



HOL-2501-11-VCF-L  
Getting More Out of It!

## Table of contents

|  |     |
|--|-----|
| Lab Overview - HOL-2501-11-VCF-L - VCF Automation Orchestrator - Getting More Out of It! | 4   |
| Lab Guidance .....   | 4   |
| Lab Description.....   | 5   |
| Module 1 - Aria Automation Orchestrator Overview (15 minutes) Basic                      | 6   |
| Introduction .....   | 6   |
| Log in to Aria Automation Orchestrator .....   | 6   |
| Introduction to the Aria Automation Orchestrator Client .....                            | 9   |
| Running our first workflow .....   | 15  |
| Log Out of Aria Automation Orchestrator .....  | 27  |
| Conclusion.....  | 28  |
| Module 2 - Create a Basic Orchestrator Workflow (30 min) Basic                           | 30  |
| Introduction.....  | 30  |
| Log in to Aria Automation Orchestrator .....   | 30  |
| Create a custom workflow .....   | 33  |
| Understand a Workflow Token.....   | 51  |
| Log Out of Aria Automation Orchestrator .....  | 54  |
| Conclusion.....  | 55  |
| Module 3 -Understand Parameters, Attributes and Scripting Objects (30 minutes)           |     |
| Basic  | 57  |
| Introduction.....  | 57  |
| Log in to Aria Automation Orchestrator .....   | 57  |
| Understanding Input and Output Parameters.....   | 61  |
| Understanding Workflow Variables .....   | 64  |
| Using Scripting Objects .....  | 81  |
| Managing Actions .....   | 95  |
| Log Out of Aria Automation Orchestrator .....  | 152 |
| Conclusion.....  | 153 |
| Module 4 - Leverage Existing Scripts in Powershell and Python (45 min)                   |     |
| Advanced   | 155 |
| Introduction.....  | 155 |
| Log in to Aria Automation Orchestrator .....   | 157 |
| Create a simple PowerCLI script .....  | 161 |

|   |     |
|---|-----|
| Leverage Python to integrate with NSX .....   | 176 |
| Log Out of Aria Automation Orchestrator .....   | 217 |
| Conclusion.....   | 218 |
| Module 5 - Integrate Aria Orchestrator and Aria Operations (45 minutes)                                     |     |
| Intermediate  | 220 |
| Introduction .....  | 220 |
| Log in to Aria Automation.....  | 220 |
| Deploy a virtual machine .....  | 223 |
| Review Aria Orchestrator Workflow .....   | 232 |
| Configure Aria Operations for Orchestrator Management Pack .....  | 236 |
| Configure Aria Operations Monitoring Policy .....   | 247 |
| Trigger High CPU Alarm .....  | 271 |
| Log Out of Aria Automation Orchestrator .....   | 276 |
| Conclusion.....   | 277 |
| Module 6 - Enhance Lifecycle Management using Extensibility in Aria Automation<br>(30 minutes) Intermediate | 279 |
| Introduction.....   | 279 |
| Log in to Aria Automation as holadmin.....  | 279 |
| Custom Day 2 Action utilizing an Aria Automation Orchestrator<br>Workflow.....                              | 283 |
| Log Out of Aria Automation Orchestrator .....   | 305 |
| Conclusion.....   | 306 |
| Appendix  | 308 |
| Hands-on Labs Interface (Windows Main Console).....   | 308 |
| Hands-on Labs Interface (Ubuntu Main Console) .....   | 312 |

## Lab Overview - HOL-2501-11-VCF-L - VCF Automation Orchestrator - Getting More Out of It!

### Lab Guidance

[2]

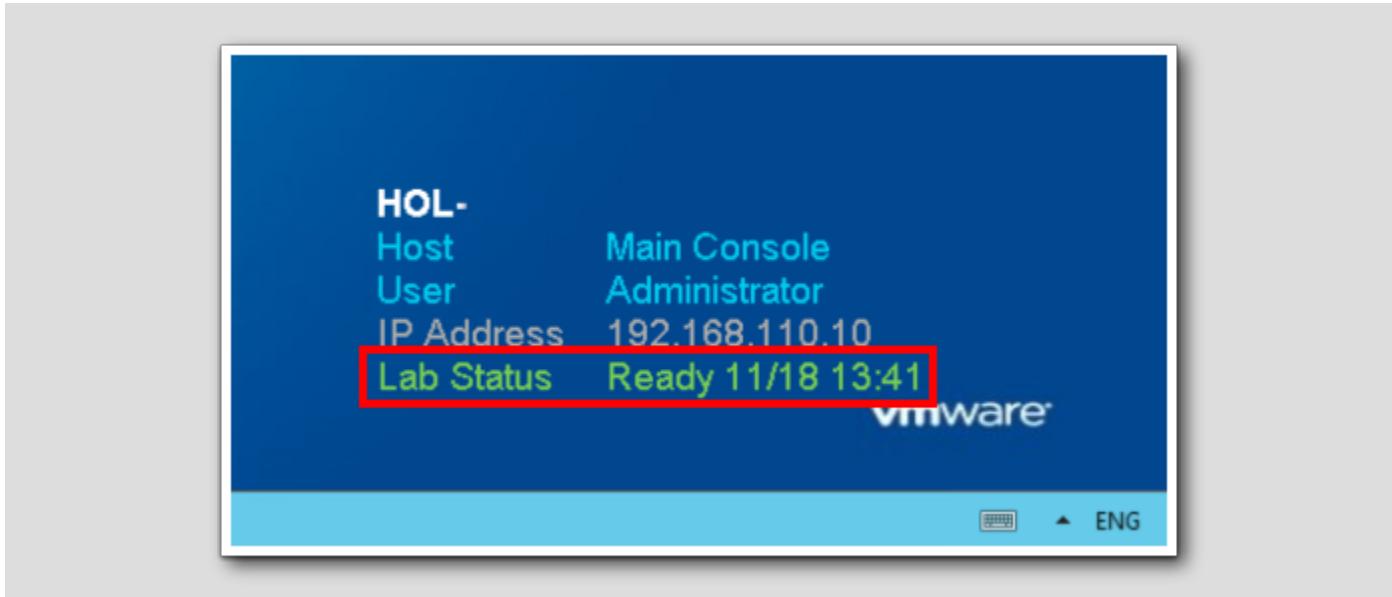
Welcome! This lab is available for you to repeat as many times as you want. To start somewhere other than the beginning, use the Table of Contents in the upper right-hand corner of the Lab Manual or click on one of the modules below.

- [Module 1 - Aria Automation Orchestrator Overview](#) (15 minutes) (Basic) Get an overview of the Aria Automation Orchestrator client and how to run your first workflow.
- [Module 2 - Create a Basic Orchestrator Workflow](#) (30 minutes) (Basic) Build your knowledge of Aria Automation Orchestrator and create your first workflow to perform a simple operation within vCenter.
- [Module 3 - Understand Parameters, Attributes and Scripting Objects](#) (30 minutes) (Basic) Take a deeper look at the key components used by Aria Automation Orchestrator as part of workflows and actions. See how we can manipulate data and include custom code within workflows.
- [Module 4 - Leverage Existing Scripts in PowerShell and Python](#) (45 minutes) (Advanced) Aria Automation Orchestrator supports multiple scripting languages. In this module we will look at how we can bring existing scripts in PowerShell and Python into workflows to reuse the code.
- [Module 5 - Integrate Aria Orchestrator and Aria Operations](#) (45 minutes) (Intermediate) Use the power of Aria Automation Orchestrator to automatically remediate a virtual machine in response to an alert generated in Aria Operations.
- [Module 6 - Enhance Lifecycle Management using Extensibility in Aria Automation](#) (30 minutes) (Intermediate) Extend the lifecycle of a virtual machine by creating custom day 2 actions in Aria Automation using Aria Automation Orchestrator workflows.

### Lab Captains:

- Module 1 - Katherine Skilling, Senior Architect, United Kingdom
- Module 2 - Katherine Skilling, Senior Architect, United Kingdom
- Module 3 - Katherine Skilling, Senior Architect, United Kingdom
- Module 4 - Katherine Skilling, Senior Architect, United Kingdom
- Module 5 - Scott Bowe, Solutions Architect, USA
- Module 6 - Fred Hofer, Staff Technical Adoption Manager, Austria

You are ready....is your lab?



The lab console will indicate when your lab has finished all the startup routines and is ready for you to start. If you see anything other than "Ready", please wait for the status to update. If after 5 minutes your lab has not changed to "Ready", please ask for assistance.

## Lab Description

Run and create workflows to simplify IT tasks. Understand the parameters, attributes, and scripting objects that are used to extend the functionality of workflows and leverage existing scripts.

## Module 1 - Aria Automation Orchestrator Overview (15 minutes) Basic

### Introduction

[6]

In this module, the Aria Automation Orchestrator client interface will be explained. As part of the walkthrough, a workflow will be executed to perform a basic operation on vSphere.

#### Lab Captain(s):

- Katherine Skilling, Senior Architect, United Kingdom

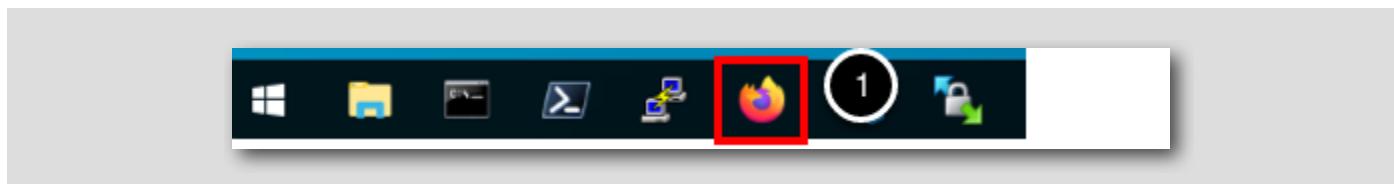
### Log in to Aria Automation Orchestrator

[7]

In the following few pages, we will log in to Aria Automation Orchestrator.

### Open the Firefox Browser from Windows Quick Launch Task Bar

[8]

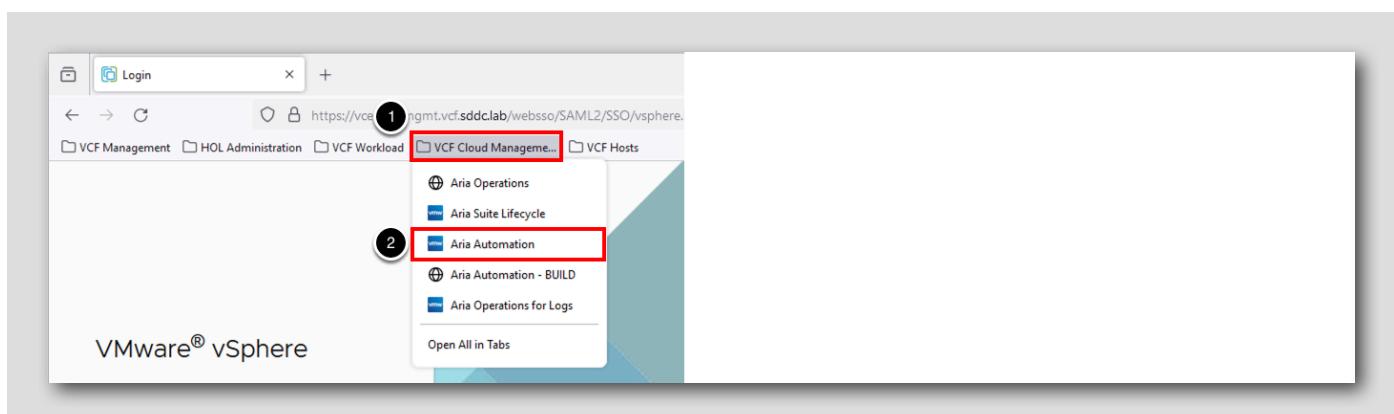


If the browser is not already open, launch Firefox.

1. Click the Firefox icon on the Windows Quick Launch Task Bar.

### Log in to Aria Automation

[9]

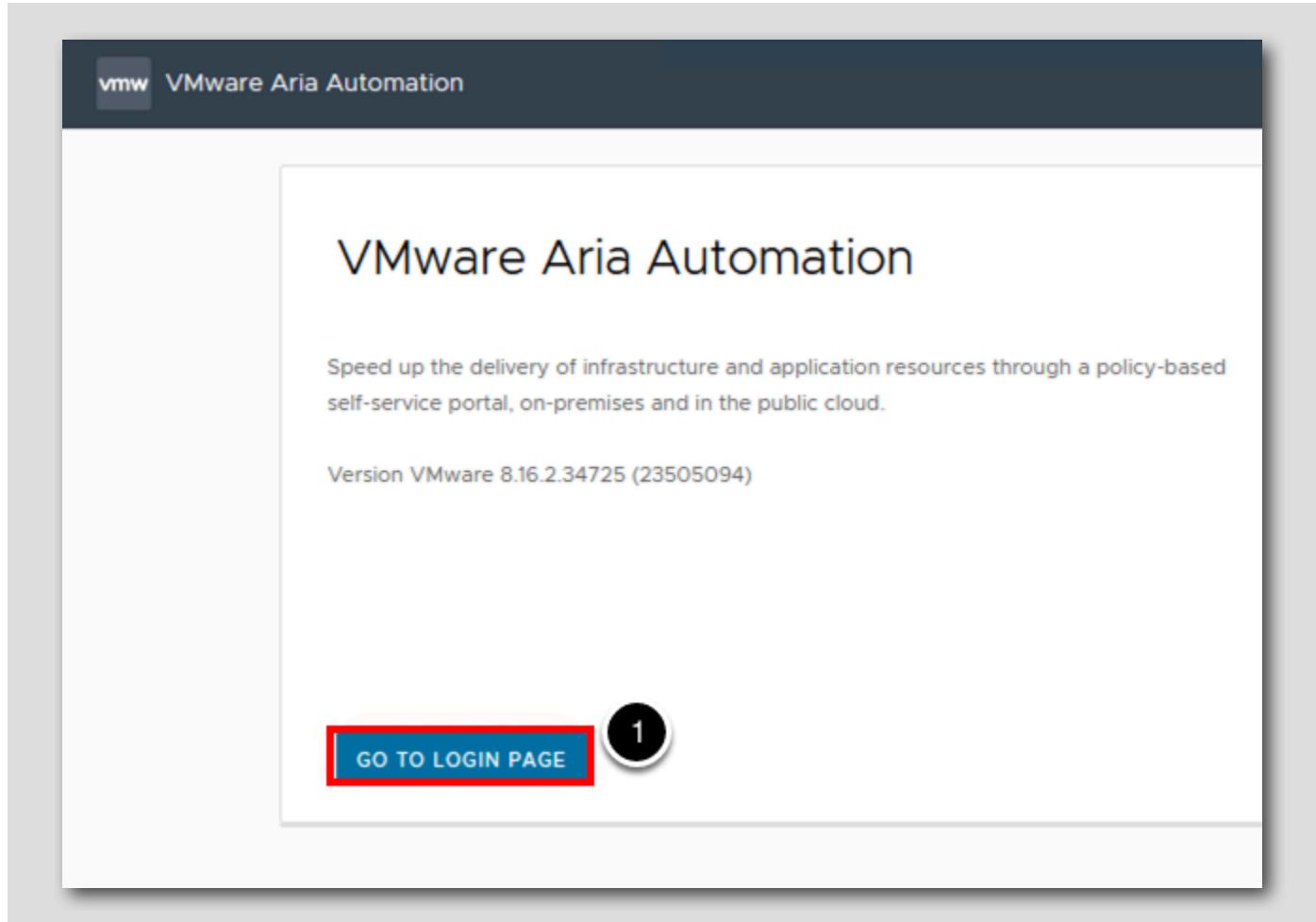


Once Firefox has loaded:

1. Click the VCF Cloud Management bookmark folder
2. Click Aria Automation.

Redirect to Workspace ONE Access for Sign-On

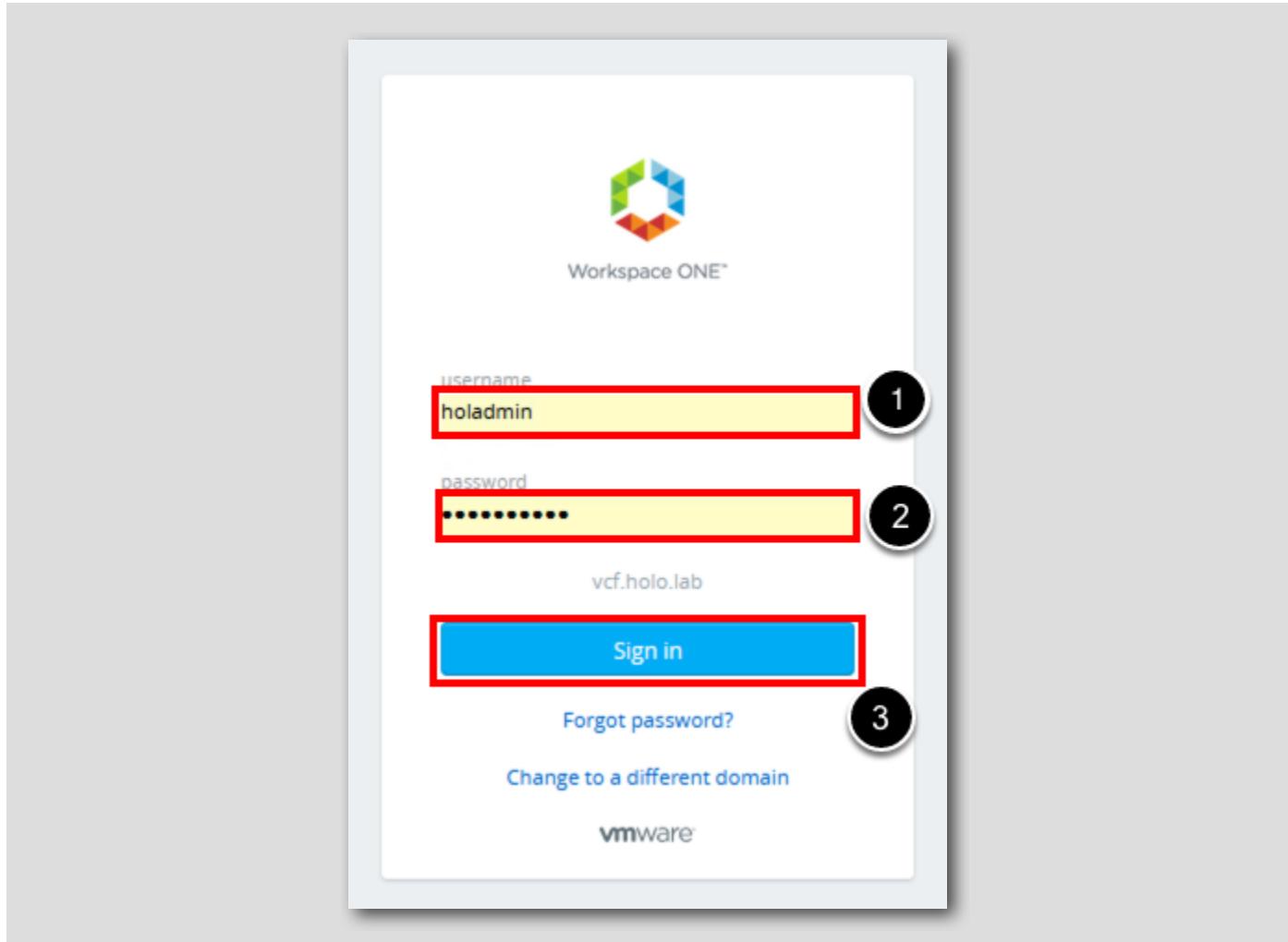
[10]



Aria Automation is integrated with Workspace ONE Access (aka VMware Identity Manager) and we need to redirect to the Workspace ONE Access login page to complete our log in progress.

1. At the VMware Aria Automation page, click GO TO LOGIN PAGE.

## Workspace ONE Access Login

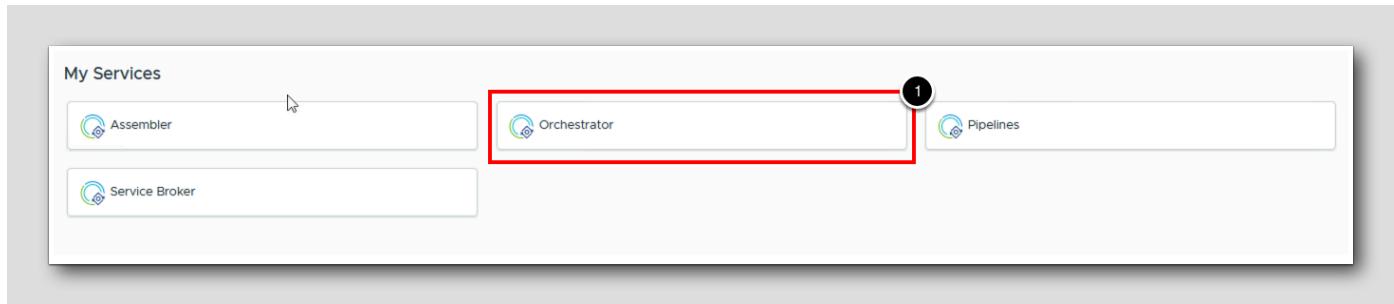


Note: The credentials for **holadmin** should already be cached in the browser window.

At the **Workspace ONE Access** prompt, type in the following user and password information.

1. At the **username** field, type **holadmin**.
2. At the **password** field, type **VMware123!**.
3. Click **Sign in**.

## Launch the Orchestrator Service



From within the Cloud Services Console, under **My Services**:

1. Click the Orchestrator service.

## Introduction to the Aria Automation Orchestrator Client

The Aria Automation Orchestrator Client is the main interface to develop, see and run workflows. In this exercise, we will familiarize ourselves with the key features of the Aria Automation Orchestrator Client user interface.

## Dashboard view - System

The screenshot shows the VMware Aria Automation Orchestrator dashboard with the 'System' tab selected. The left sidebar contains navigation links for Dashboard, Library, Workflows, Actions, Policies, Activity, Assets, and Administration. The main area displays a table of nodes, a pie chart of thread usage, and various resource utilization metrics.

| Number | Node ID                              | State   | Uptime | Load    |
|--------|--------------------------------------|---------|--------|---------|
| 1      | 23fd6e65-9396-429c-b925-63e376ccb937 | Running | 2 d    | 12.34 % |

**Threads Info:**

- Daemon number of threads: 50
- Number of threads: 98
- Peak number of threads: 158

**Heap Memory:**

- USED: 37% (0.79 GB / 2 GB capacity)
- COMMITTED: 100% (2.13 GB / 2 GB capacity)

**Non Heap Memory:**

- USED: 32% (0.35 GB / 1 GB capacity)
- COMMITTED: 32% (0.36 GB / 1 GB capacity)

**File System Usage:**

- overlay: 25% (35 GB / 141 GB capacity)
- /dev/sda4: 35% (17 GB / 48 GB capacity)
- /dev/mapper/data\_vg-data: 25%

**Cluster Settings:**

- Heartbeat Interval: 12,000
- Number Of Active Nodes: 1,000
- Failover Interval: 126,000

1. Click the SYSTEM tab at the top

This view gives the current resource utilization of the orchestrator instance. This is useful to get an idea of how the system may be performing, and whether there are any resource constraints.

