploy and manage a new user account. In this section, we'll log in as a developer and create a user account.

Restore Google Chrome Zoom to 100%

1. Select the 3 vertical dots on the upper right of Google Chrome
2. Select the + symbol until value is at 100%
3. Click on the vRealize Automation Window to return (not shown)

Log Out of vRealize Automation
1. Click **Logout** in the top-right corner

**Go Back to Login Page**

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

**Log in as Devuser**

1. Enter **devuser** as the username
2. Enter **VMware1!** for the password
3. Click **Sign in**
Request the XaaS Blueprint

1. Select the Catalog tab
2. Click on the User Management service that was created earlier
3. Click Request on the Create a user with a password... Blueprint
Configure the Request

This request will create a new user in the Developers OU. Remember that corp.local was set as a read-only field.

1. Type **Dev** in the ouContainer field
2. Select **Developers** from the resulting drop-down (Note: this value may change as when you edit the password. Leave the field as is)
3. Enter **TestUser1** for the account name
4. Enter **VMware1!** in both password fields
5. Enter **Test User** as the display name
6. Click **Submit**
Monitor the request

1. Select the Requests tab
2. Click the Refresh icon on the bottom of the Requests screen until the status until the request shows Successful (refresh icon not shown)

Manage the New User

1. Select the Items tab
2. Click on the description of the user account
3. Click on the Actions drop-down
4. Select the OK button (not shown)

In previous steps, three actions were created, Enable, Disable, and Destroy. In the menu shown there are only two. This is due to the conditional visibility on the Enable and Disable actions.

4. Select Disable this user from the menu
5. Click Submit (Not shown)
Monitor the Request

1. Select the Requests tab
2. Click Refresh Icon on the bottom of the requests screen until the status until the request shows Successful (refresh icon not shown)

Manage the New User

1. Select the Items tab
2. Click on the description of the user account
3. Click on the Actions drop-down
4. Note that Enable this user is now available due to the conditional availability
5. Optional: You can enable this user by selecting the Enable this user action and selecting submit if you wish (Not required)
Conclusion

Congratulations on creating your own custom resource along with day-2 actions to go along with it.

You've finished Module 5

If you are looking for additional information on deploying vRealize Automation, try one of these:

- Or use your smart device to scan the QR Code above

Proceed to any module below which interests you most, or move to the next step to end your lab.

- **(30 minutes)** Explore the limitless flexibility that extensibility brings to vRealize Automation, and learn about different ways you can implement and leverage it to streamline and automate complex processes.
- **(45 minutes)** Discover IPAM integration between VMware NSX, Infoblox NIOS and vRealize Automation. Learn how the On-Demand network in NSX consumes Infoblox IPAM for workloads deployed through vRealize Automation.
- **(45 minutes)** Examine the new ServiceNow integration capabilities in vRealize Automation 7. Learn how to provision and perform day 2 operations on resources in the ServiceNow portal.
- **(45 minutes)** Learn how vRealize Automation 7.4’s Configuration Management Framework integrates with Puppet to install and configure software on a newly provisioned machine.
- **(45 minutes)** Learn how to extend and customize existing virtual machine blueprints to leverage the new Custom Forms Designer introduced in vRealize Automation 7.4. Make your request forms more dynamic and responsive with ClarityUI from VMware. Covering topics such as retrieving external values with
vRealize Orchestrator, Customizing Look and Feel with CSS, and external validation.

How to End Lab

To end your lab click on the END button.
Module 6 - Modifying Blueprint Request Forms with Custom Forms Designer (60 Minutes)
Introduction

Custom Forms Designer is a powerful new feature introduced in vRealize Automation 7.4 used to extend and enhance your existing blueprints with a dynamic and feature rich form. Custom Forms Designer enhances existing functionality such as external values and field validation, as well as introduces new functionality through Cascading Style Sheet (CSS) importing, form generation, and regex validation.

This HOL-1921-03-CMP lab is broken up into 5 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 you have 90 minutes available per session.