## Dashboard view - Usage

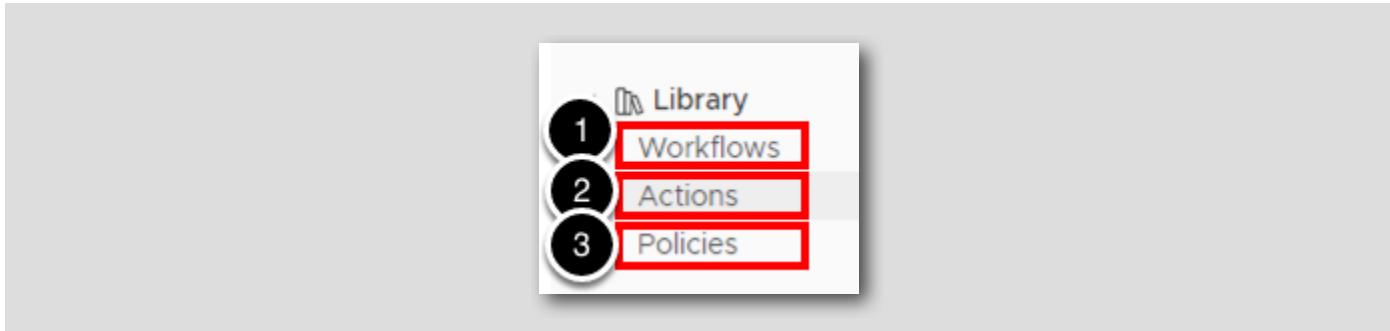
The screenshot shows the VMware Aria Automation Orchestrator Usage dashboard. The left sidebar is titled 'Embedded-VRO' and contains several sections: Dashboard, Library, Workflows, Actions, Policies, Activity (Workflow Runs, Scheduled, Waiting for Input (0), Policy Runs), Assets (Packages, Configurations, Resources, Environments), Administration (Groups, Inventory, Audit Logs, Git Repositories, Git History, Deleted Items), and System. A red box highlights the 'Usage' tab at the top center. Another red box highlights the 'Workflow Runs' section, which shows a green circle indicating 'Completed 100%' and a count of '1 Failed, 1 Completed'. Below this is a section for 'Favorite Workflows (0)' with a note about assigning Favorite tags. To the right are sections for 'Recent Workflow Runs' (listing two completed runs), 'ALL WORKFLOW RUNS' (empty), and 'Requiring Attention' (empty).

1. Click the USAGE tab at the top

This page gives an overview of the usage of the Orchestrator instance. We can see how which workflows have recently been run with their final exit status and any that are currently awaiting user input before they can continue. The Requiring Attention section allows you to review any workflows that have recently failed, are taking a long time to complete, or have the most transition states. These statuses could be an indication that the workflow should be reviewed for errors, or restructured to improve its performance. It is also possible to jump to a specific workflow that has been previously assigned as favorite.

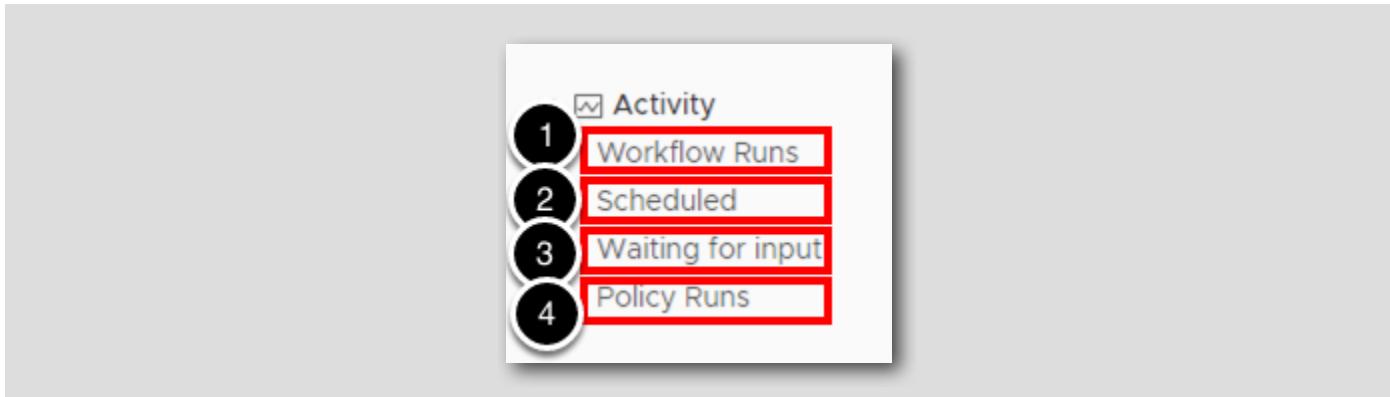
2. On the left, the menu allows us to navigate between the different Aria Orchestrator features and concepts. The next sections will describe each of these items one by one.

## Library



1. **Workflows** - Provides access to the Orchestrator workflow library. We can use the Workflows view to view information about each workflow. We can also use this view to create, edit, run and interact with workflows.
2. **Actions** - Provides access to the libraries of predefined actions. We can use the Actions view to duplicate actions, export them to a file, or move them to a different module in the actions hierarchical list.
3. **Policies** - Displays existing policies. We can use the Policies view to create and apply policies. Policies in Aria Orchestrator are a series of rules, gauges, thresholds, and event filters that run certain workflows or scripts when specific predefined events occur either in Orchestrator or in the platforms that Orchestrator can access through plug-ins.

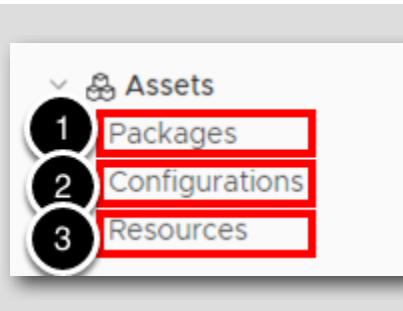
## Activity



1. **Workflow Runs** - Also known as "workflow token", this is the list of all the instances of a workflow that has been executed.  
Each workflow run is independent from each other, has its own logs, can be re-run and can run concurrently with others.
2. **Scheduled** - Displays a list of all scheduled workflows. The workflows are sorted by name or date, together with their status.
3. **Waiting for input** - Displays the list of workflows waiting for required user input in order to finish executing. During execution, some workflows may require additional input from the user (i.e. approval). If this happens, the workflow will be paused while waiting for the information. To make them more easily identifiable amongst all the running workflows, they can be found in this section.
4. **Policy Runs** - Displays all the policies currently running and actively waiting for an event.

## Assets

[18]



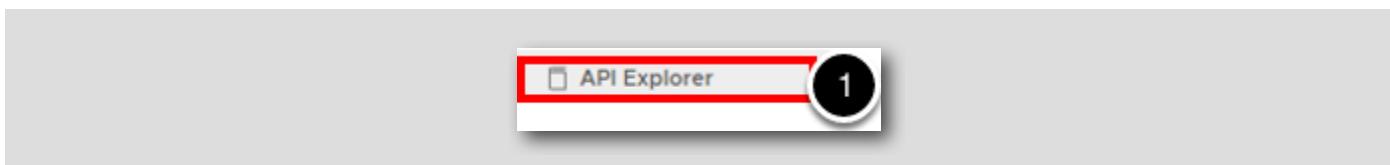
1. **Packages** - Displays a list of the available packages and where a selected package is used. We can use the Packages view to add, import, export, and synchronize packages which can consist of workflows, actions and other Orchestrator assets.
2. **Configurations** - Provides access to the available configuration elements. We can use the Configurations view to create configuration elements to define common attributes across an Orchestrator server. Configuration elements can be used as a reference table of values we want to use within workflows. A single configuration element can hold multiple different attributes and associated values. An example of where a configuration element can be useful is when making REST calls to a remote endpoint within a workflow. We can use a configuration element to store the url, username, password or token and any additional parameters we will need to use for each REST call. We can define these values once and then reference them as many times as needed across different workflows.
3. **Resources** - Provides access to the list of resource elements. We can use the Resources view to import external objects such as images, sysprep files, HTML templates, XML templates, and custom scripts. In turn, we can then use these as resource elements within workflows.

## Administration



1. **Groups** - Allow to create groups of users and assign them to specific elements (like workflows, actions). We can use Groups to create a basic Role Based Access Control (RBAC) configuration to restrict workflow visibility and usage.
2. **Inventory** - Displays the objects of the plug-ins that are enabled in Orchestrator. Within each of the plug-in sections we can see any configured instances. For example under the vCenter plug-in we can see each vCenter instance we have added to our inventory and any objects supported by the plugin within the vCenter inventory such as Virtual Machines.
3. **Audit logs** - Displays an aggregated view of all the workflow events (started, completed, failed, and saved), it also includes output values of workflows. We can use these to track down activity and monitor what's happening in Aria Automation Orchestrator.
4. **Git Repositories** - Allows us to integrate with a git repository for the versioning of the workflows. Once configured we can perform Git Push and Git Pull operations within Orchestrator to sync our content with the repository.
5. **Git History** - Displays a history of the git interactions in terms of the operations performed over time, and allows us to visualize the history of git activity.

## API Explorer



1. API Explorer - Shows the documentation for all of the javascript classes available out-of-the-box in Orchestrator. We can use this documentation to help us understand the interactions (methods) we can perform on an object, and the properties each of the object is assigned.

## Running our first workflow

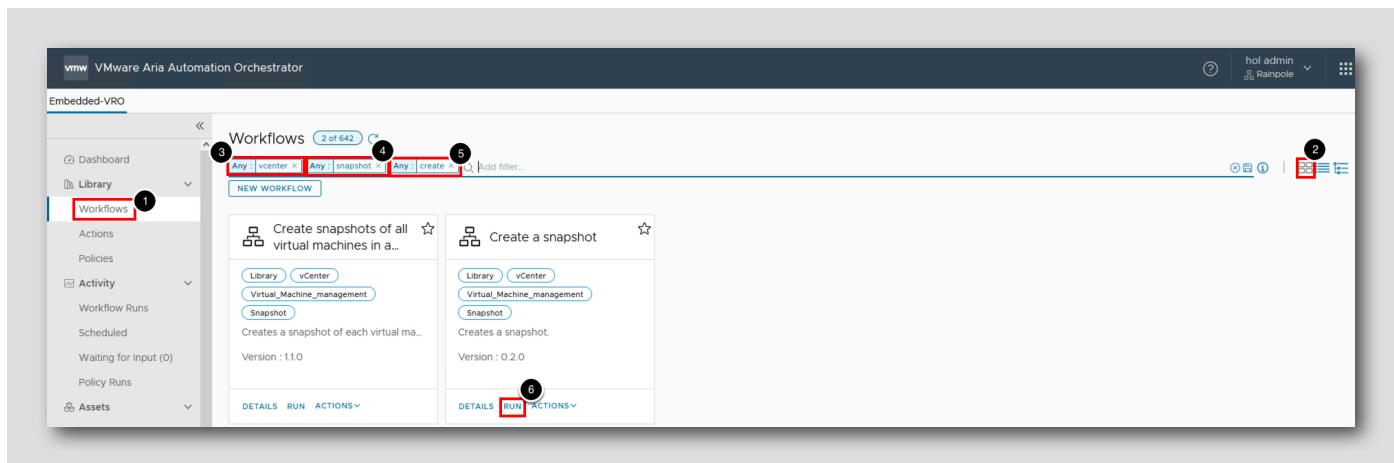
[21]

Now that we have a basic understanding of the key features of the Aria Automation Orchestrator Client, let's use the client to run our first workflow. In this lesson we will use the client to filter the available workflows to locate the workflow we want to run, provide the workflows input values needed at run time and then review the outcome of the workflow run.

We will use a workflow that is provided out of the box with Orchestrator, which will create a snapshot on a Virtual Machine. The workflow will perform the same steps that we as an administrator can perform using the vCenter interface. In later lessons we will look at creating new workflows and editing the code within them, for now we will keep it simple and reuse a workflow without making any changes.

## Search for a Workflow

[22]

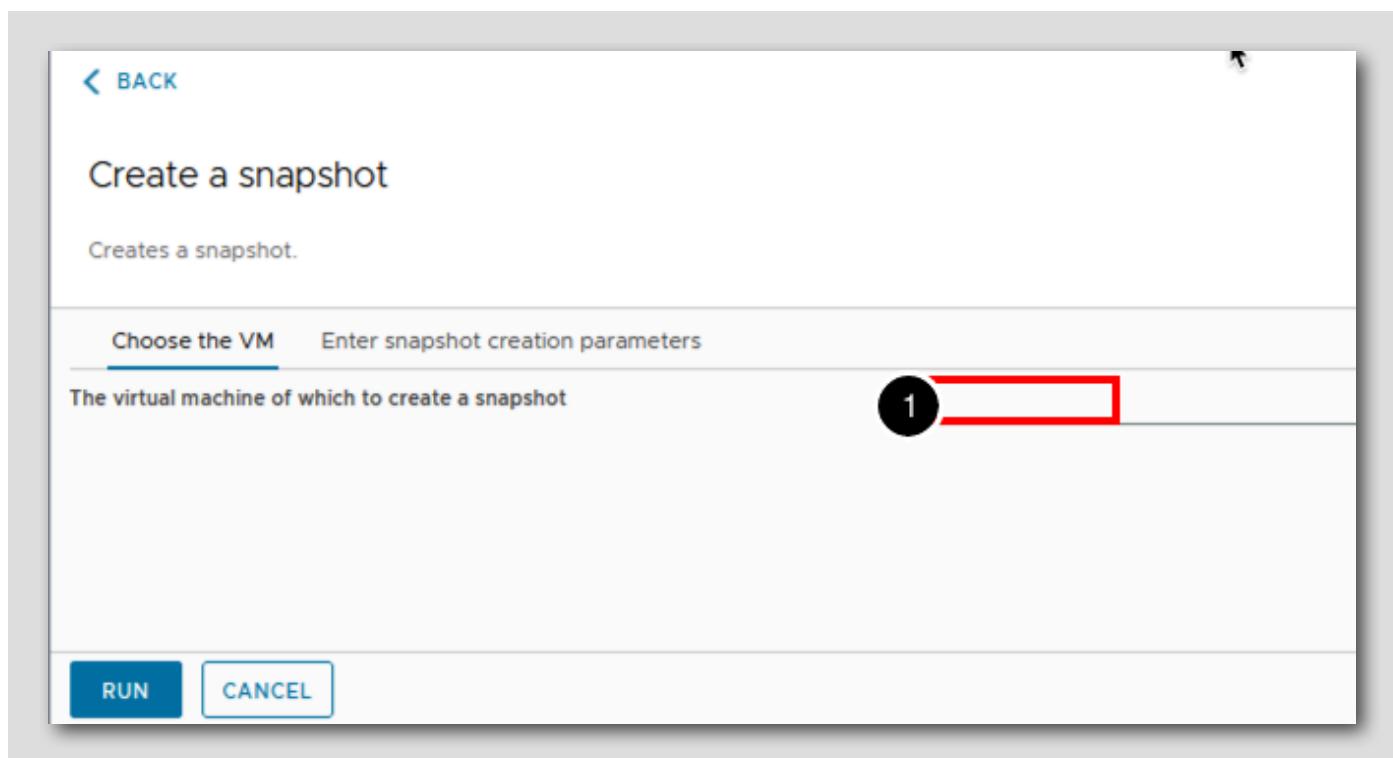


In the 8.16 release of Aria Automation Orchestrator there are over 600 workflows provided with the default installation. While this makes it a very powerful tool, it can also make it difficult to locate the workflows we may need. Let's start by filtering the workflows to find the Create a snapshot workflow instance.

1. Click the **Workflows** tab
2. Make sure the card view is selected, this displays each workflow in its own card in a flat structure within the interface. We will use some of the other view options in later lessons.
3. In the search box, enter **vcenter** and press return. This filters the workflows to any that contain the word vcenter (case insensitive) within their name or tag values.
4. We still have a large number of workflows in the results, let's refine them further by adding more search criteria. Enter **snapshot** in the search bar and press return.
5. Enter **create** and press return
6. Now we have located the workflow we want to use let's start running it. For the workflow **Create a snapshot**, click the **RUN** button.

Select the VM to use as an input

[23]



When the workflow run screen appears we can see that we are prompted to provide some information. These are the workflow inputs. Each workflow has a set of inputs, outputs and attributes assigned to it. We will discuss these elements more in a later lesson, for now we are just going to choose the Virtual Machine on which we want to take a snapshot. This is effectively the steps an administrator performs by logging into vCenter, browsing the inventory and clicking on a Virtual Machine ready to use the menu options to take a snapshot. Orchestrator will perform some of these steps for us, however we still need to direct it to the correct Virtual Machine.

1. Click in the field to browse for the Virtual Machine.

## Choose the VM object

[24]

| id                          | vm-11009                               |
|-----------------------------|--|
| Orchestrator unique ID      | vm-11009                               |
| Annotation                  |  |
| VM Connection State         | connected                              |
| Consumed Host CPU           | 59 MHz                                 |
| guestHeartbeatStatus        | green                                  |
| IP address                  | 10.64.12.10                            |
| Active Guest Memory         | 0 MB                                   |
| CPU                         | 1 vCPUs                                |
| Product Vendor              |  |
| Memory Overhead             |  |
| Guest OS                    | Ubuntu Linux (64-bit)                  |
| dunesId                     | vcenter-mgmt.vcf.sddc.lab.jd:v m-11009 |
| VM Template                 | false                                  |
| sdkId                       | vcenter-mgmt.vcf.sddc.lab/v m-11009    |
| Used Storage                | 13200 MB                               |
| name                        | ubuntu-000308                          |
| VMware Tools Version Status | guestToolsUnmanaged                    |
| VM Version                  | vmx-21                                 |
| alarmActionsEnabled         | true                                   |
| overallStatus               | green                                  |

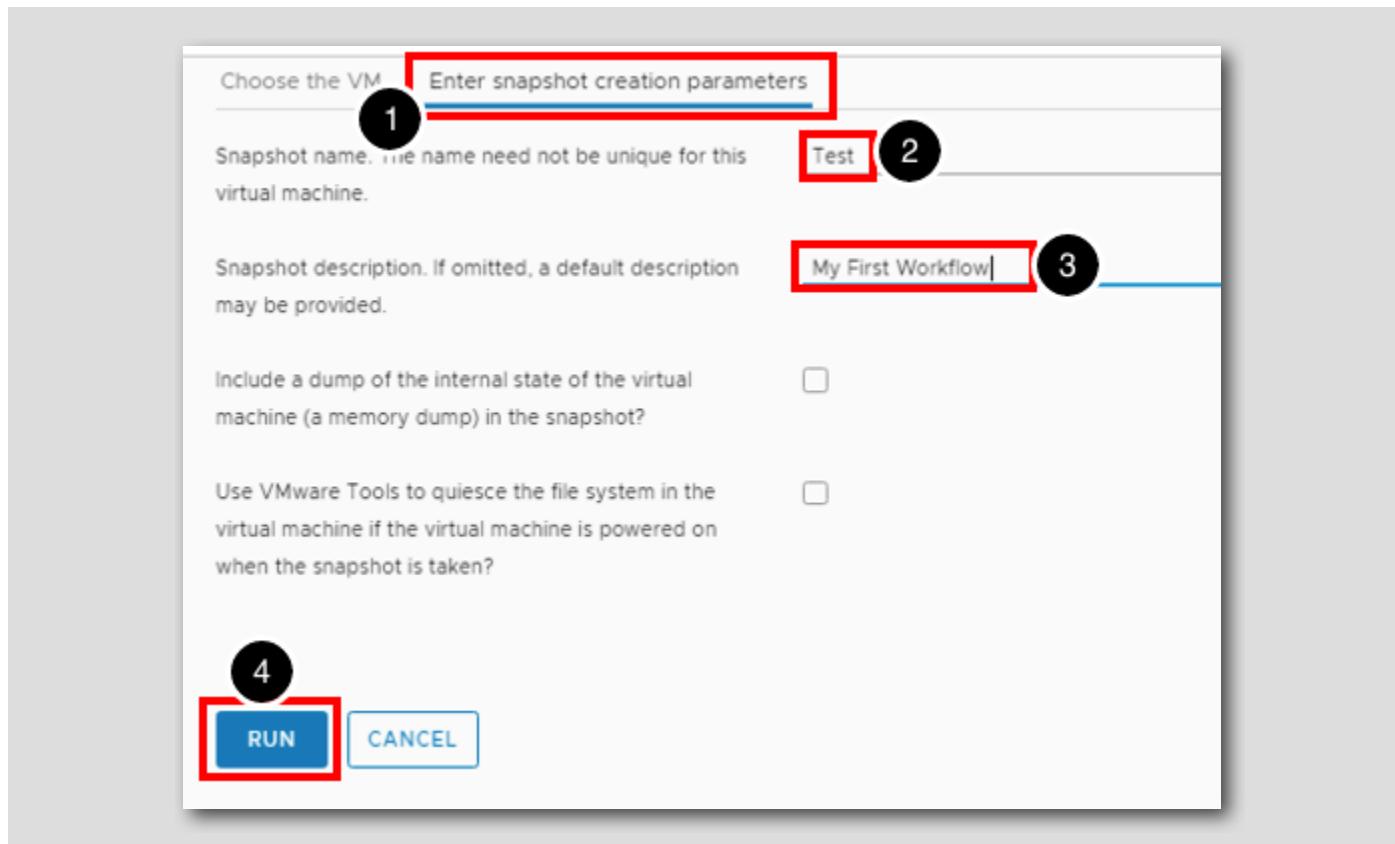
When we clicked into the field we were brought to a screen known as a tree picker by Orchestrator. This is because our input has a type of VC:VirtualMachine, which is the Orchestrator representation of a vCenter Virtual Machine. Orchestrator now prompts us to browse all of the vCenter inventories it has access to, and to select the Virtual Machine we want to assign to our workflow input. As we browse the inventory notice how we can expand sections such as Datacenters and Folders but can only select Virtual Machine inventory entries. This is Orchestrator filtering the inventory based on the input parameter type. It knows it is expecting a Virtual Machine and will prevent any other inventory object type being selected.

1. Browse to vSphere vCenter Plug-in > <https://vcenter-mgmt.vcf.sddc.lab:443/sdk> > Datacenters > mgmt-datacenter-01 > vm >Workloads> ubuntu-000308 and select the Virtual Machine by clicking in the box to the left of the name.
2. Click the SELECT button to confirm the choice of Virtual Machine.

The name of the VM within your lab may be different from the screenshot as the VM is named automatically during deployment from Aria Automation. Select any VM with the name in the format ubuntu-xxxxxx e.g. ubuntu-000308.