If you have not already reviewed it, the 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 the cloud admin for Rainpole to deploy Custom Forms to existing blueprint configurations, enhancing both their visual appeal and core functionality

You will need about 45 minutes to complete this module.
Enabling Custom Forms on a Blueprint

Custom Forms Designer is a highly anticipated new feature in vRealize Automation 7.4 which enables end users the ability to customize blueprint request forms without leveraging the Anything-as-a-Service (XaaS) platform.

In order to begin designing and consuming Custom Forms Blueprints, we first need to Activate the capability on a per-blueprint basis.

Open Chrome Browser from Windows Quick Launch Task Bar

1. Click on the Chrome icon in the task bar to open the browser

Zoom Out Google Chrome (Optional)
1. Select the 3 vertical dots on the top right of Google Chrome
2. Reduce zoom to 80% to allow full view of Catalog

**Sign In to vRealize Automation**

![Sign in to vRealize Automation](image)

1. Enter **cloudadmin** as the username
2. Enter **VMware1!** as the password
3. Click **Sign in**

**Select CentOS Blueprint**

![Select CentOS Blueprint](image)
1. Select the **Design** Tab
2. Select the **Text Description** field next to CentOS
3. Select **Custom Forms** Drop Down
4. Select **Edit**

**Review Custom Forms Design Canvas**

The **Design Canvas** has many elements that we can use to craft our request forms

1. **Blueprint Elements** represent custom properties and components that are a part of the traditional blueprint request. These might be existing form drop downs you have created or core components such as CPU or Memory
2. **Generic Elements** represent custom fields you can place directly onto the form. This might be a Text heading for a specific part of the form, an override password, or a text area to give instructions on how to consume parts of your blueprint
3. When you place elements on the design canvas, you are able to see a preview of how they will display and function within the form. In the example above we have placed the **Lease days** field onto the canvas
4. When selecting an element, it’s configuration options will display on the right
5. **Edit Blueprint** will allow us to configure the core Blueprint
6. Selecting **Activate Custom Form** will make the Custom Form you created the default request form for this blueprint moving forward
Add Image to Custom Form

1. Drag **Generic Elements** Window up to expose more elements (optional)
2. Drag **Image Element** to the Design Canvas

Select the Image Icon

1. Left Click the **Image** element
Edit Image Default Value

1. Select the **Values** tab
2. Enter http://util-01a.corp.local/centos-logo.png for Default Value

Drag Checkbox Element to Canvas

1. Drag **Checkbox** element onto Canvas

Select Checkbox Element
1. Click on **Checkbox** Element

**Edit Checkbox Appearance**

1. Ensure **Appearance** tab is selected
2. Enter **Show Image?** for Label

**Set Default Checkbox Value**

1. Select the **Values** tab
2. Select the **Default value** drop down
3. Set value to **Yes**
Select Our Image Element

1. Select the **CentOS Image** on the Canvas

Set Conditional Value on Image Visibility

1. Select the **Arrow** to the **left** of **Visibility** to expose the configuration options
2. Set Value source to **Conditional value**
Create Yes Expression for Image Visibility

1. **Click** and **drag** side pane **left** if needed to render text
2. Select **Yes** for **Set Value**
3. Select **Show Image**
4. Select **Equals**
5. Select **Yes**

Add Another Expression

1. Select **Add Expression**

Create No Expression for Image Visibility

1. **Click** and **drag** side pane **left** if needed to render text
2. Expression will be displayed after the previous step where you can configure our expression. Select **No** for the value in the drop down
3. Select the **Image Element** from the list. In this step we are referencing the existing "Show Image?" field we created. **Note:** Due to resolution constraints in the environment you may not see the full text.