## Enter the snapshot creation parameters

[25]



With the Virtual Machine selected we have a few more input parameter values to consider. Again, these are aligned to the options an Administrator would be presented with if they were performing the operation via vCenter manually. We need to tell Orchestrator what options to use when creating the snapshot, so that it can instruct vCenter how the snapshot should be created. Let's move to the second tab in our workflow run screen and complete the remaining options. We can then start the workflow.

1. Select the tab **Enter snapshot creation parameters**
2. Type in **Test** for the snapshot name
3. Type in **My First Workflow** for the description

Leave all the other settings at the default values.

3. Click the **RUN** button to start the workflow

Observe the workflow running



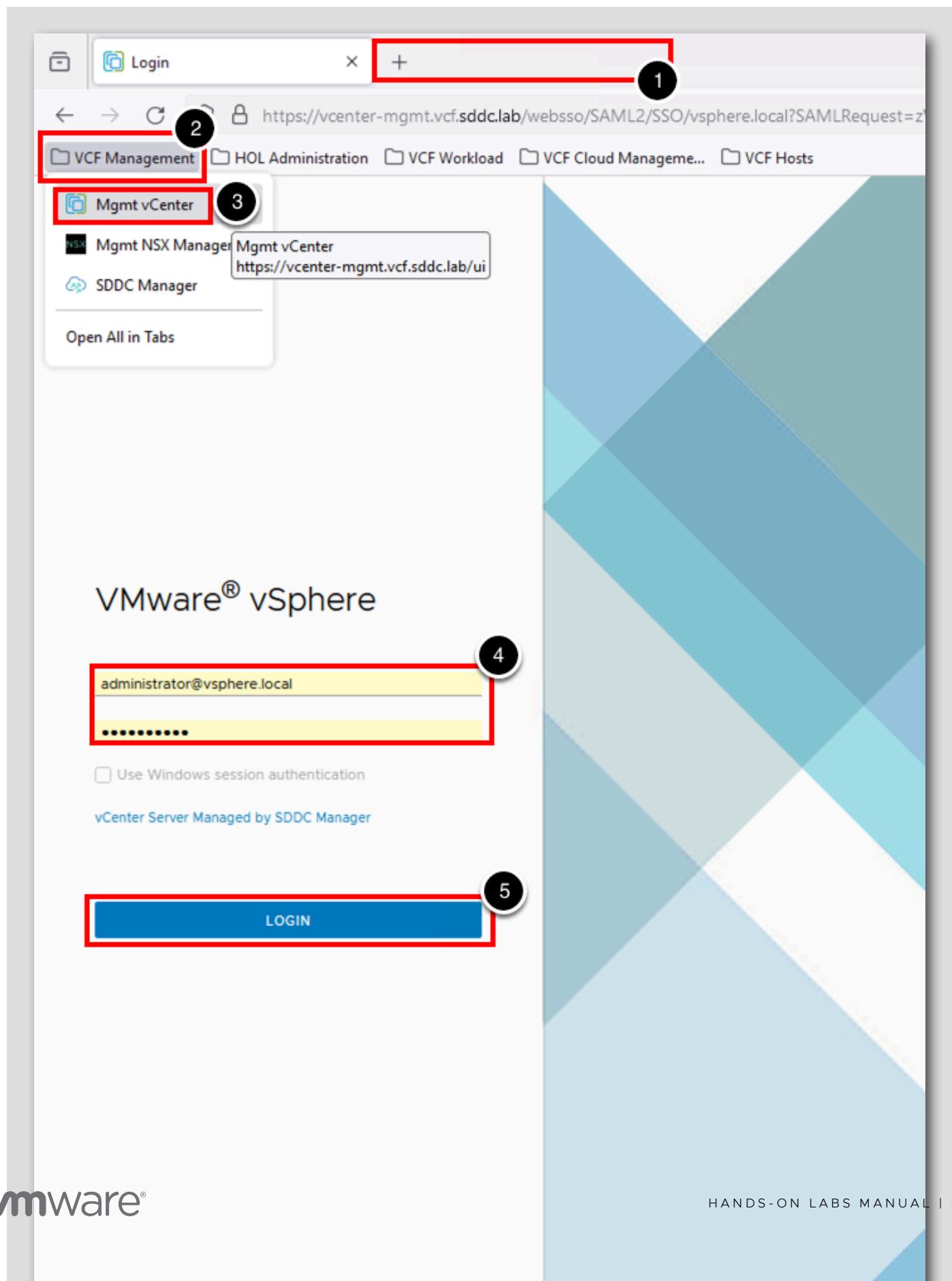
While the workflow is running, or after it has completed we can observe the following information:

1. The status of the workflow. We can see in the example that the workflow has completed, you may see a blue status of running if the workflow is still in progress.
2. For each step within the workflow, we can see how long it took for Orchestrator to complete this step. This is useful information for Administrators if a workflow takes a long time to complete. They are able to see where within the overall workflow a bottleneck, or long running task may be. In our example we can see the createSnapshot action took 155ms and the vim3WaitTaskEnd action took 8 seconds.

We will explore some of the other information provided by a workflow run in a later lesson. For now, let's wait until the workflow has completed and then check in vCenter to see what the outcome of the workflow was. We should see a snapshot created on the Virtual Machine we selected as a workflow input parameter.

Log in to the vSphere Web Client

[27]

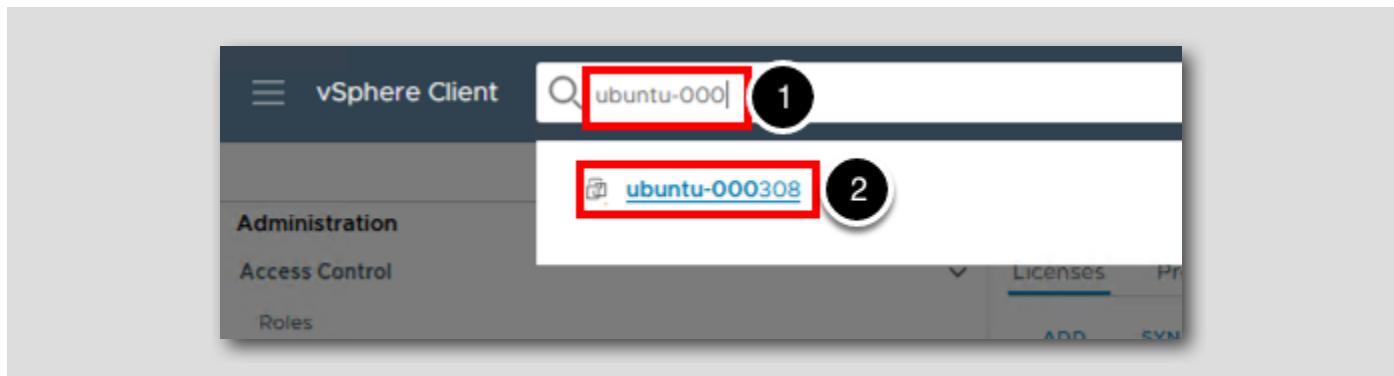


1. Within the Firefox browser, open a new tab
2. Click the VCF Management bookmark folder
3. Click on the Mgmt vCenter bookmark
4. The username and password fields will automatically populate with the credentials for the administrator@vsphere.local user
5. Click the LOGIN button

Wait while the vCenter inventory loads, we can then locate our Virtual Machine and check for snapshots.

## Search for the VM

[28]

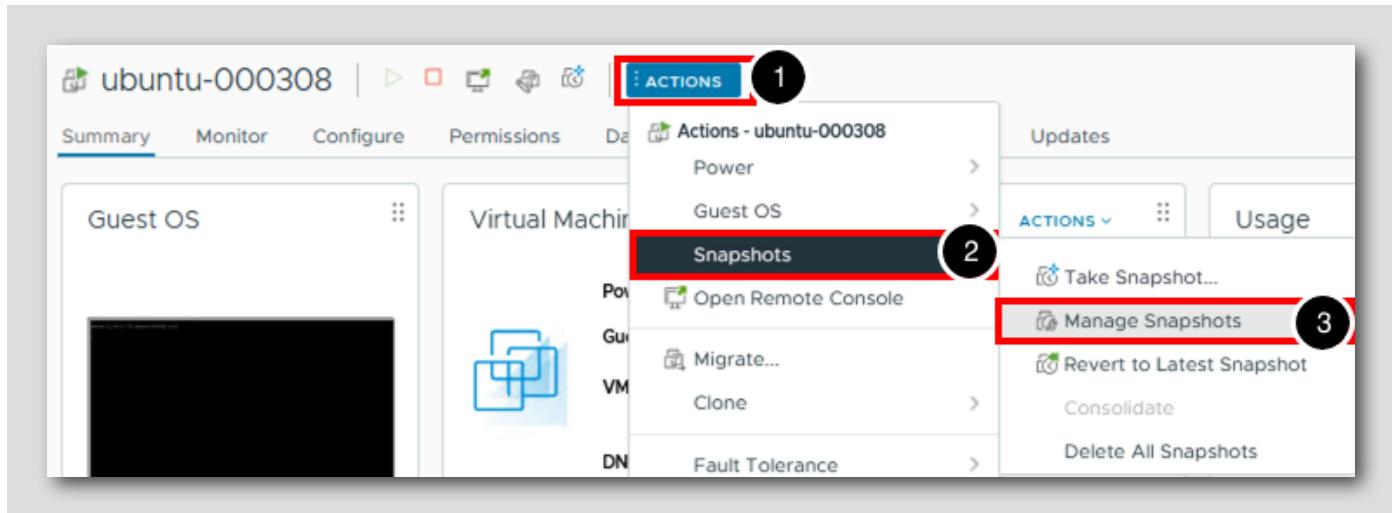


We could browse the inventory tree to locate our Virtual Machine, however since we know it's name let's use the Search functionality to jump straight to it.

1. Enter **ubuntu-000** into the search box at the top of the screen
2. Click on the **ubuntu-000xxx** machine in the search results, where **ubuntu-000xxx** is the name of the Virtual Machine you selected as the input parameter for the workflow earlier.

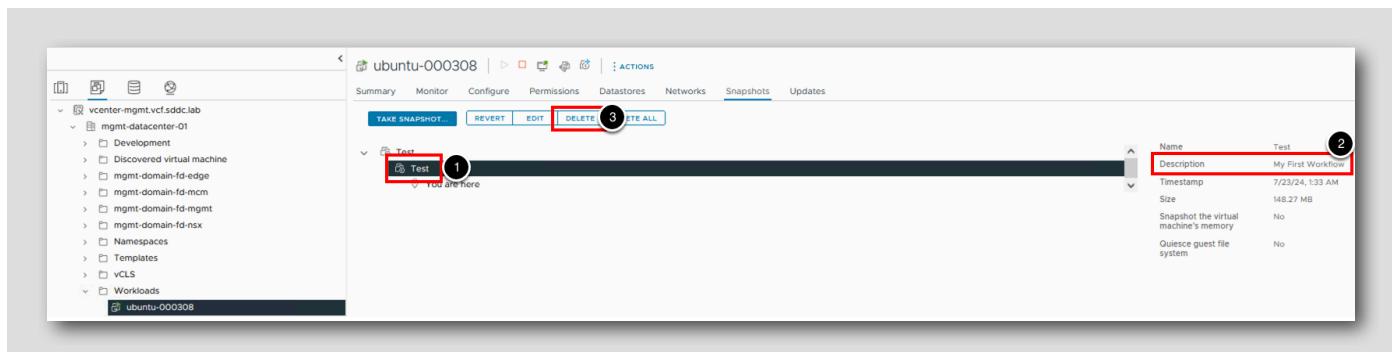
The name of the VM within your lab may be different from the screenshot as the VM is named automatically during deployment from Aria Automation. Select the VM with the name in the format **ubuntu-xxxxxx** you used as the input to the Orchestrator workflow e.g. **ubuntu-000308**.

## Open the Manage Snapshots page



1. When the Summary page for the Virtual Machine loads, click the ACTIONS button
2. Select Snapshots
3. Click on Manage Snapshots

## Review the snapshots on the Virtual Machine



Let's now review the snapshots that exist on the Virtual Machine and check that the snapshot we created via Orchestrator is present.

1. Select the **Test** snapshot
2. We can see that the workflow description is set to **My First Workflow**, matching the value we used in the Orchestrator workflow.
3. Now we have seen that the snapshot name and description match our Orchestrator workflow input values, let's go ahead and clean up the snapshot by deleting it. Click the **DELETE** button to start the deletion process.
4. In the confirmation prompt, click **DELETE** (not shown)

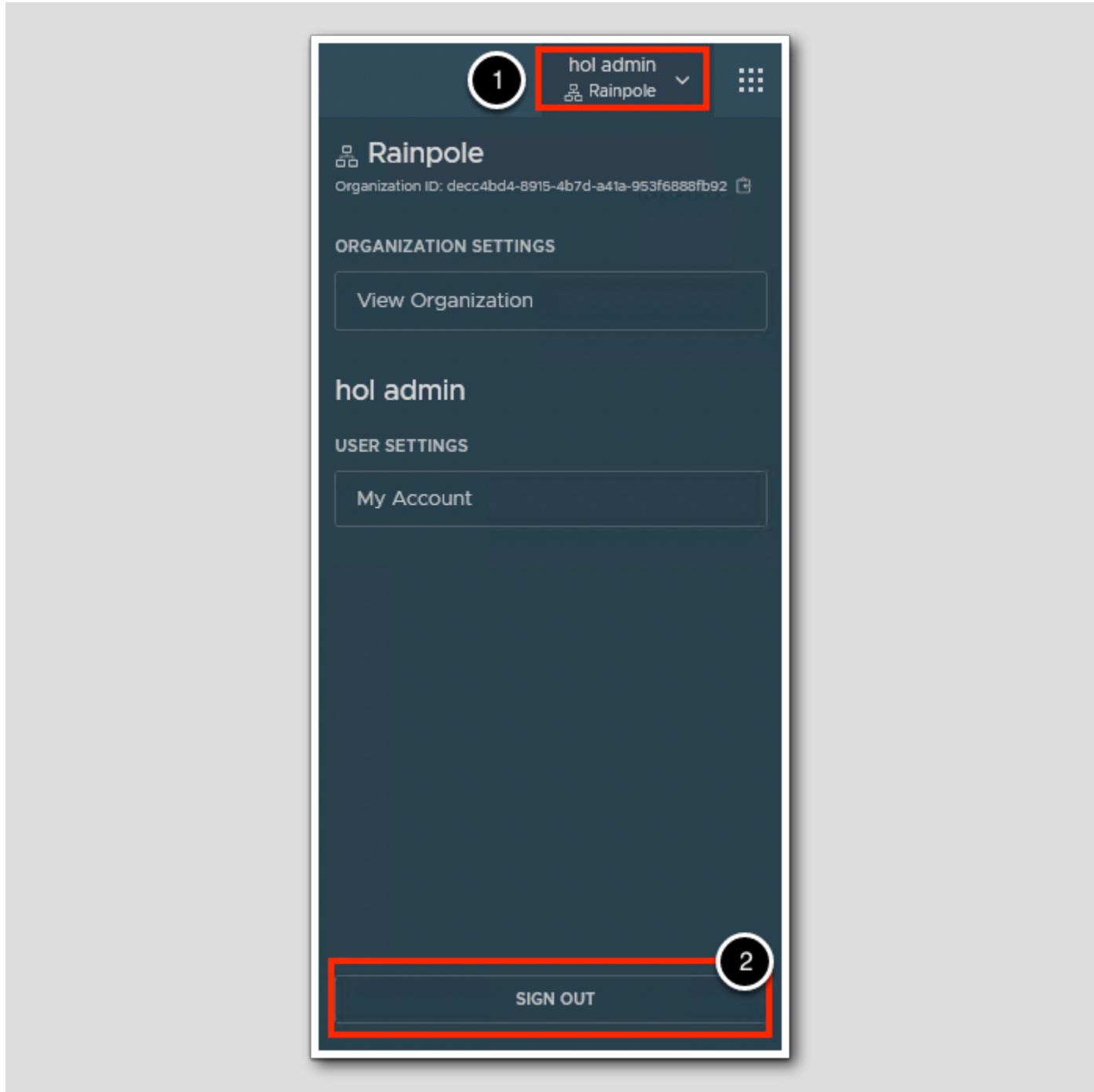
Close the Firefox Browser

[31]



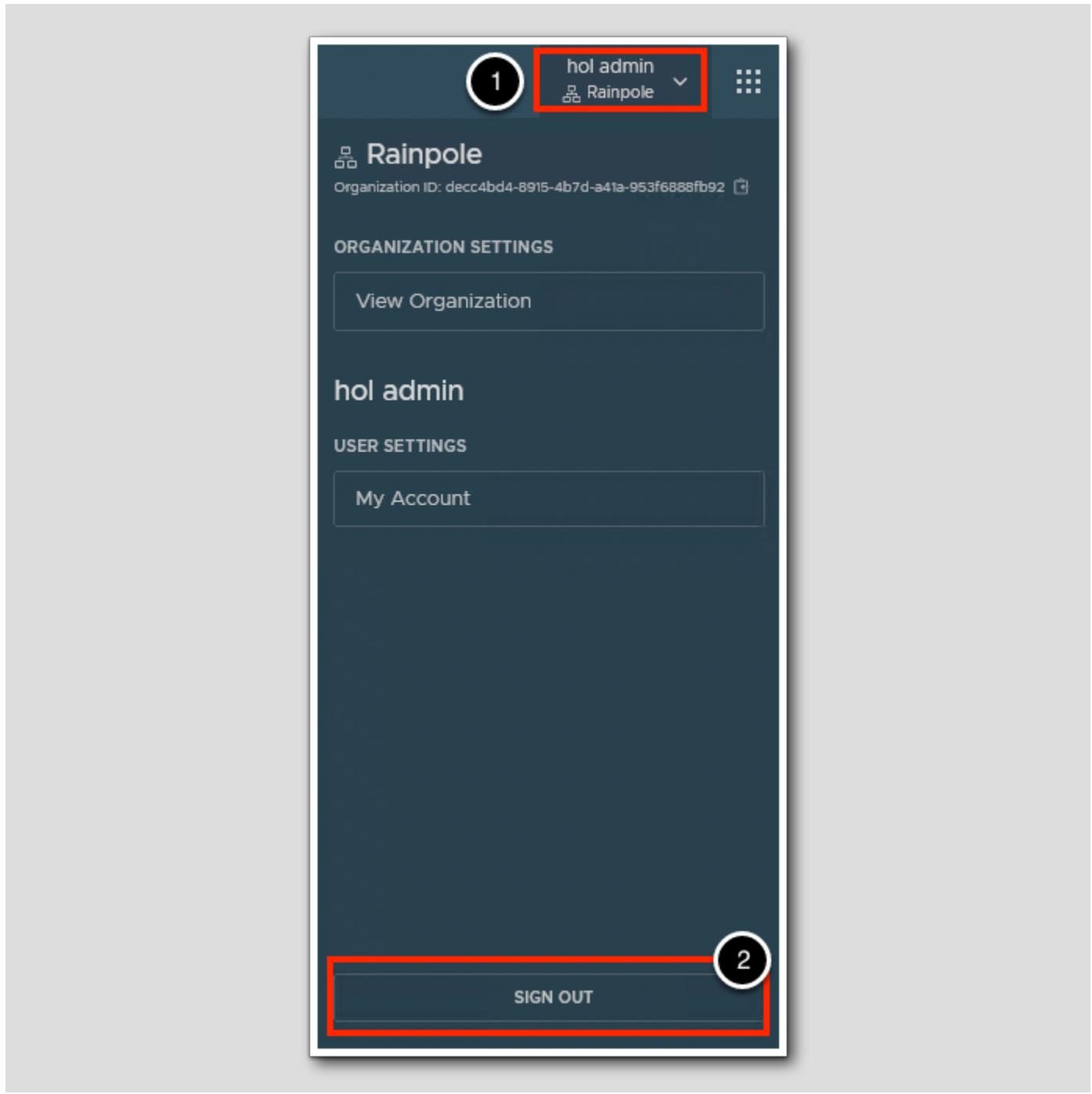
1. Click the X in the top right hand corner of the browser to close it

## Log Out of Aria Automation Orchestrator



To Log out of Aria Automation Orchestrator:

1. Click **hol admin** to open the organization menu.
2. Click **SIGN OUT**.



## Conclusion

[33]

Aria Automation Orchestrator has a very simple interface to perform a lot of operations on various systems (VMware products and non-VMware products).

Aria Automation Orchestrator's main interface is web based. This interface allows development and execution of new workflows and is accessed via the Aria Automation web interface.

From the inventory dashboard, a global view of all connected endpoints is now available.

## You've finished the module 1

[34]

Congratulations on completing the lab module.

If you are looking for additional information on Aria Automation Orchestrator basics, visit the VMware documentation on [Developing Workflows with VMware Aria Automation Orchestrator](#).

From here you can:

1. Continue with the next lab module
2. Click [vlp:table-of-contents] Show Table of Contents] to jump to any module or lesson in this lab
3. End your lab and return in the future

## Module 2 - Create a Basic Orchestrator Workflow (30 min) Basic

### Introduction

[36]

In this lesson, we will review the basic steps involved in the creation of Aria Automation Orchestrator workflows. The goal will be to create a simple Aria Automation Orchestrator workflow, by leveraging out-of-the-box content, that provide an example of how workflows can be created, run and reviewed.

#### Lab Captain(s):

- Katherine Skilling, Senior Architect, United Kingdom

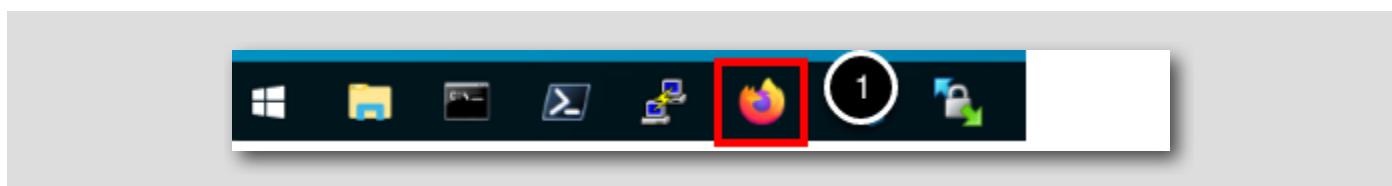
### Log in to Aria Automation Orchestrator

[37]

In the following few pages, we will log in to Aria Automation Orchestrator.

### Open the Firefox Browser from Windows Quick Launch Task Bar

[38]

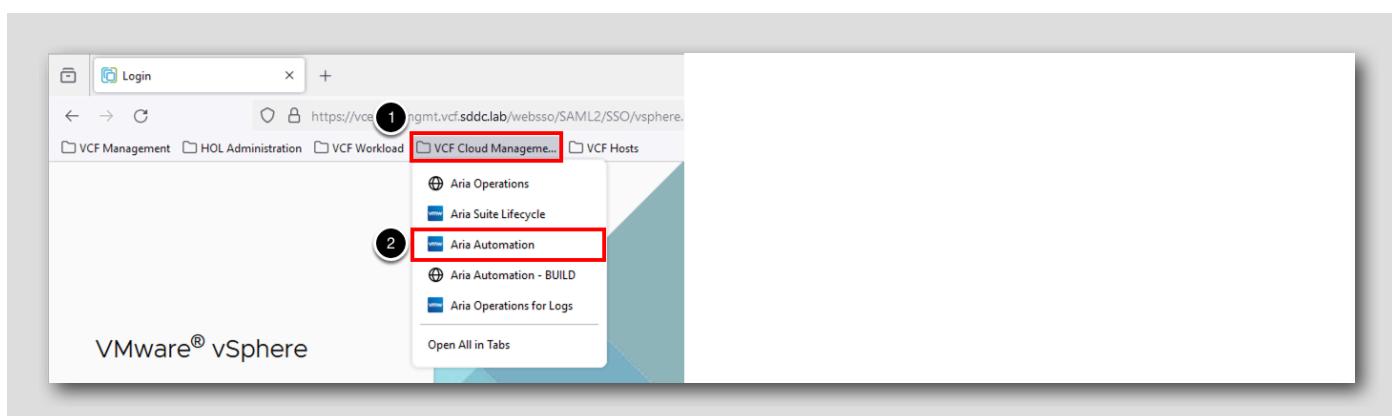


If the browser is not already open, launch Firefox.

1. Click the Firefox icon on the Windows Quick Launch Task Bar.

### Log in to Aria Automation

[39]

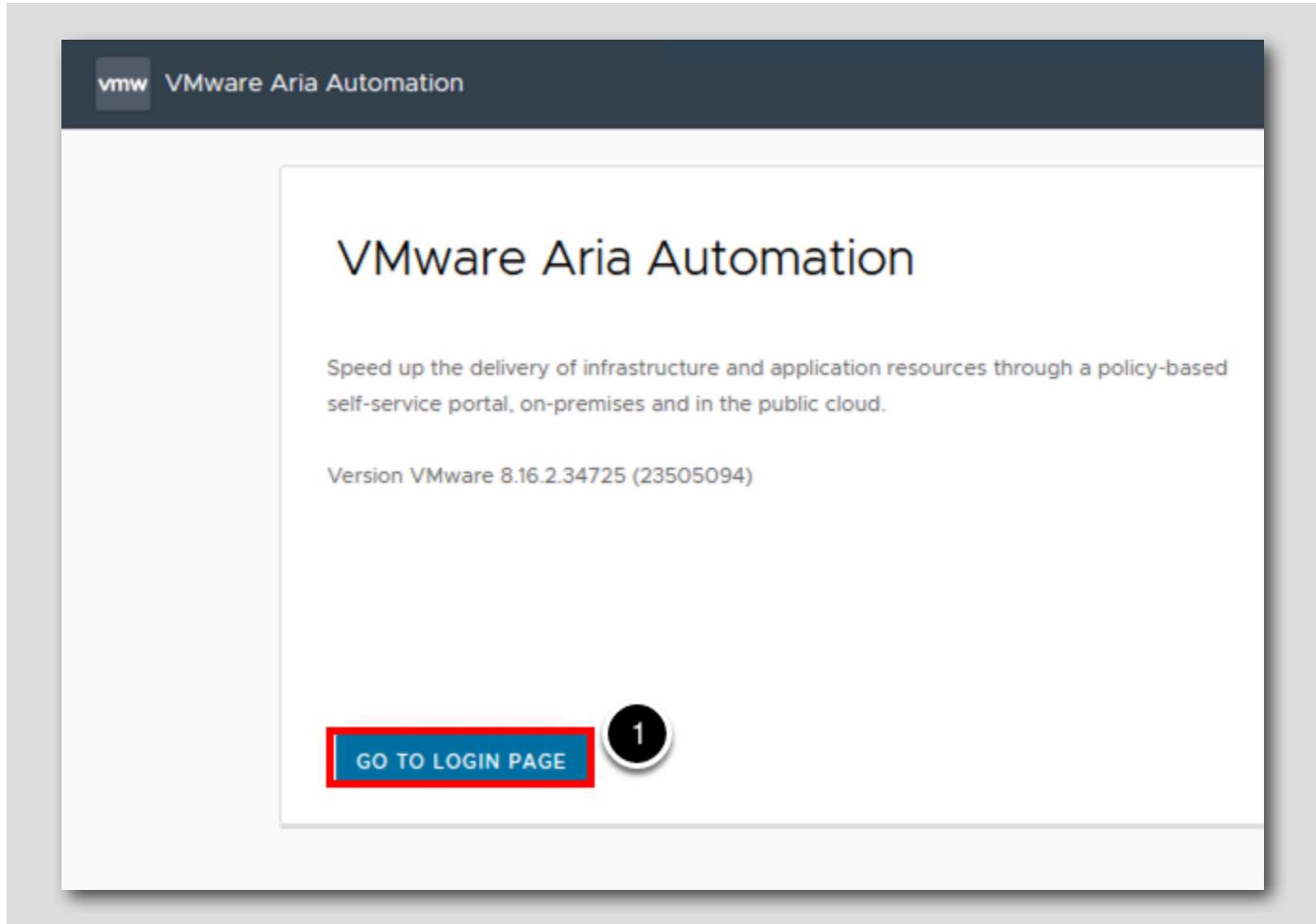


Once Firefox has loaded:

1. Click the VCF Cloud Management bookmark folder
2. Click Aria Automation.

Redirect to Workspace ONE Access for Sign-On

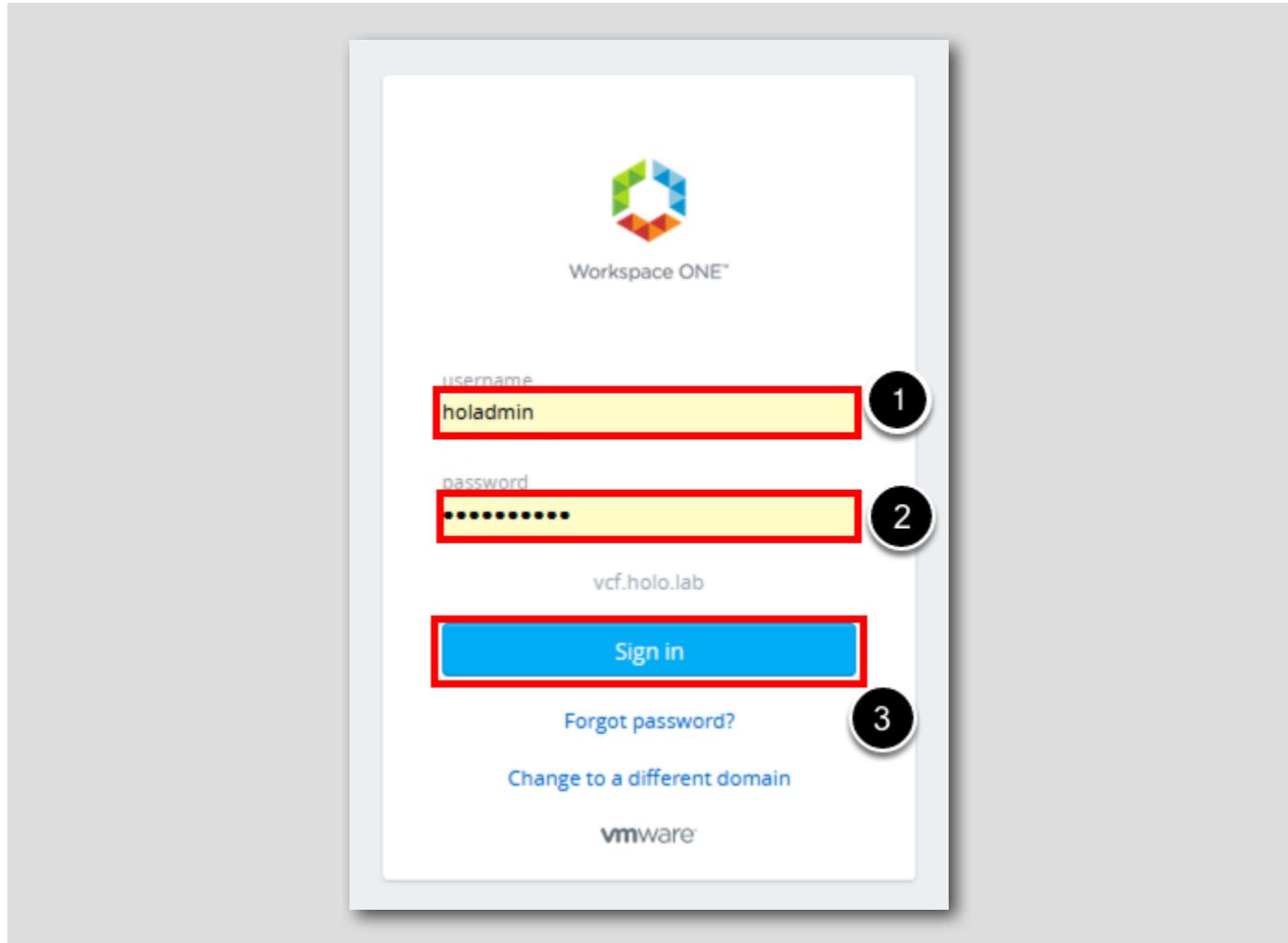
[40]



Aria Automation is integrated with Workspace ONE Access (aka VMware Identity Manager) and we need to redirect to the Workspace ONE Access login page to complete our log in progress.

1. At the VMware Aria Automation page, click GO TO LOGIN PAGE.

## Workspace ONE Access Login

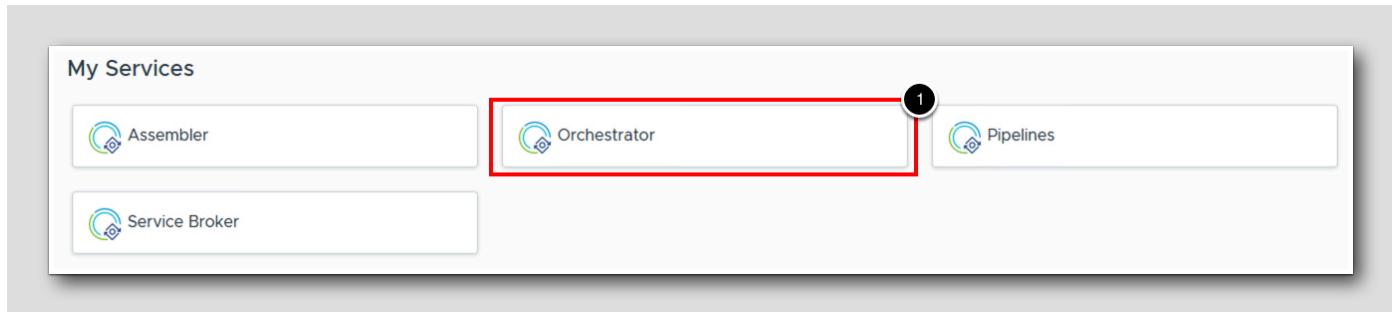


The credentials for **holadmin** should already be cached in the browser window.

At the **Workspace ONE Access** prompt, type in the following user and password information.

1. At the **username** field, type **holadmin**.
2. At the **password** field, type **VMware123!**.
3. Click **Sign in**.

## Launch the Orchestrator Service



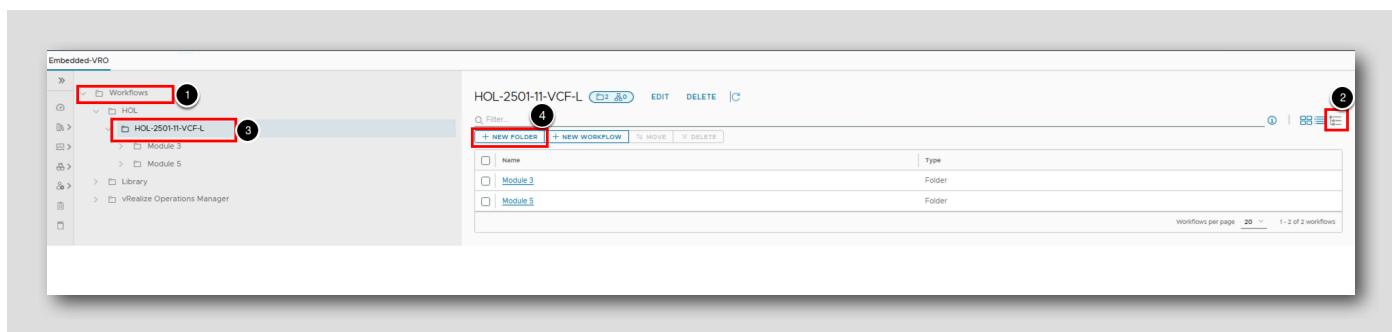
From within the Cloud Services Console, under **My Services**:

1. Click the Orchestrator service.

## Create a custom workflow

Aria Automation Orchestrator workflows allow us to perform a variety of tasks. Each workflow can have multiple inputs and return multiple outputs. In this lesson we will cover the creation and running of a simple workflow with 2 steps.

## Create a new workflow folder



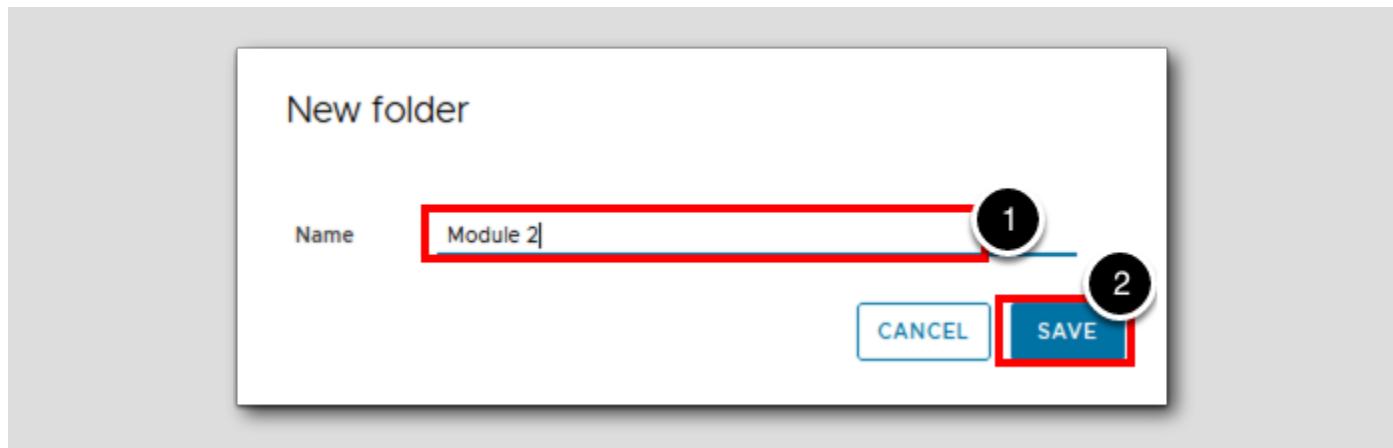
The first task we will complete is to create a new workflow folder. Folders can be used to organise workflows together, for example a folder could contain all workflows that are related to integrating with Aria Automation day 2 actions while another folder contains all workflows that an individual administrator is in the process of developing.

We will create a new folder based on the module we are currently working on, module 2.

1. Select the **Workflows** tab
2. Make sure the **folder view** is selected
3. Select the **HOL>HOL-2501-11-VCF-L** folder
4. Click on the **+ NEW FOLDER** button

Complete the folder creation

[45]

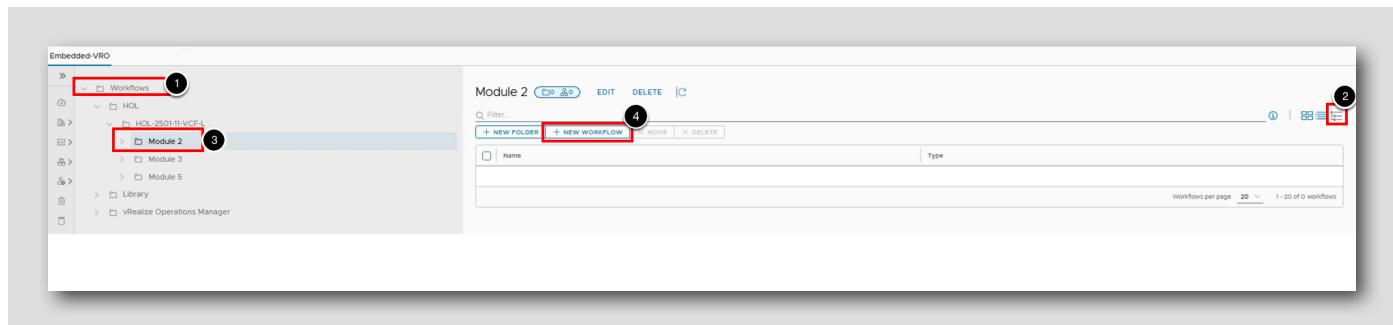


1. In the name box type **Module 2**
2. Click **SAVE** to create the folder

Once the folder is created notice that we already have folders in our inventory for other modules within this lab.

Create a new workflow

[46]

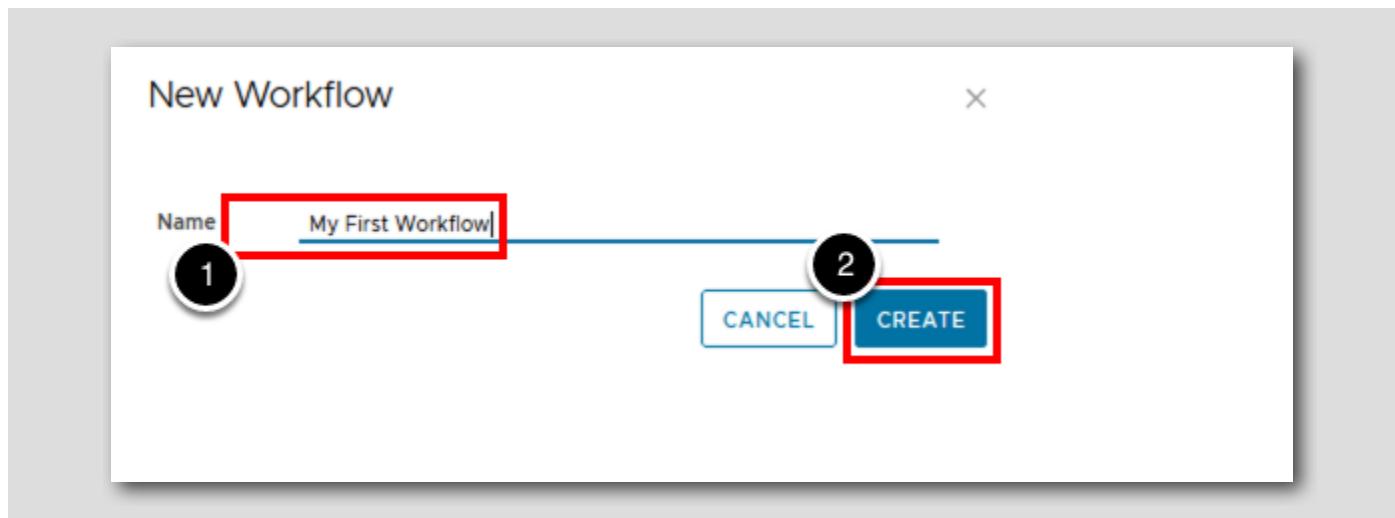


In Aria Automation Orchestrator, to automate a process, a workflow needs to be created. Let's now create our first custom workflow.

1. Select **Workflows** tab
2. Make sure the **folder view** is selected
3. Select the **HOL>HOL-2501-11-VCF-L>Module 2** folder
4. Click the **+ NEW WORKFLOW** button

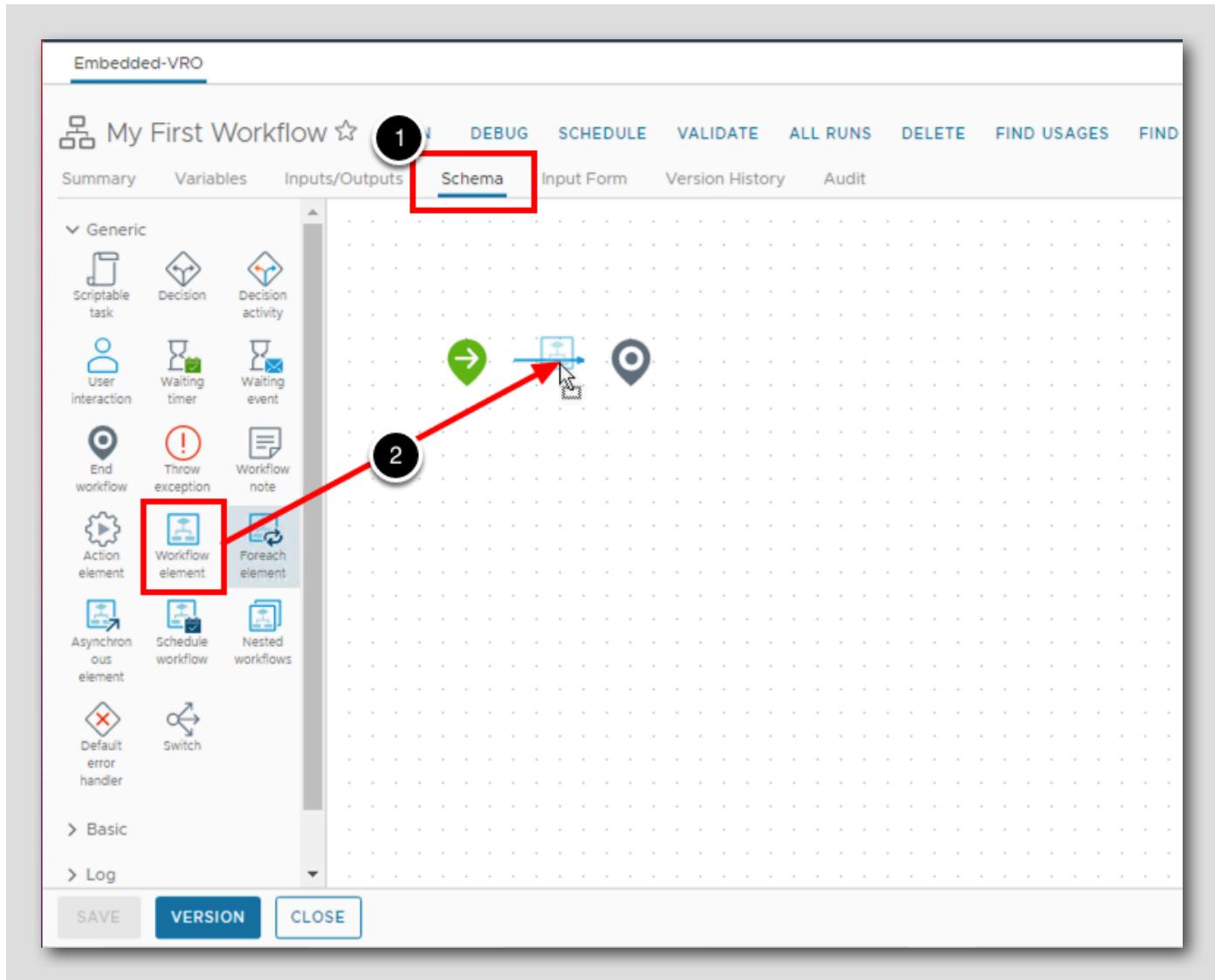
Name the workflow

[47]



1. Type **My First Workflow** for the workflow name
2. Click the **CREATE** button

## Add a workflow element



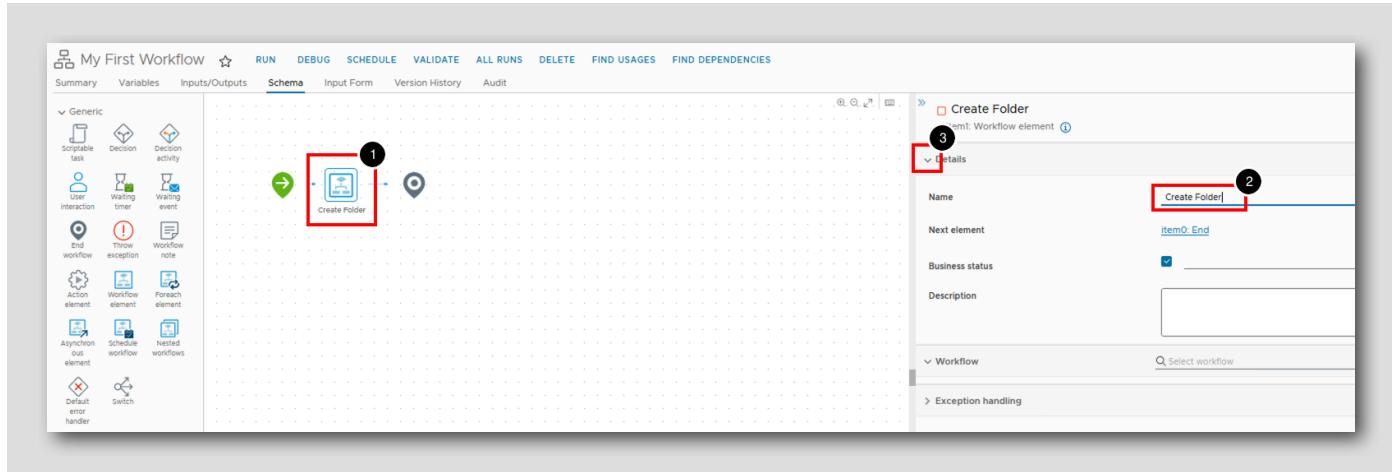
The green arrow represents the **Start workflow** element and the one to the far right is the **End workflow** element. These two elements are mandatory, and must be present for the workflow to run.