4. Select the **Equals** option

5. Set value expression to **No**

**Add Lease days Blueprint Element**

1. Expand the **General** Tab in Blueprint Elements
2. Drag **Lease days** onto the canvas
Add CPU and Memory (MB) to Canvas

1. Expand the CentOS drop down
2. Drag CPUs onto the Design Canvas
3. Drag Memory (MB) onto the Design Canvas

Add DropDownList Element to Canvas

1. Scroll Generic Elements menu down
2. Drag DropDownList element onto Canvas, above Lease days
3. Select DropDownList Element
Configuring DropDown Menu Appearance

1. Drag **properties** window up to expand view
2. Scroll Down as needed to show properties
3. Enter **Deployment Type** as the DropDown Label
4. Expand **Custom help** Field
5. Enter **Web, Application, or Database** into Signpost help field
Populate DropDown Values

1. Select the **Values** tab
2. Expand the **Value options** section
3. Enter `app|Application, web|Website, db|Database`

Activate the Custom Form

Click on the switch next to **Activate custom form** on the top right hand corner of the **Custom Forms Designer** to enable the form
Implementing Blueprint Constraints

Constraints allow us to restrict the values of Elements to minimum/maximum/expected value. Constraints can be calculated as constants, calculated conditionally, or using external sources (vRealize Orchestrator). Currently, our Blueprint has minimum and maximum values set at a blueprint level which will restrict our ability to set constraints within Custom Forms Designer. In order for us to progress, we will need to modify these values on our CentOS Blueprint directly.

Edit CentOS Blueprint

1. Click on **Edit Blueprint**
2. Select **Yes** when prompted to Save Changes to the Form

Adjust CPU and Memory Minimum and Maximum

1. Select our **CentOS** Component
2. Select the **Machine Resources** Tab
3. Modify the **Maximum** CPU to **4**
4. Modify the **Maximum** Memory (MB) to **4096**
Edit Custom Form

1. Select **Edit Custom Form**
2. Select **Yes** when prompted to save changes to your blueprint (not shown)

Set Deployment Type Constraints

1. Select **Deployment Type** field
2. Select **Constraints** Tab
3. Select **Yes** on Required
Set Lease Constraints

1. Select the Lease days field
2. Select the Constraints tab
3. Set Required to Yes
4. Set Minimum value to 1
5. Set Maximum value to 4

Set Lease days Default Value

1. Select Values tab
2. Set Default value to 1
Configure CPU Constraints

As we modified our base Blueprint, we need to update the constraints tab on our Custom Form to reflect the new Maximum Value of 4 CPU

1. Select the **CPUs** Element
2. Select the **Constraints** Tab
3. Set **Required** to **Yes**
4. Set **Maximum value** to **4**
Configure Memory Constraints

1. Select the **Memory (MB)** Element
2. Select the **Constraints** Tab
3. Set **Required** to **Yes**
4. Set **Maximum value** to 4096
Working with External Sources

Change Deployment Type Value to External Source

1. Select our **Deployment Type** Drop Down Element from our Design Canvas
2. Select the **Values** tab
3. Expand **Value options**
4. Set **Value source** to **External Source**
5. Search for **returnAppList** by typing **returnAppList** in the Select Action field, select to accept

You can observe the code to return the drop down list from within vRealize Orchestrator below. Orchestrator returns these values as a **Properties** object

```javascript
var dropDownProps = new Properties();
    pickerProperties.put("app","Application");
    pickerProperties.put("web","Web");
    pickerProperties.put("db","Database");
return dropDownProps;
```
Drag a New Text Element Onto the Design Canvas

1. Move **Generic Elements** window up if needed (optional)
2. Drag **Text Element** onto Design Canvas
3. Select **Text Element**

Edit Text Values
1. Select the **Values Tab**
2. Expand **Default value**
3. Change **Value Source** to **External source**
4. Search for the **returnHeaderText** action
5. Set the app input to a **Field**
6. Select **Deployment Type** for binding field