Workflow schemas are made up of elements. Take a moment to look through the list of available elements.

Now let's add a workflow element to the schema.

1. Select the Schema tab
2. Drag the **Workflow element** item onto the schema between the two existing elements on the canvas

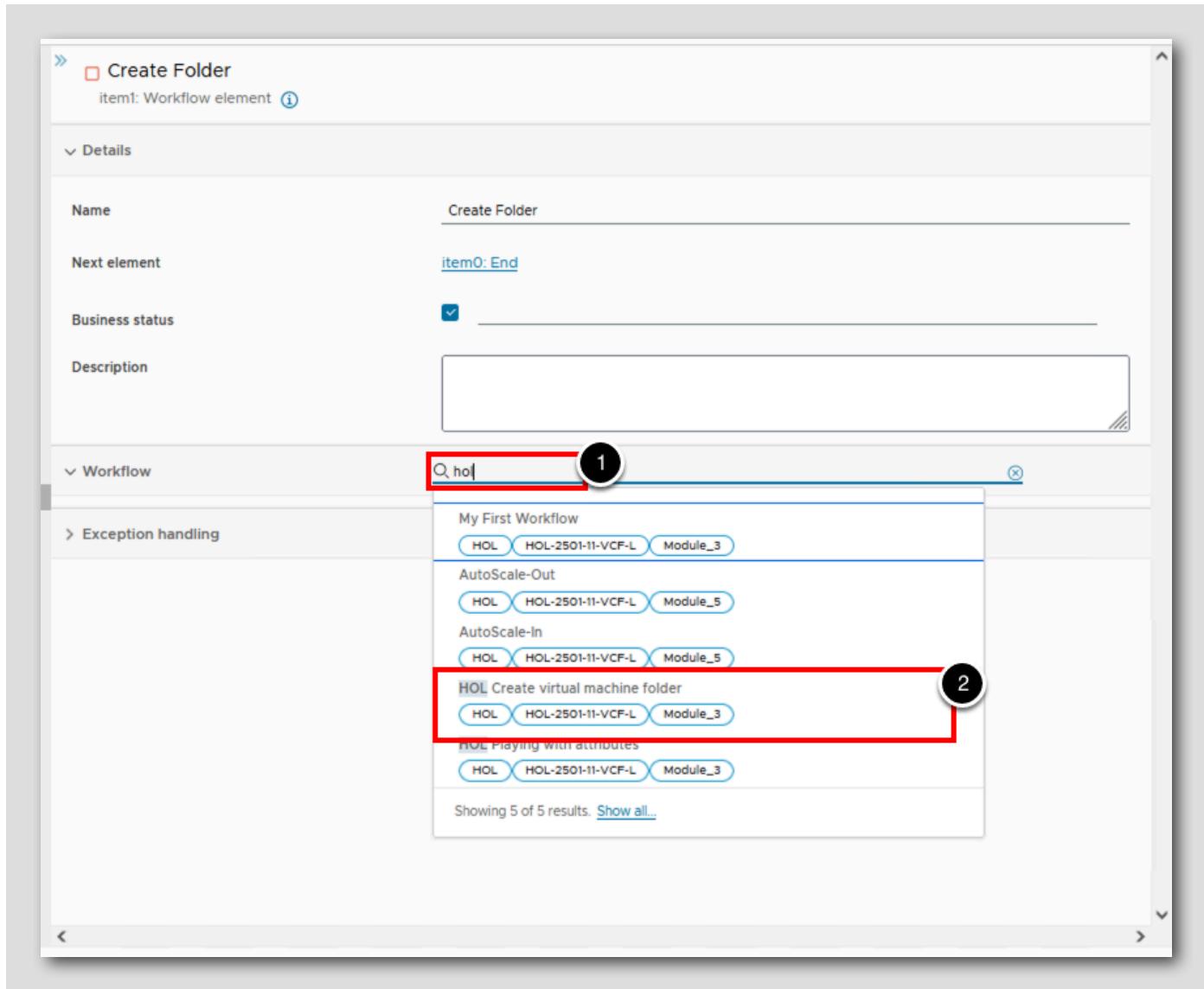
## Configuring the new workflow element



To make the overall workflow more understandable and easier to maintain, it is a good practice to properly name each element. In this way, the steps in the workflow become descriptive and help support documentation.

1. Make sure the workflow element is selected in the Schema
2. Enter a name for the element: **Create Folder**
3. Collapse the Details section

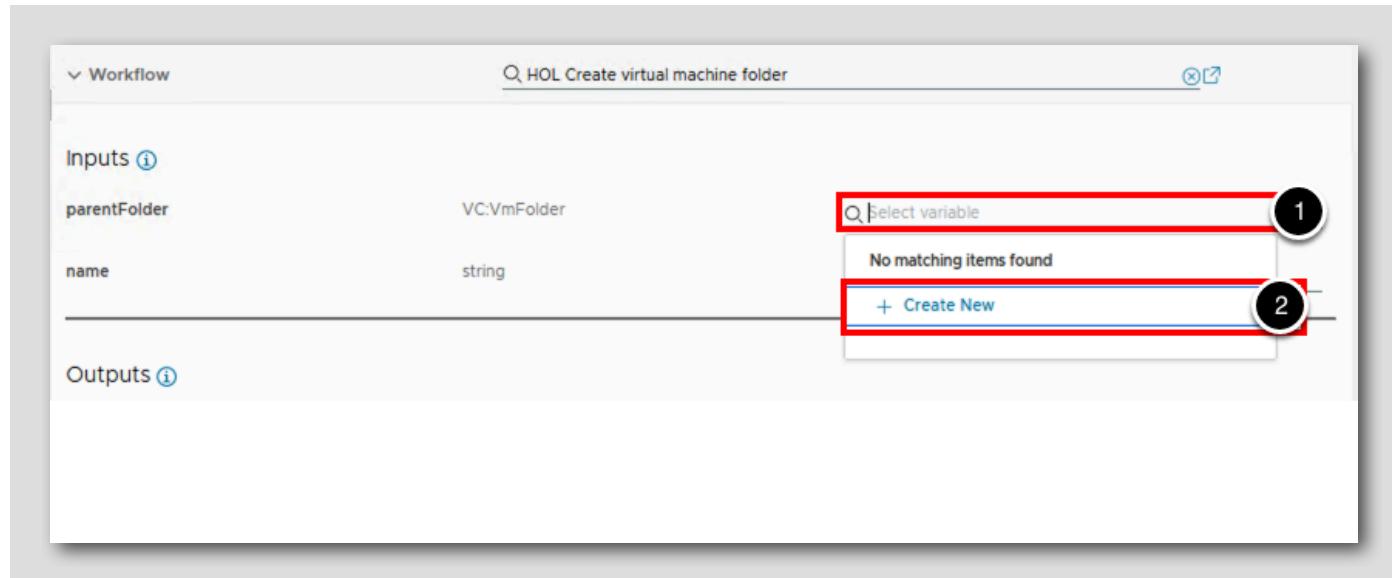
## Select workflow



The workflow element we dragged onto the canvas allows us to call an existing Orchestrator workflow from within our workflow. This allows us to reuse the functionality without having to copy the code or schema of the workflow. Let's now specify the workflow we want to reuse by selecting it from the inventory.

1. Click the Workflow field and type: hol
2. Select the workflow HOL Create virtual machine folder from the list of workflows

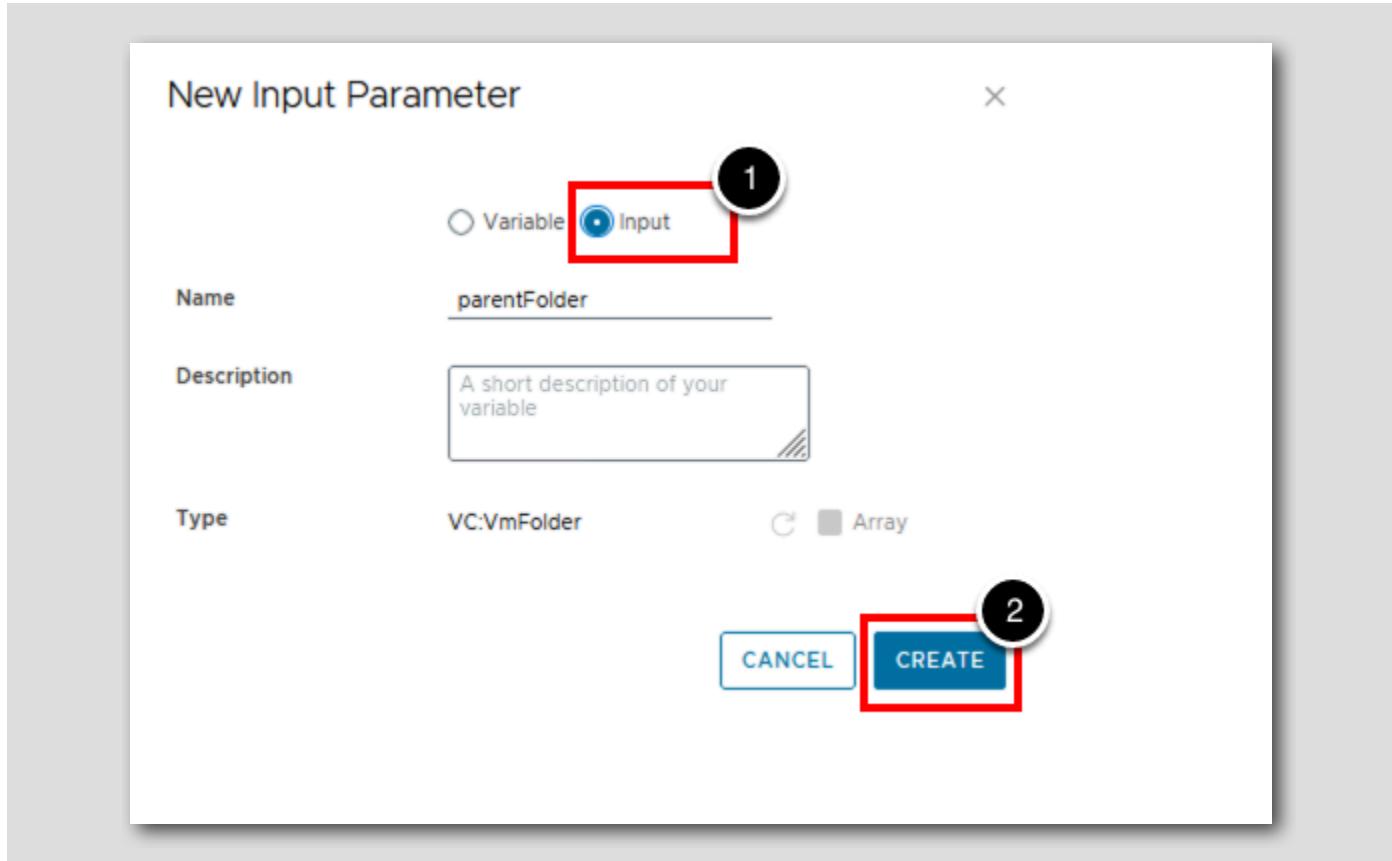
## Assign parentFolder variable



Once a workflow has been selected, a list of inputs is needed. These inputs will either come from the current workflow inputs (and will be filled by the user when the workflow is executed) or they will come from the workflow local variable that can have a predefined value.

1. Click the field for parentFolder
2. Click Create New

## Create parentFolder input



Let's create a new input parameter to represent the parentFolder value.

1. Select Input

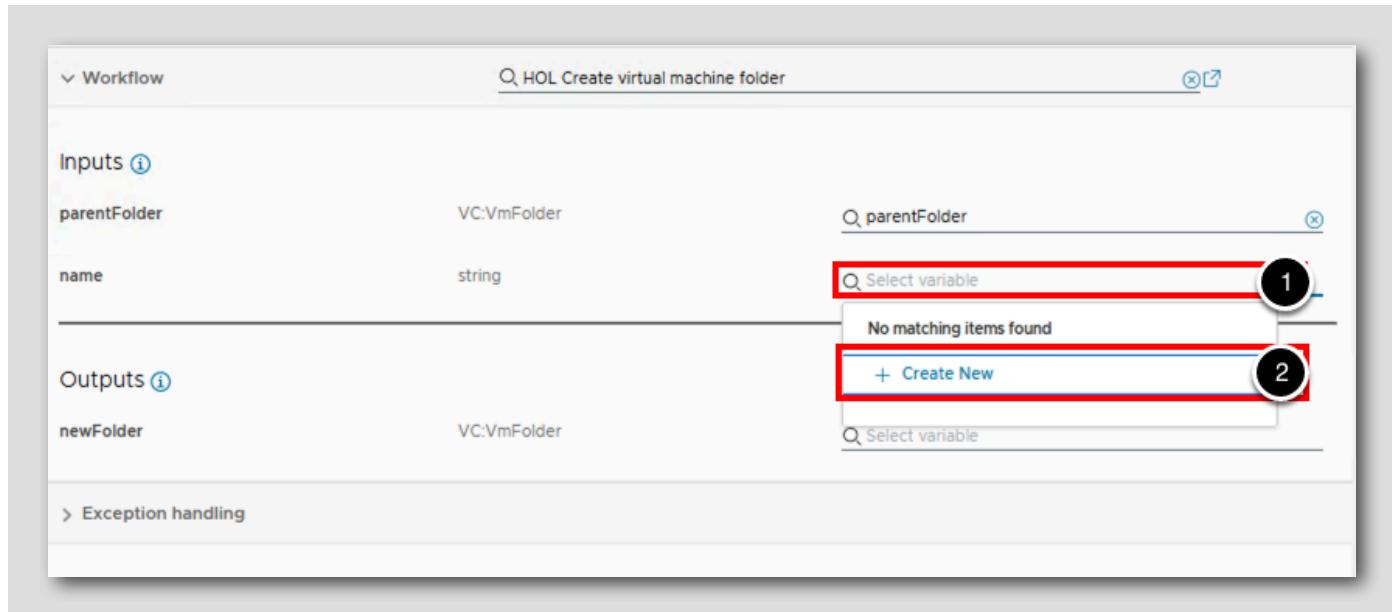
(it indicates that a value will be expected from the user who executes the workflow)

2. Click the CREATE button

This process creates a new input for the current workflow which, during the execution, will be passed to the "Create Folder" workflow.

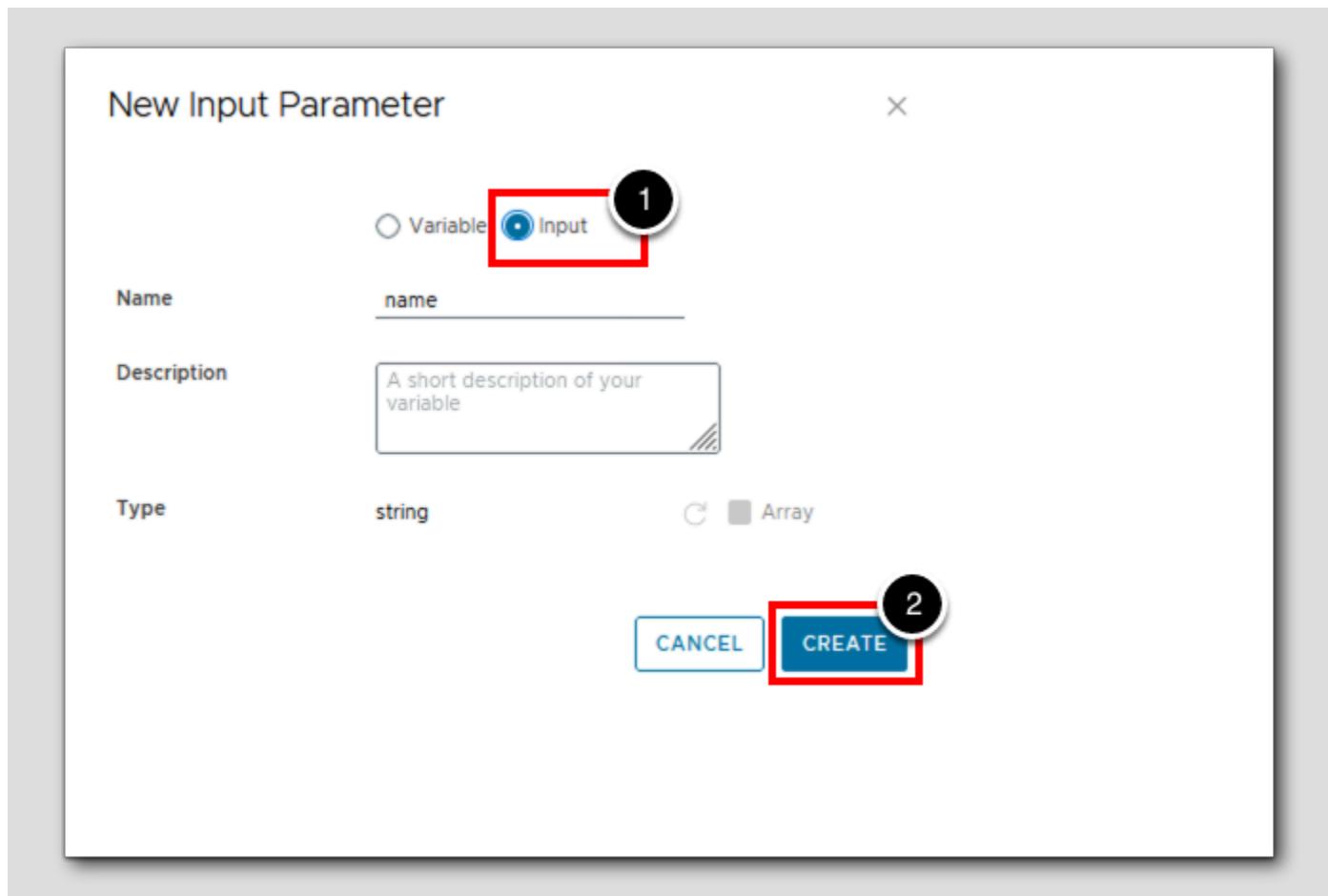
The next steps consist of repeating this operation for all the remaining inputs required by the "Create Folder" workflow.

## Assign name variable



1. Click the field for name
2. Click Create New

## Create name input



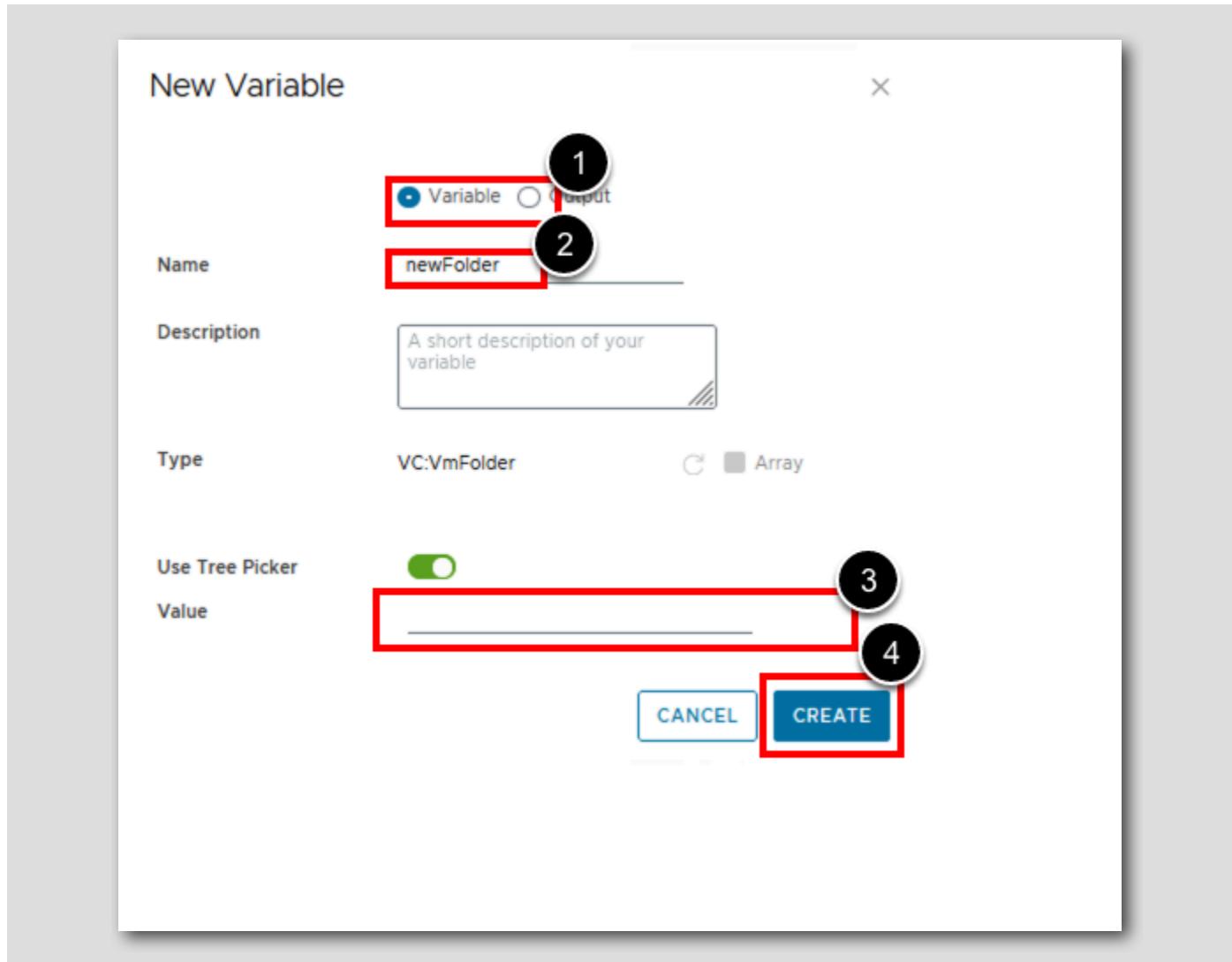
1. Select Input
2. Click the CREATE button

## Assign newFolder variable

The screenshot shows the 'Workflow' interface for creating a virtual machine folder. The 'parentFolder' input is set to 'VC:VmFolder'. The 'name' input is empty. The 'newFolder' output is set to 'VC:VmFolder'. A dropdown menu is open over the 'newFolder' field, showing 'No matching items found' at the top, followed by two options: '+ Create New' (option 2) and 'Select variable' (option 1). Both options are highlighted with red boxes and numbered circles.