You can observe the code to return the header text from vRealize Orchestrator below. Orchestrator returns these values as a string object in this case. As you can see, the app variable comes from our Deployment Type drop down. When the drop down item is selected, the action is evaluated and the appropriate text is returned.

```javascript
if (app === null){
    return "I have no idea what you are deploying. Select an Application Below!"
} else if (app == "web"){
    return "Looks like you're building a website today! @vaficionado prefers Frontpage"
} else if (app == "app"){
    return "Application server huh? Sounds like fun!"
} else if (app == "db"){
    return "Databases are cool. Watch that log size though!"
}
```
Implementing External Validations

External Validations extend the External Sources capability to encompass checking multiple fields for validity before submitting a catalog request. Examples of this might be checking cluster capacity, checking Active Directory for a hostname prior to creation, checking DNS for an existing record, or checking IPAM for an already reserved IP Address.

In our example, we will leverage a simple External Validation that compares the Lease days with the requested Memory (GB) value to determine if a request is appropriate.

Select External Validation Tab

1. Once in the Design Canvas, select the External Validations tab
2. Select Orchestrator Validation and drag onto the Design Canvas
3. Left click on the Orchestrator validation Element

Configure External Validation
Select and Edit External Validation

1. Set Validation Label to Lease to Memory Validation
2. Enter `validateLeaseWithSize` into the search box, select entry when found (not shown)
3. Map the **Action input lease** field to the Lease days input from our Custom Form
4. Map the **Action input memoryMB** field to the Memory (MB) field from our Custom Form
5. Add the **Memory (MB)** field as the Highlighted field for validation

Save our Custom Form

1. Select **Save** on the bottom right of the screen
Launch a New Browser Window

1. On the top right of Google Chrome, select "New incognito window"

Access vRealize Automation

1. Select vRealize Automation bookmark from Google Chrome
Select corp.local domain

1. Select Next

Log In as devuser

1. Enter devuser as the username
2. Enter VMware1! as the password

Log in as a developer to request the new Blueprint
3. Click **Sign in**

**Request CentOS Blueprint**

![Diagram showing the process of selecting the Catalog and requesting a CentOS Blueprint]

1. Select the **Catalog** navigation item
2. Select **Request** on our **CentOS Blueprint**
Populate and Submit CentOS Custom Form

1. Choose Anything in Deployment Type
2. Enter 8 for lease days
3. Enter 1 for CPUs
4. Enter 2048 for Memory (MB)
5. Select Submit on the bottom right (not shown)
Observe Validation Error

Databases are cool. Watch that log size though!

Virtual Machine lease cannot be greater than 5 days when memory requested is more than 1024 megabytes. Please reduce lease or memory and resubmit.

Log Out of devuser

1. Select **Logout**
2. Close this browser window as it is no longer needed (not shown)
Extending Custom Form Design with Cascading Style Sheets (CSS)

Another major feature of Custom Forms Designer is the ability to augment any of the fields and web structure of the Custom Form with Cascading Style Sheets (CSS). Each Field and Element are provided a Field ID that acts as a CSS selector. Additionally, many of the standard CSS class selectors are able to be called as well.

In this simple CSS example, we will modify the standard font for the form, as well as change the default font size. We will gather the Field ID for our Header text, as well as the descriptive text, and modify the text style for these. Let's get started.

Select CentOS Blueprint

1. Select the Design Tab (if not selected already)
2. Select the Text Description field next to CentOS
3. Select Custom Forms Drop Down
4. Select Edit
Identify Header Text Field ID

1. Select the **Blueprint Request Parameters** Text Element we created earlier
2. Take note of the **Field ID** generated. **NOTE: Yours will likely be different, do not use the ID in this screenshot**

Launch File Explorer

1. Left-Click Explorer Icon in taskbar
Browse to the CSS File

1. Navigate to C:\ho\HOL-1921-03-CMP
2. Double click on 1921-03-css.css to launch the CSS file

Edit CSS Template

1. Replace highlighted text the Field ID we noted above. **NOTE: You must Include a # symbol before the field ID, for example #text_5b50dc73**
2. **Save and Exit** the file (Not Shown)
Apply the CSS Template