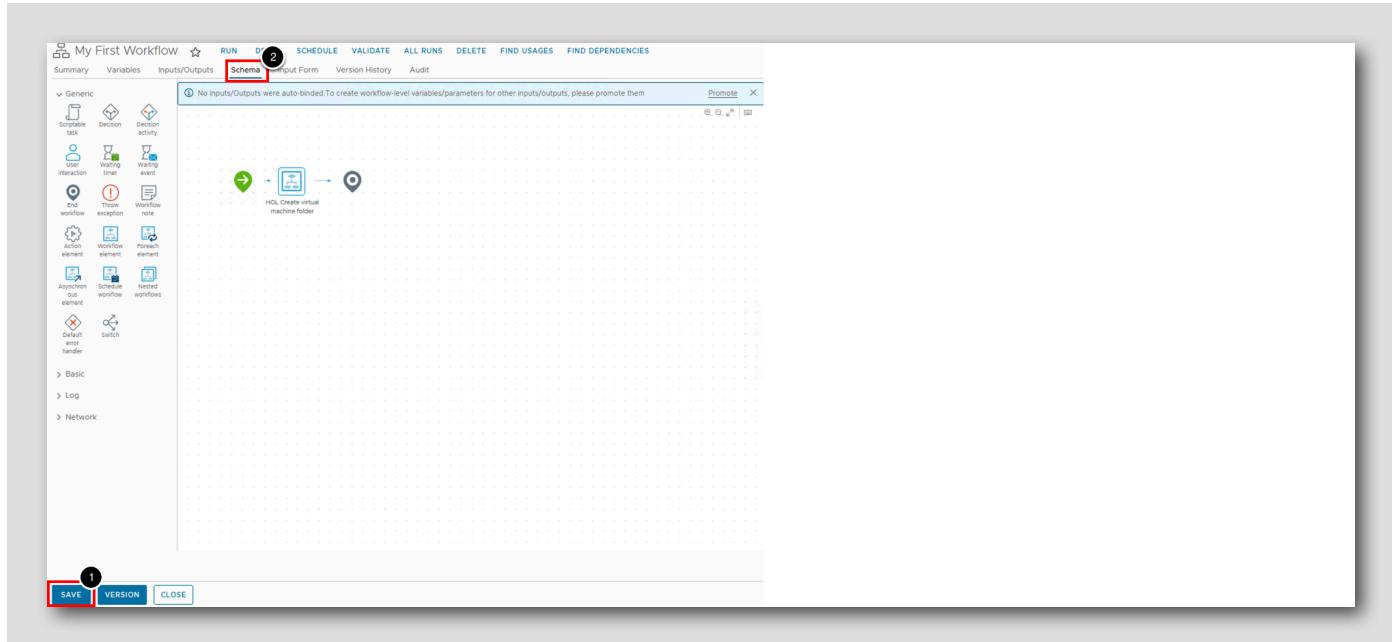
1. Click the field for newFolder
2. Click Create New

## Create newFolder variable



1. Keep **Variable** selected
2. Keep the pre-populated name of **newFolder**
3. Leave the **Value** field empty
4. Click the **CREATE** button

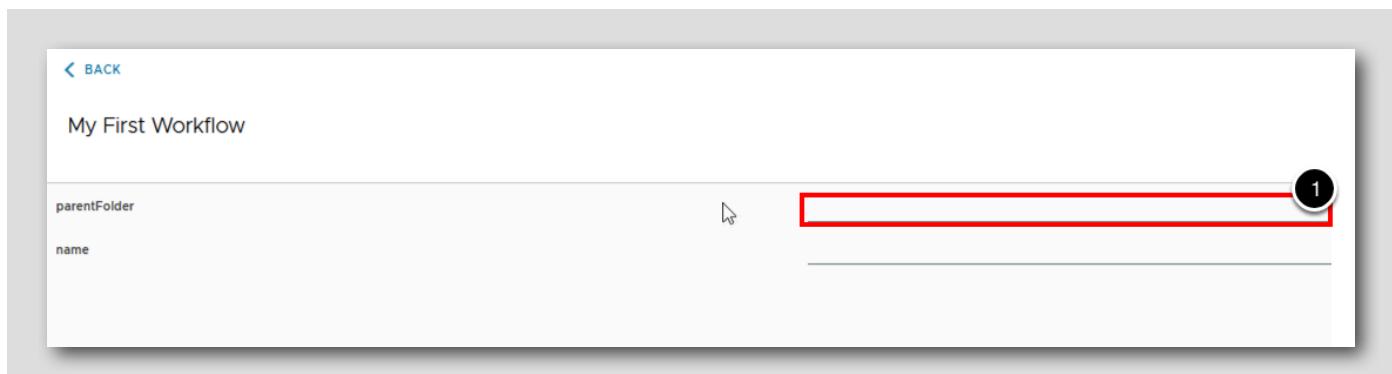
## Run the workflow



With all of the inputs mapped, we are now ready to run the workflow.

1. Click the **SAVE** button. [Not shown] In the prompt, click the **SAVE** button.
2. Click **RUN**

## Set the parentFolder input value

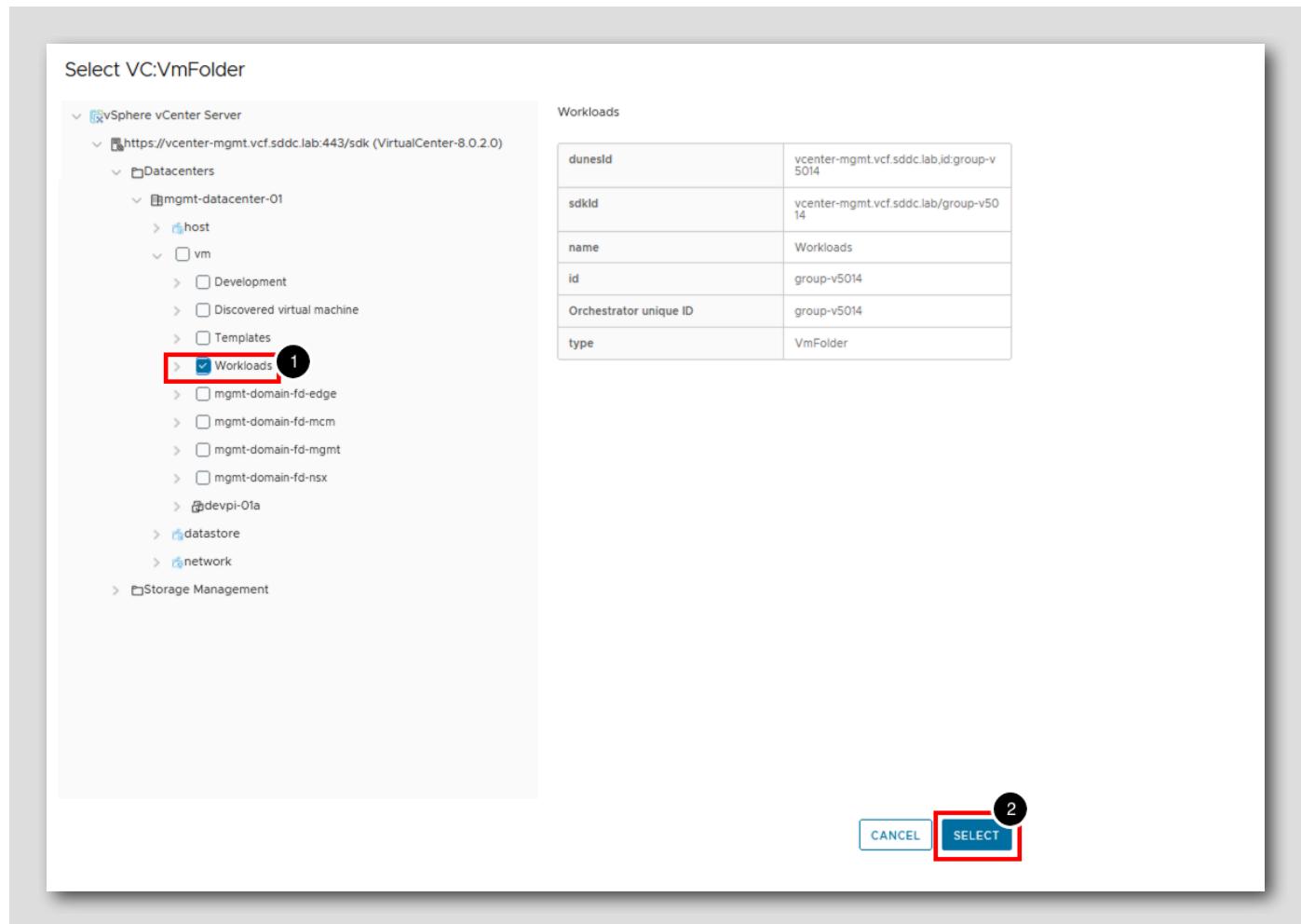


As we set the parentFolder parameter as an input parameter, we are prompted to select the folder to use as the parent within which our new folder will be created.

1. Click the field for parentFolder

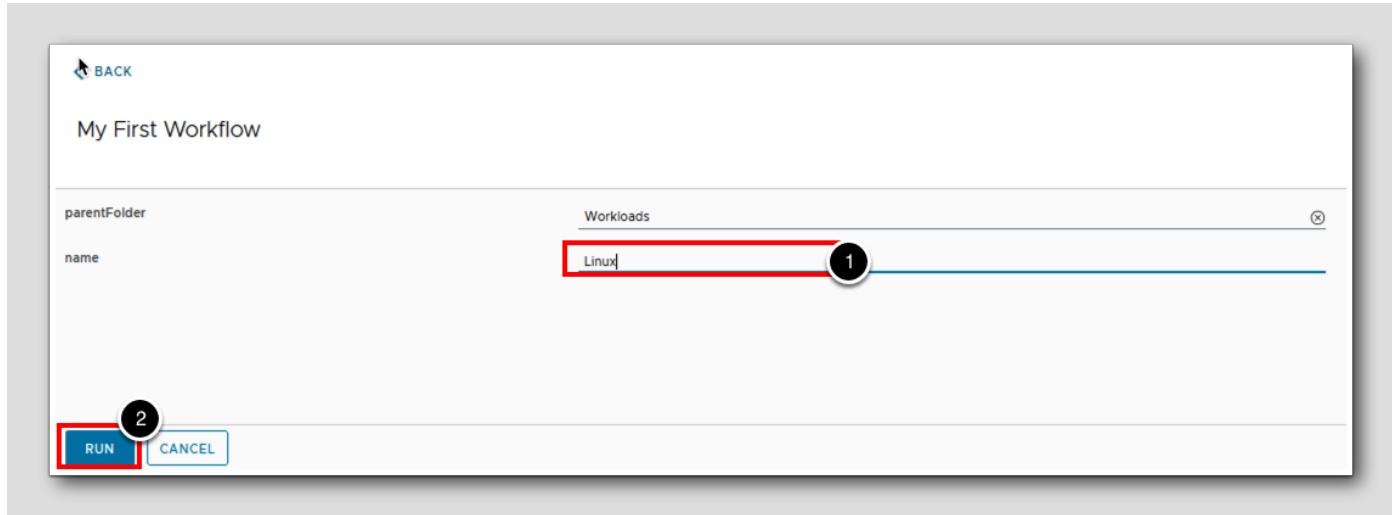
Select the folder from the inventory

[59]



1. Browse and select the folder vSphere vCenter Plug-in > <https://vcenter-mgmt.vcf.sddc.lab:443/sdk> > Datacenters > mgmt-datacenter-01 > vm > Workloads
2. Click the SELECT button

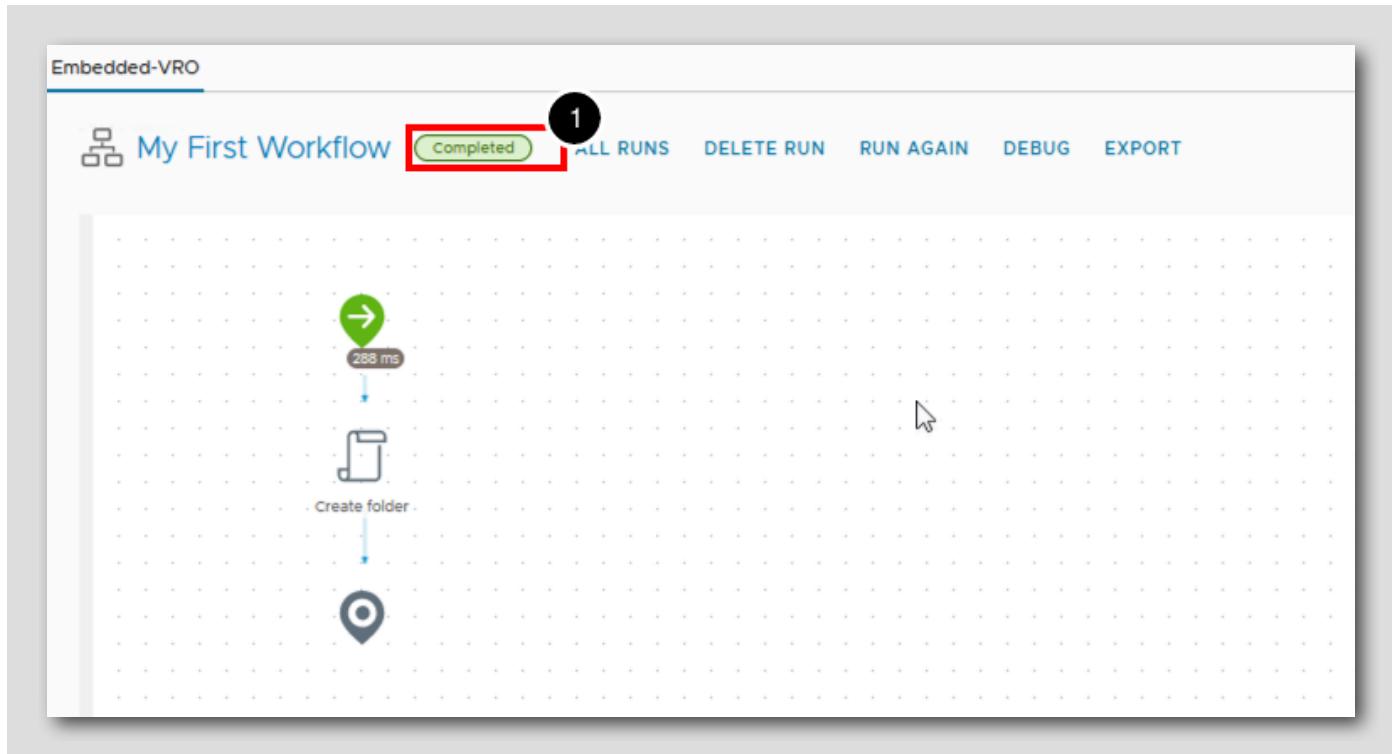
Enter the name input value and run the workflow



Before we can run our workflow we need to specify a name for the new folder we want to create.

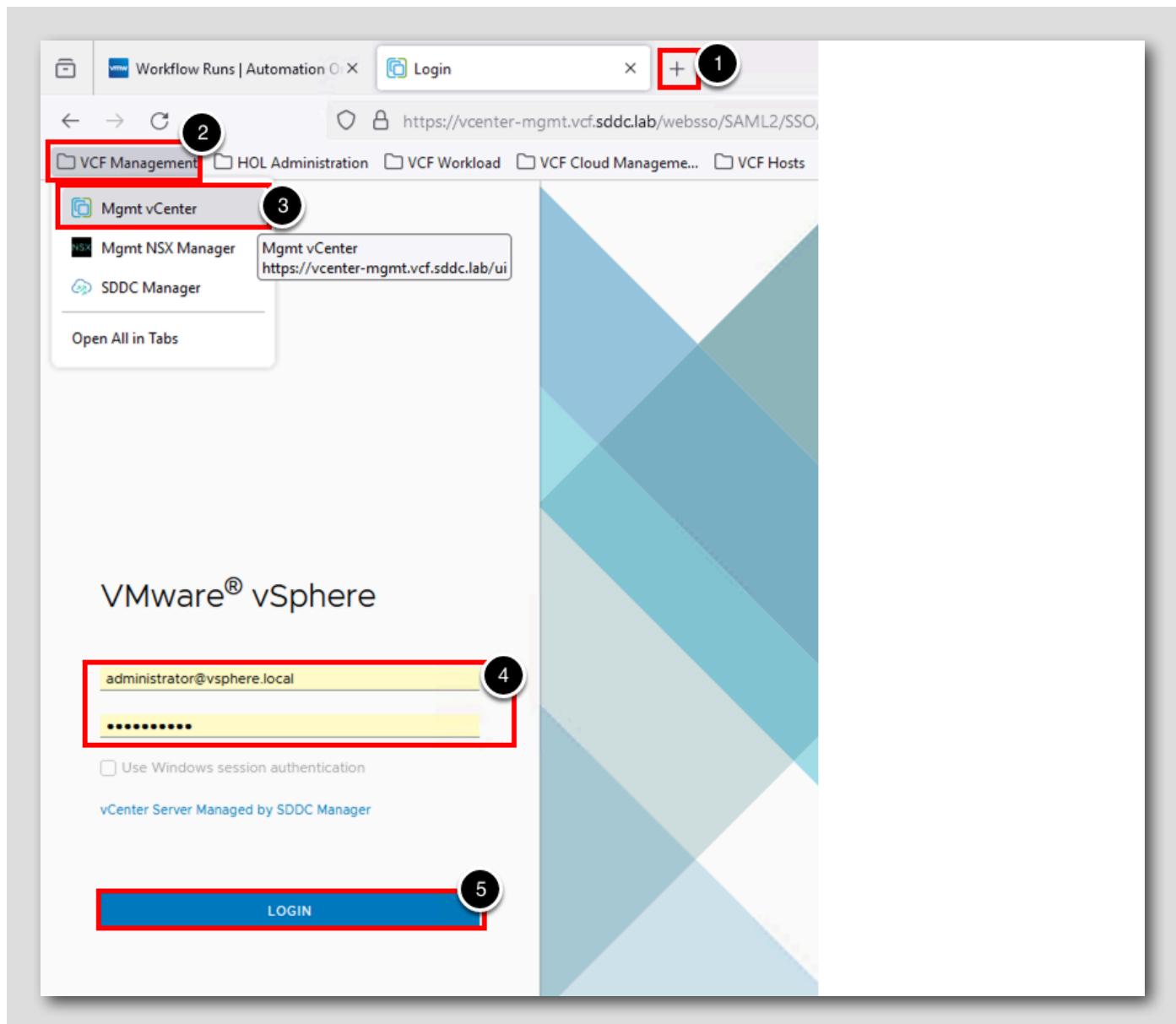
1. In the **name** field enter the value **Linux**
2. Click the **RUN** button to start the workflow

Wait for the workflow to complete



1. The workflow will take a few seconds to run and complete. We can monitor its progress via the status icon next to the workflow name. Wait until the workflow status shows as **Completed**. Leave the workflow run screen open for use in a later lesson.

## Login to vCenter

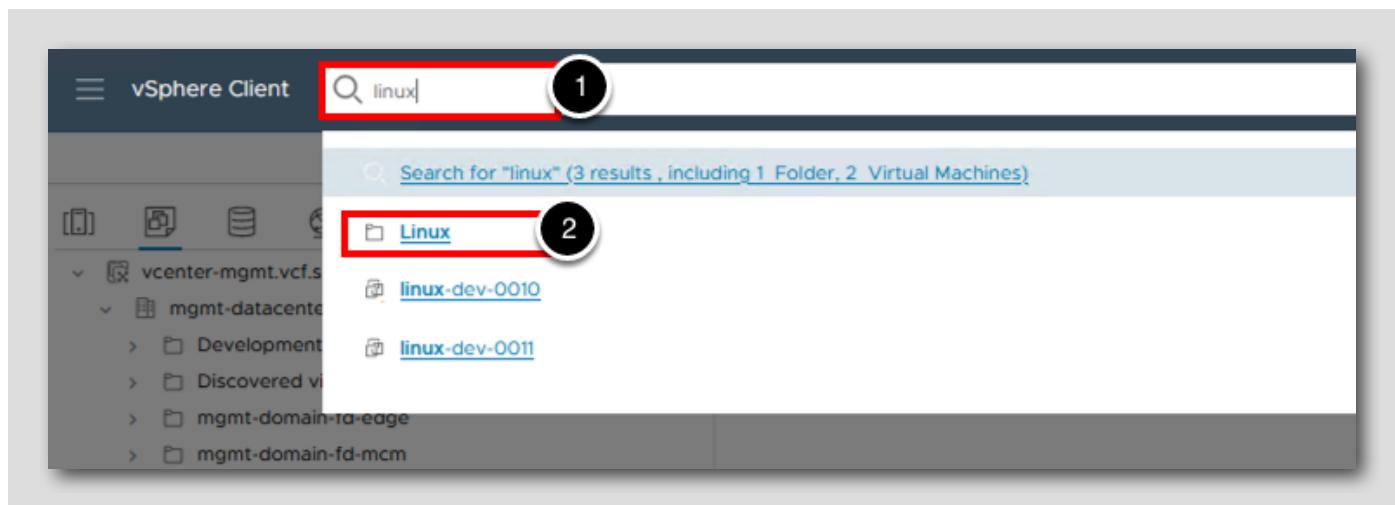


With the workflow completed successfully we should now have a new folder within our vCenter inventory. Let's log in to vCenter and review the folder structure.