1. In the Custom Forms Designer select the **Actions** Menu
2. Select **Import CSS**
Browse to Template File

1. Browse to CSS location - C:\hol\HOL-1921-03-CMP
2. Open CSS file at this location - 1921-03-cs.css

Save our Custom Form

1. Select Save on the bottom right of the screen
Launch a New Browser Window

1. On the top right of Google Chrome, select "New incognito window"

Launch vRealize Automation
Log In as devuser

Log in as a developer to request the new Blueprint

1. Enter **devuser** as the username
2. Enter **VMware1!** as the password
3. Click **Sign in**
Request Our Catalog Item

1. Catalog

2. Request

Service Catalog
Browse the catalog for services you need.

Infrastructure (3)

CentOS
Deploys a basic CentOS IaaS Blueprint

Request
1. Select the **Catalog** navigation item
2. Select **Request** on our **CentOS Blueprint**
3. Step not shown - We do not need to submit the request as we are observing the look of the blueprint.

![CentOS Blueprint Image]

**Log Out of devuser**

1. Click **Logout**
2. Close this browser window (not shown)
Restoring Our Configuration Changes

Deactivate Custom Form

1. Select the **Design** Tab. **Note:** If you are still within the blueprint interface from the previous steps. You should click finish to exit.
2. Select the **Description** for the CentOS Blueprint
3. Select the **Custom Form** drop down
4. Select **Deactivate** if you wish to reuse the form at a later date or **Delete** if you wish to clear the form completely. **Note:** Deleting a form cannot be undone.
As part of our Conditions and Constraints configurations, we increased our Maximums for CPU and Memory. We will walk through restoring those to their default values now.

1. Select the **CentOS** Blueprint
Modify Values Back to Default

1. Select our CentOS Machine Object in the Blueprint Canvas
2. Select Machine Resources
3. Set Maximum CPUs to 2
4. Set Maximum Memory (MB) to 1024
5. Select Finish
Conclusion

In this module we updated an existing CentOS blueprint with Custom Forms functionality. We experimented with implementing Generic Elements to provide additional informative context to our form. We implemented Conditional Expressions to automatically populate values. We implemented drop downs to help with our Conditional Expressions and eventually moved those Expressions to External Actions. We implemented External Validations to ensure our Custom Forms request was valid. Finally, we implemented Cascading Style Sheets (CSS) to change the look of our form even further.

You've Finished Module 6

If you are looking for additional information on deploying vRealize Automation, try one of these:

- Visit the vRealize Automation Documentation Center [http://bit.ly/2m3IfIy]
- Or use your smart device to scan the QR Code above

Proceed to any module below which interests you most, or move to the next step to end your lab.

- **Module 1 - Introduction to Extensibility** (30 minutes) Explore the limitless flexibility that extensibility brings to vRealize Automation, and learn about different ways you can implement and leverage it to streamline and automate complex processes.
- **Module 2 - Infoblox IPAM Integration with vRealize Automation** (45 minutes) Discover IPAM integration between VMware NSX, Infoblox NIOS and vRealize Automation. Learn how the On-Demand network in NSX consumes Infoblox IPAM for workloads deployed through vRealize Automation.
- **Module 3 - Integrating vRealize Automation with ServiceNow** (45 minutes) Examine the new ServiceNow integration capabilities in vRealize Automation 7. Learn how to provision and perform day 2 operations on resources in the ServiceNow portal.
• **Module 4 - Puppet Integration with the Configuration Management Framework** (45 minutes) Learn how vRealize Automation 7.4’s Configuration Management Framework integrates with Puppet to install and configure software on a newly provisioned machine.

• **Module 5 - Creating and Managing Custom Resources with vRealize Automation** (45 minutes) Learn how to use vRealize Orchestrator Custom Resources to extend vRealize Automation's ability to manage existing objects throughout your business. From user accounts to coffee makers, Custom Resources are a powerful way to bring automation to all aspects of your business.

**How to End Lab**

To end your lab click 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-1921-03-CMP

Version: 20190914-211918