1. In Firefox, click the + button to **add a new tab**
2. Select bookmark **VCF Management** from the bookmarks bar
3. Select **Mgmt vCenter** from the drop down list of bookmarks
4. The credentials for the **administrator@vsphere.local** user will automatically populate
5. Click the **LOGIN** button

### Search for the folder

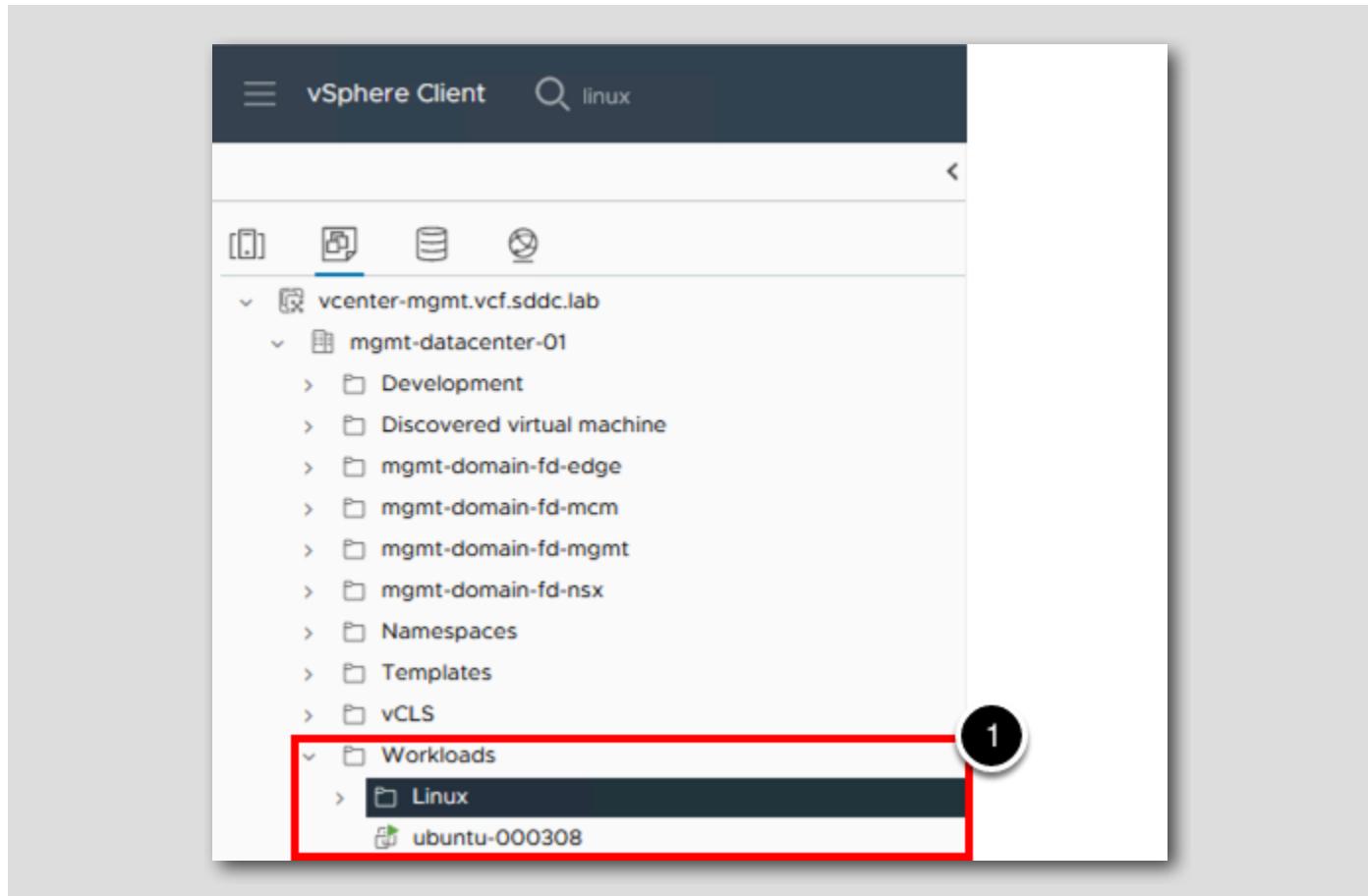
[63]



1. Click the search bar at the top and enter the text: **linux**
2. Click the folder **Linux**

## View the vCenter inventory

[64]



1. The Linux folder appears within the vCenter inventory as a child of the Workloads folder, at the same level as the virtual machines. This confirms that our workflow completed successfully and the folder was created using the values we specified.

## Understand a Workflow Token

[65]

A workflow token represents an instance of a workflow that is running or has run previously. Let's review the workflow we created and ran in the last lesson to explore an example workflow token.

## Observe the workflow token

The screenshot shows the Embedded-VRO interface for a workflow named "My First Workflow". The workflow has completed one run, indicated by the green "Completed" button at the top. The run details are shown in the center, including a green arrow icon with a "288 ms" label, a "Create folder" step, and a location pin icon. Below this, the "General" tab of the run details is selected, showing the following information:

|            |                                      |
|------------|--------------------------------------|
| ID         | 6afb02e9-f9fb-43aa-a2a6-6c4a20554568 |
| Start Date | Aug 5, 2024 3:15:48 AM               |
| End Date   | Aug 5, 2024 3:15:49 AM               |
| Status     | Completed                            |
| Started by | holadmin@vcf.holo.lab                |
| Workflow   | My First Workflow                    |

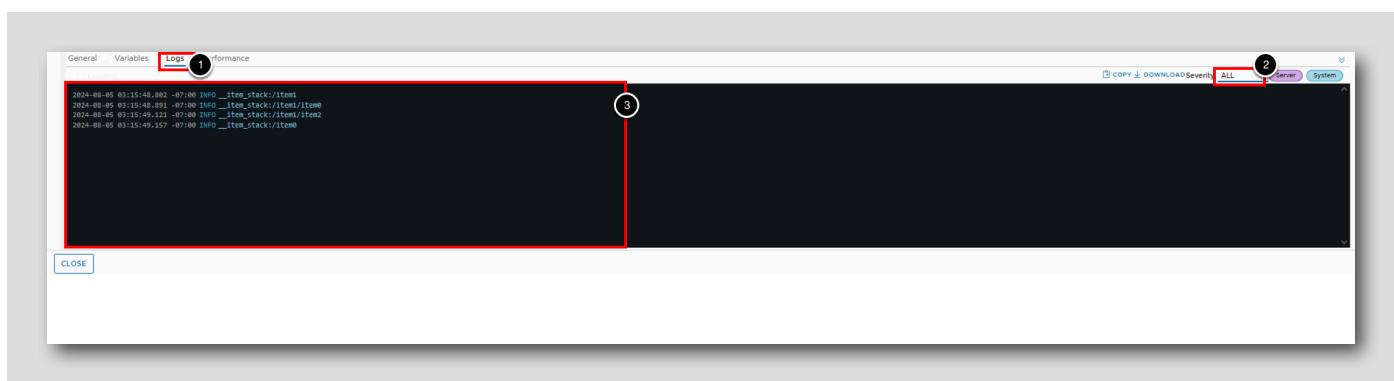
A red box highlights the "Completed" button and the "General" tab. A black circle with the number "1" is over the "Completed" button, and another black circle with the number "2" is over the "General" tab. A third black circle with the number "3" is over the "288 ms" label.

After a workflow has been run, the view is automatically updated to show the workflow token view.

1. Observe the current **status** of the workflow is shown as Completed. Other possible statuses include in Progress and Failed.
2. There are details regarding the **start date** and **end date** of the workflow run, the workflow **status** and the **user account** who ran the workflow.
3. For each workflow element, there are details of their duration
  - Creating the folder took 288ms in this example

## Look at the Logs

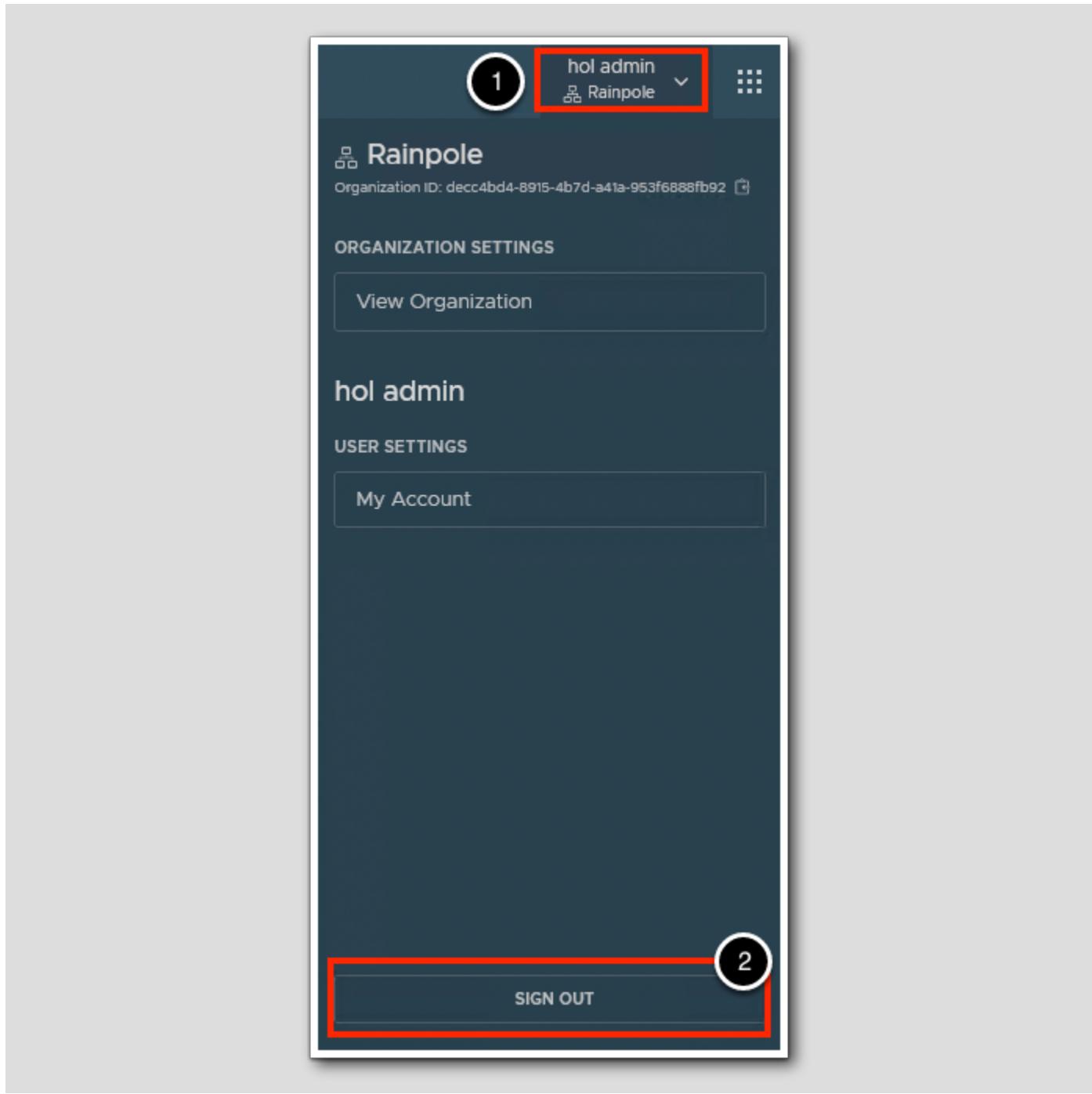
[67]



Each time a workflow is run it generates log entries. We can review the contents of the logs, including any custom logging added by the workflow creator using the Logs tab. Let's explore the logs from our workflow.

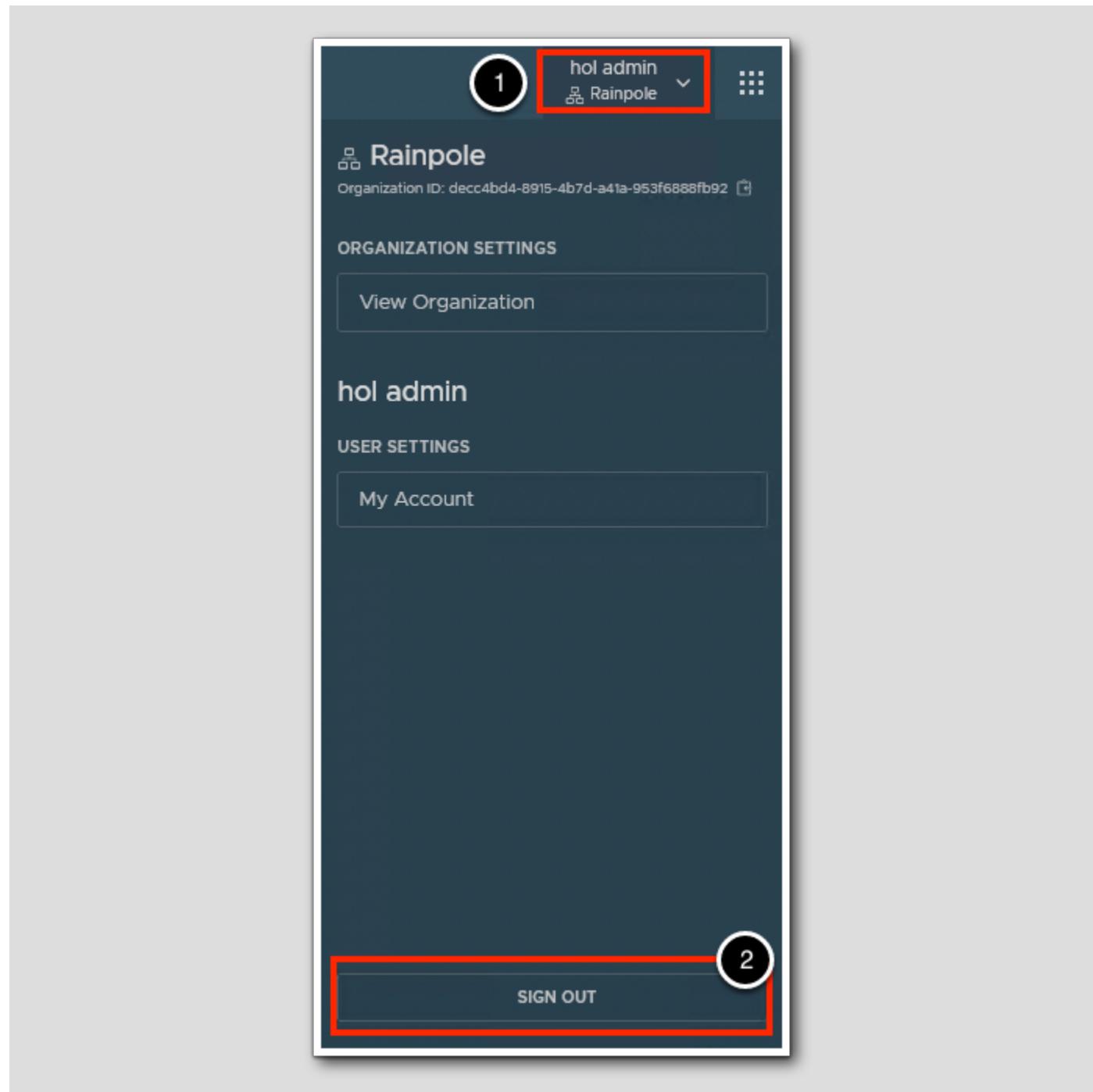
1. Click the **Logs** tab on the lower right hand side of the window
2. Make sure Severity **ALL** is selected. This displays all log entries regardless of the severity level assigned to them. Examples of severity levels include DEBUG, INFO, WARNING and ERROR. Workflow developer can use these different severity levels to pass key updates into the logs during a workflow run.
3. The logs window displays the log output from the workflow. In our example the workflow logs are very basic, we did not include any custom logging commands into our workflow. As we develop workflows we can add custom logging commands to print key progress messages, or variable values into the logs to aid with troubleshooting or for audit purposes. For example, if a workflow needs a long time to complete a developer can add INFO logging commands into the workflow to print key status updates as the workflow progresses. Or if they are troubleshooting an issue where an unexpected value is being reported at the end of a workflow, they can use DEBUG level logging commands to print information to the logs to show the value of certain attributes at different points in the workflow.

## Log Out of Aria Automation Orchestrator



To Log out of Aria Automation Orchestrator:

1. Click **hol admin** to open the organization menu.
2. Click **SIGN OUT**.



## Conclusion

[69]

An Aria Automation workflow is very easy to create, and it can be run once or multiple times. A workflow contains parameters (inputs, outputs, attributes) that can be bound to workflow elements. Each workflow run is referred to as a workflow token.

In couple of minutes, it is very easy to create a meaningful process that can be used by other, possibly less technical people through a web frontend.

Discover more about workflows in the next module.

## You've finished the module

[70]

Congratulations on completing the lab module.

If you are looking for additional information and key concepts of workflows, visit the VMware documentation on [Developing Workflows with VMware Aria Automation](#).

From here you can:

1. Continue with the next lab module
2. Click [vlp:table-of-contents] Show Table of Contents] to jump to any module or lesson in this lab
3. End your lab and return in the future

## Module 3 -Understand Parameters, Attributes and Scripting Objects (30 minutes) Basic

### Introduction

[72]

In this module we cover the basics to make a workflow, starting with inputs and outputs, then looking at attributes, and finally scripting objects provided by Aria Automation Orchestrator plugins.

We will look at how all these constructs are interconnected, how values and objects flow between the various elements in a workflow, and how they are used to do real work!

Lab Captain(s):

- Katherine Skilling, Senior Architect, United Kingdom

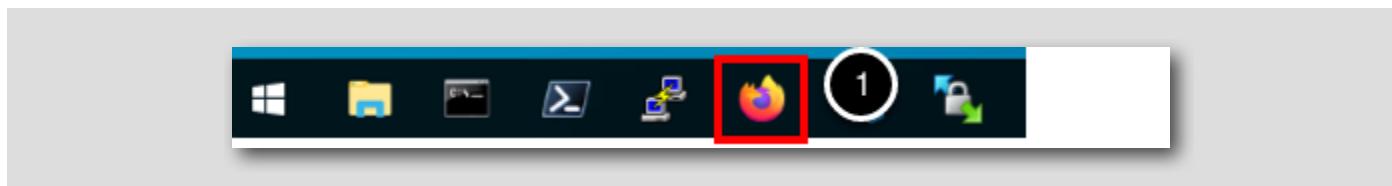
### Log in to Aria Automation Orchestrator

[73]

In the following few pages, we will log in to Aria Automation Orchestrator.

### Open the Firefox Browser from Windows Quick Launch Task Bar

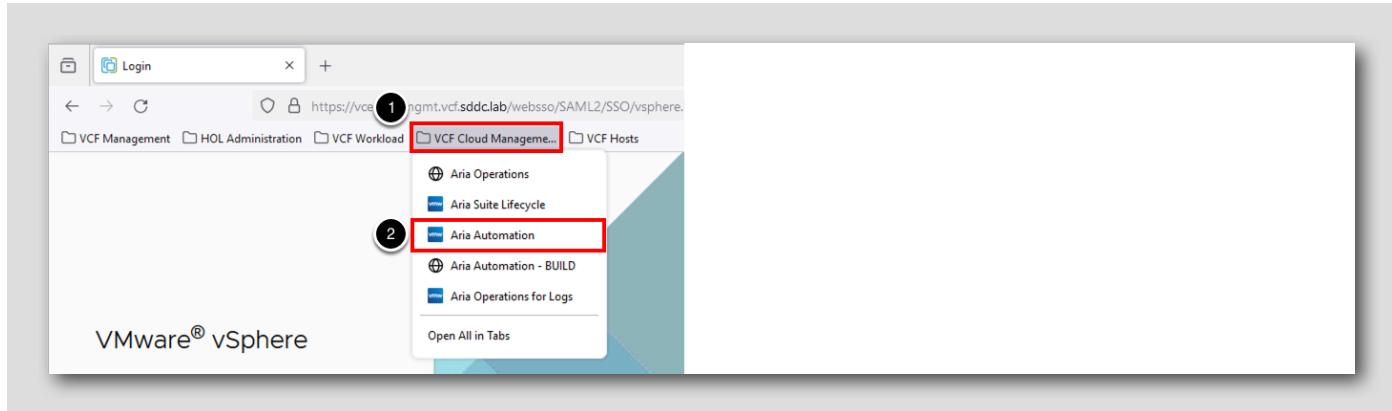
[74]



If the browser is not already open, launch Firefox.

1. Click the Firefox icon on the Windows Quick Launch Task Bar.

## Log in to Aria Automation

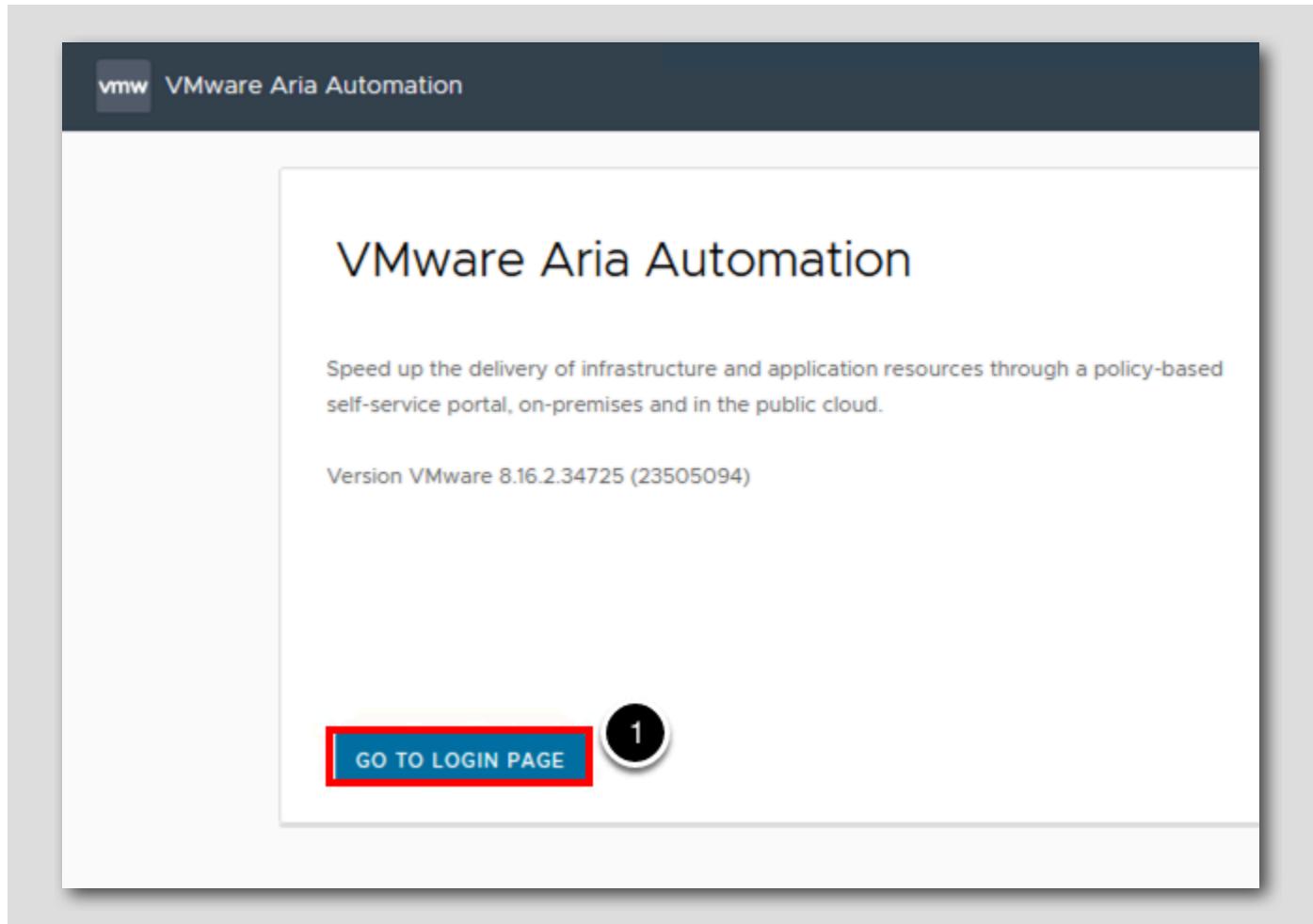


Once Firefox has loaded:

1. Click the VCF Cloud Management bookmark folder
2. Click Aria Automation.

## Redirect to Workspace ONE Access for Sign-On

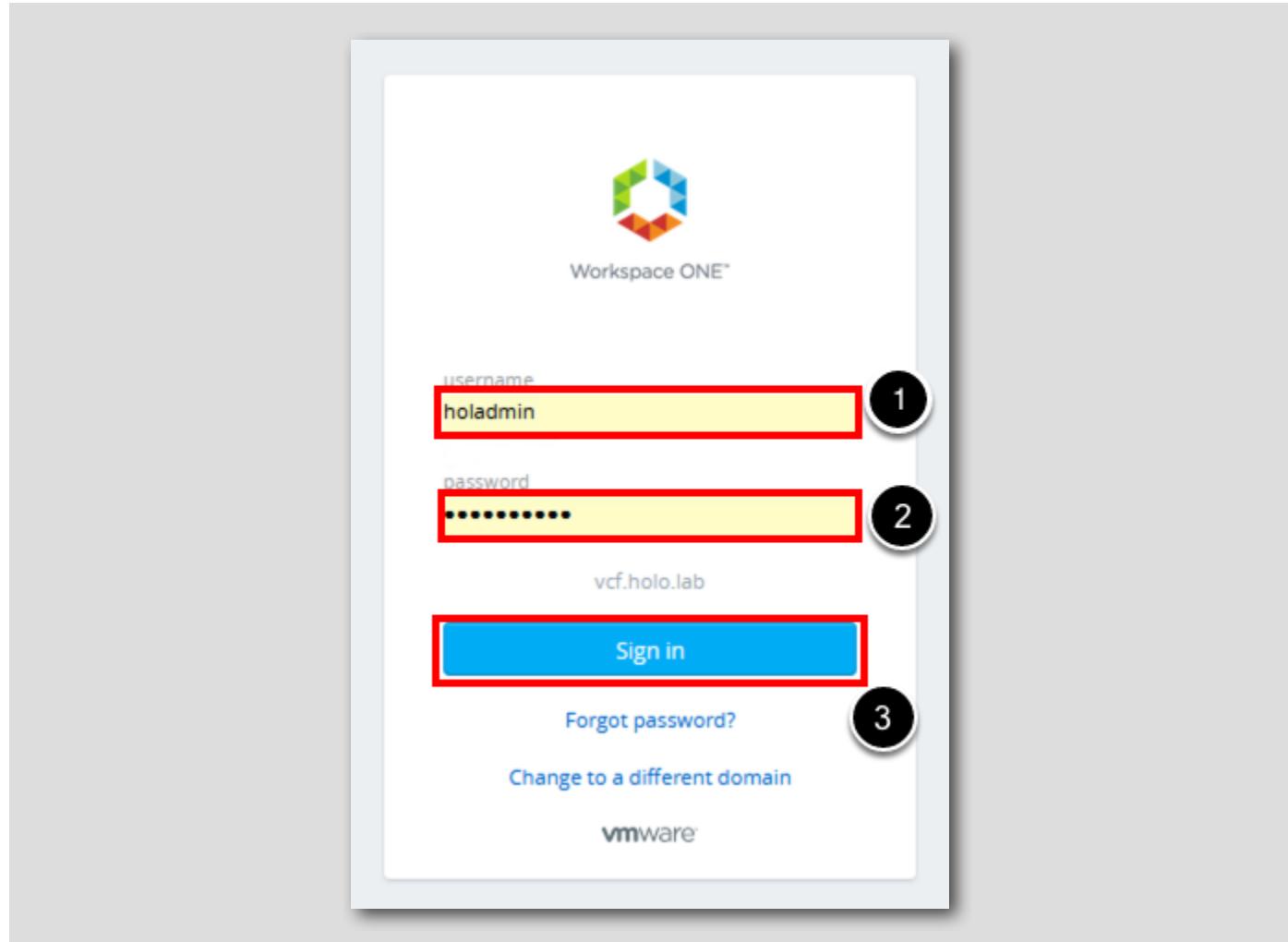
[76]



Aria Automation is integrated with Workspace ONE Access (aka VMware Identity Manager) and we need to redirect to the Workspace ONE Access login page to complete our log in progress.

1. At the VMware Aria Automation page, click GO TO LOGIN PAGE.

## Workspace ONE Access Login

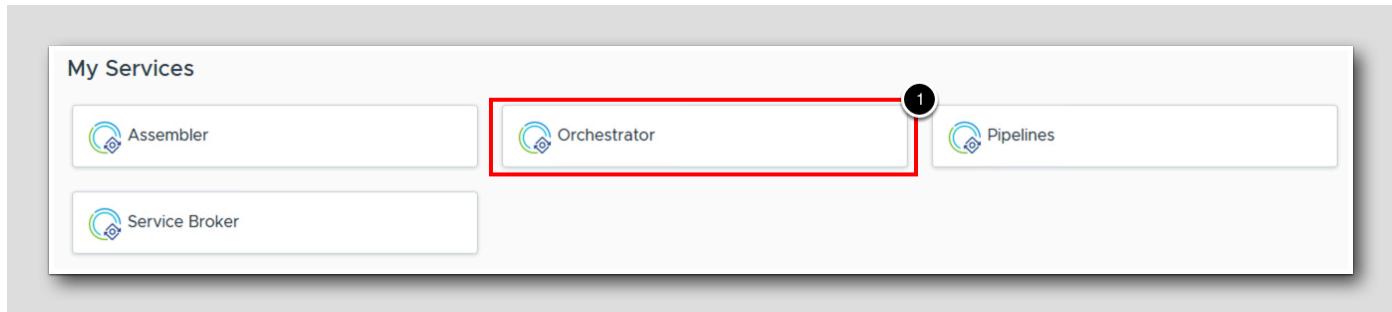


The credentials for **holadmin** should already be cached in the browser window.

At the **Workspace ONE Access** prompt, type in the following user and password information.

1. At the **username** field, type **holadmin**.
2. At the **password** field, type **VMware123!**.
3. Click **Sign in**.

## Launch the Orchestrator Service



From within the Cloud Services Console, under **My Services**:

1. Click the Orchestrator service.

## Understanding Input and Output Parameters

Input and output parameters are essential to understand in Aria Automation Orchestrator, as they are used in nearly every available workflow.

Parameters within Aria Automation Orchestrator are basically input and output variables of a workflow in the simplest of terms.

**Input parameters** are needed for most workflows and are values that are passed into the workflow when it starts via a user, application, another workflow, or even an action. A good example of this might be a workflow that is used to snapshot a virtual machine. An input parameter for such a workflow may be the virtual machine name. Another thing to note about input parameters is that they are read-only, and cannot be changed during workflow execution.

**Output parameters** represent the result of the workflow and can be passed on to other elements when the workflow has completed. For that same virtual machine snapshot workflow, the output parameter is the resulting snapshot. Output parameters are write-only; to read the value of a parameter within a workflow, it cannot be an output parameter. However, this output parameter value is readable for any workflows chained after this one.

In this lesson we will take an introductory look at the parameters available on a workflow that is provided out of the box with the default Orchestrator installation.

## Search for a workflow

The screenshot shows the VMware Aria Automation Orchestrator interface. The left sidebar has a 'Workflows' tab selected (1). The main area is titled 'Workflows' (4 of 642) and shows three workflow cards. The first card is '(Deprecated) Add and Set default vCenter instance' (3). The second card is 'Add a vCenter Server instance'. The third card is 'Update a vCenter Server instance'. A red box highlights the search bar at the top, which contains 'Any : add', 'Any : vcenter', and 'Any : instance' (3). A red box also highlights the 'Card View' icon in the top right corner (2). A red box highlights the 'DETAILS' button in the bottom right of the first workflow card (4).

1. Select the **Workflows** tab
2. Make sure the Card View is selected
3. Click on the Search field,
  - Type **add**, and press return
  - Type **vcenter** and press return
  - Type **instance** and press return
4. In the tile named **Add a vCenter Server instance**, click the **DETAILS** button

## Identify Parameters

| Name           | Type         | Direction | Description   | References   |
|----------------|--------------|-----------|---|--|
| enabled        | boolean      | Input     | Enables this vCenter Server instance for orchestration  | Register vCenter Server instance,Validate  |
| host           | string       | Input     | IP or host name of the vCenter Server instance to add   | Register vCenter Server instance,combine host and port and generate vSphere services list,Validate |
| port           | number       | Input     | HTTPS port of the vCenter Server instance   | Register vCenter Server instance,combine host and port and generate vSphere services list,Validate |
| path           | string       | Input     | Location of the SDK to use to connect to the vCenter Server instance  | Register vCenter Server instance,Validate  |
| sessionPerUser | boolean      | Input     | Specifies the method to manage user access to the vCenter Server system. If this is set to false, the method is share a unique session. | Register vCenter Server instance   |
| userName       | string       | Input     | User name of the user that Orchestrator will use to connect to the vCenter Server instance.   | Register vCenter Server instance,Validate  |
| password       | SecureString | Input     | Password of the user that Orchestrator will use to connect to the vCenter Server instance.  | Register vCenter Server instance,Validate  |
| domain         | string       | Input     | Domain name. This is used only if method session per user is set  | Register vCenter Server instance   |

Let's take some time to explore parameters in a bit more detail.

1. Select the **Inputs/Outputs** tab

Here are the input and outputs parameters for a workflow used to add vCenter objects to the Aria Automation Orchestrator inventory. Once a vCenter object is added, it is possible to run other workflows against it; for example, a workflow to clone a virtual machine.

On this tab we can observe the different input parameters being gathered.

2. There is a **Type** column, which dictates the type of value the input parameter will accept

3. For example, the **password** input parameter type is **SecureString**, which allows Aria Automation Orchestrator to take in a string in a secure way since it is a password, and its value should not be shown in clear text.

The recommended approach is to define input parameter names in a format called **camelCase**. **camelCase** is where the first word in the name starts with a lowercase letter and all proceeding ones start with a capital letter.

4. A good example of the **camelCase** formatting for the parameter names would be the parameter **sessionPerUser**

The **camelCase** format is the standard used for parameters in Aria Automation Orchestrator and what we will be using throughout the lab.

Each parameter has a type.

This type can be simple type like string, number, array. Type can also be complex type such as VC:VirtualMachine, AD:User or REST:RESTHost. All of these complex types come from plugins that have been added to Aria Automation Orchestrator.

When an input is chosen, Aria Automation Orchestrator will only allow objects from that input's type to be selected in order to avoid user mistakes. A complex object can carry a lot of properties that can be accessed directly from within a scriptable task, which makes it very flexible.

The fact that Aria Automation Orchestrator accounts for all variable types simplifies development, and makes it more reliable.

## Understanding Workflow Variables

[82]

Workflow variables are one of the key constructs used to carry values between elements in a workflow.

Variables are generally read/write, and they can be used to transfer data between the elements of a workflow. A variable can have a default value at the beginning of the workflow (similar to hardcoding value in a script) or the value can be set during the workflow execution by one of the workflow element.

In this lesson we will review some of the variables used within a workflow, editing their bindings and then running a workflow to see the impact of our changes.

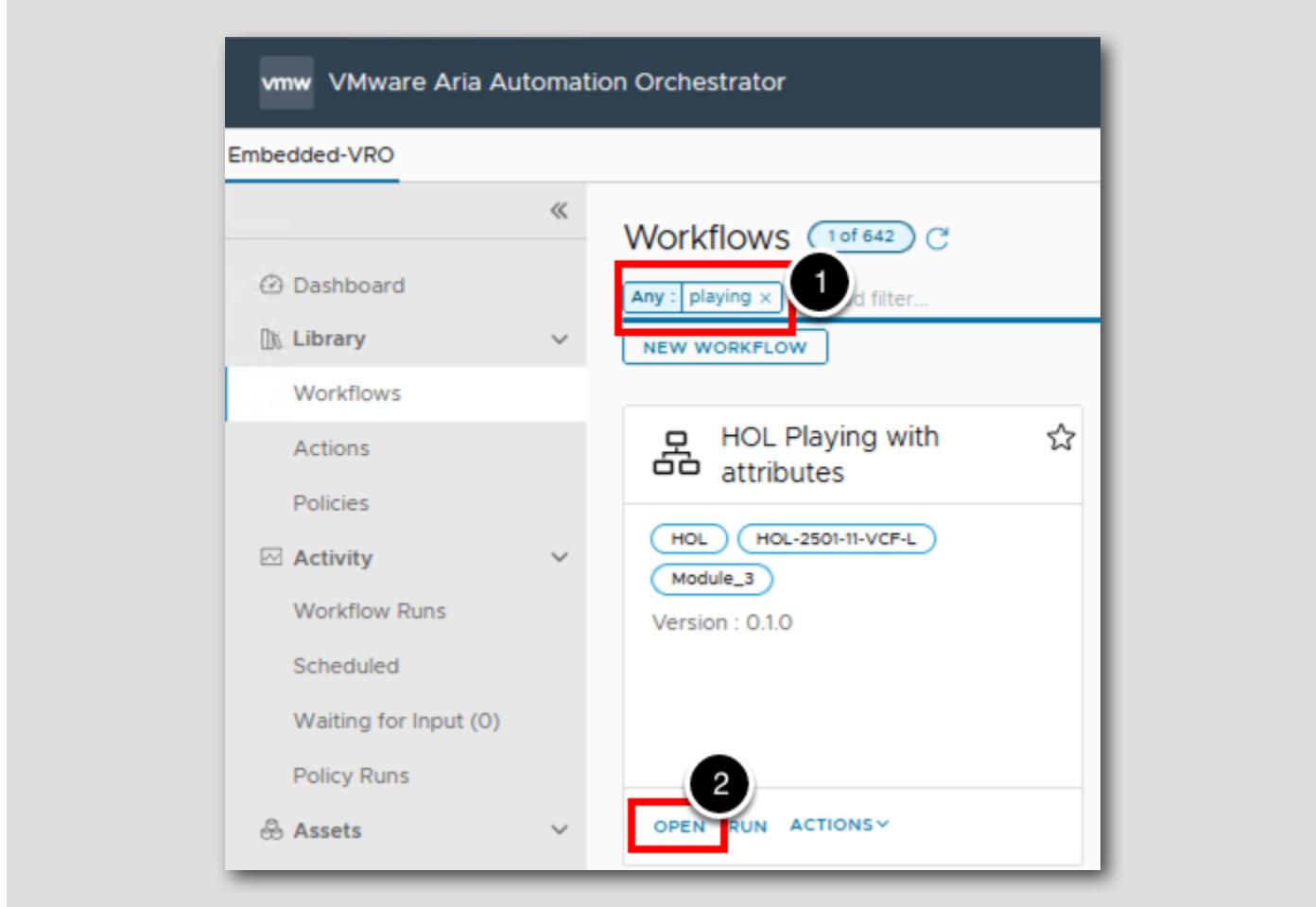
## Remove the workflow filters

[83]

The screenshot shows the VMware Aria Automation Orchestrator web interface. The top navigation bar includes the VMW logo, the title "VMware Aria Automation Orchestrator", a help icon, the user name "hol admin", and a Rainpole icon. The left sidebar has a tree view with nodes like "Dashboard", "Library" (which is expanded), "Workflows" (highlighted with a red box and numbered 1), "Actions", "Policies", "Activity" (expanded), "Workflow Runs", "Scheduled", "Waiting for Input (0)", "Policy Runs", and "Assets". The main content area is titled "Workflows 4 of 642". It features a search bar with filters: "Any : add x", "Any : vcenter x", "Any : instance x", "Add filter...", and a "NEW WORKFLOW" button. To the right of the search bar is a red box with a circled "2" and an "X" button. Below the search bar are three workflow cards: "Add and Set default vCenter wit..." (disabled), "Add a vCenter Server instance", and "Update a vCenter Server instance". Each card has "DETAILS", "RUN", and "ACTIONSV" buttons at the bottom.

1. Let's start by removing the filters we applied in a previous lesson. Select the **Workflows** tab
2. Click the X button to remove all filters

## Identify attributes



We are going to use a workflow created especially for this lesson, let's locate it in the Orchestrator inventory.

1. Click on the Search field,  
Type **playing** and press return
- 2.In the first tile named **HOL Playing with attributes**, click the **OPEN** button

## Remove variables

| Name         | Type            |
|--------------|-----------------|
| attribute1   | string          |
| attribute2   | string          |
| resourcePool | VC:ResourcePool |

1. Select the **Variables** tab

Here the variables assigned to the workflow are listed. They can be blank or have a default value, and like input/output parameters, they have a type. We are going to remove the **resourcePool** attribute from the workflow as it is no longer needed.

2. Select the variable **resourcePool** by checking its checkbox
3. Click the **DELETE** button

## Edit attributes

The screenshot shows the VMware Aria Automation Orchestrator interface. At the top, it says "VMware Aria Automation Orchestrator" and "Embedded-VRO". Below that, the workflow name is "HOL Playing with attributes". The "Variables" tab is selected. The table lists two variables:

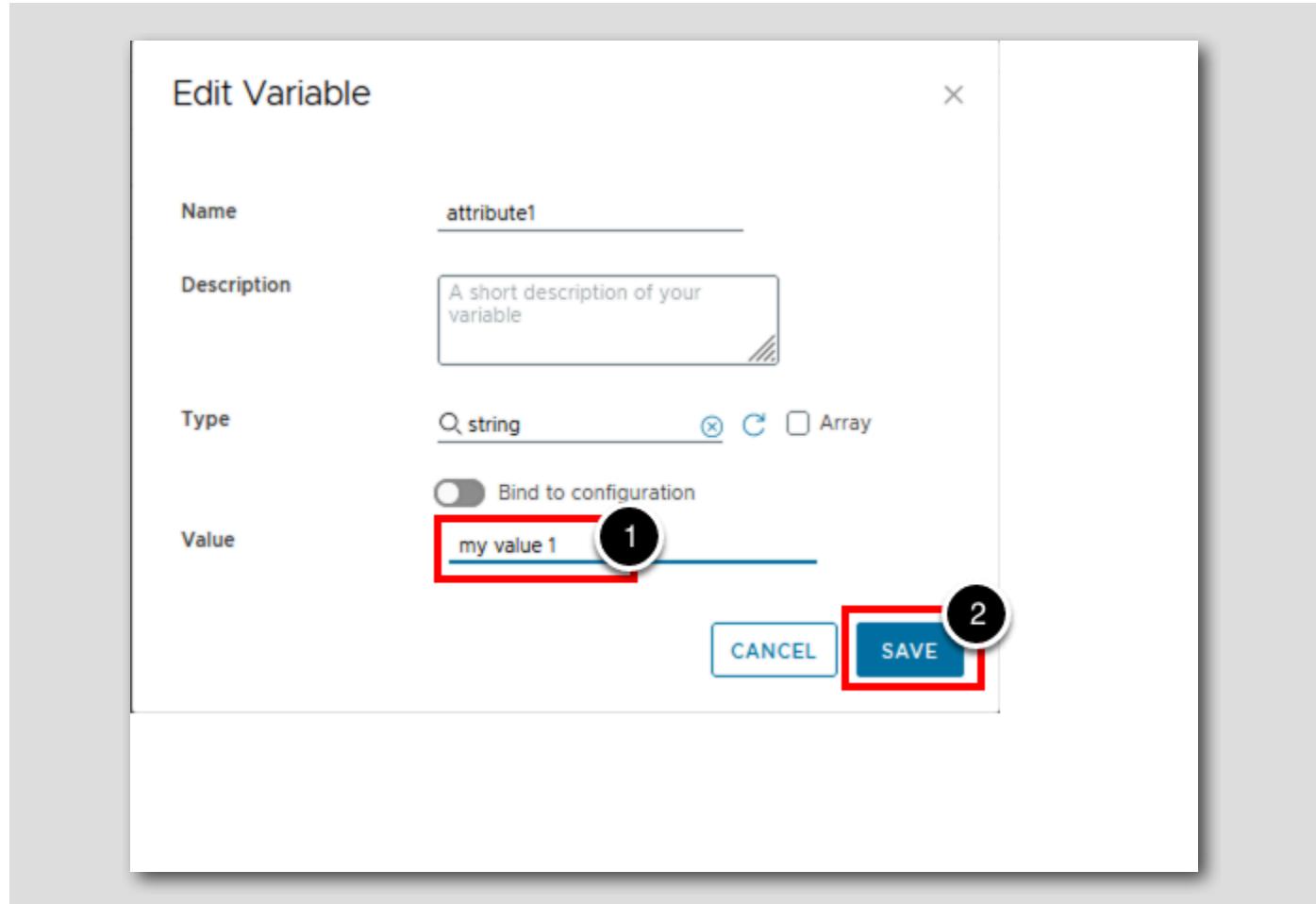
| Name       | Type   |
|------------|--------|
| attribute1 | string |
| attribute2 | string |

A red box highlights the "attribute1" row, and a circled "1" is placed over the "attribute1" column header. There are buttons at the top for "NEW", "DELETE", "COPY", "PASTE", and "MOVE TO".

Next we are going to look at changing the value assigned to an attribute, to give it a default value.

1. Click the variable **attribute1**

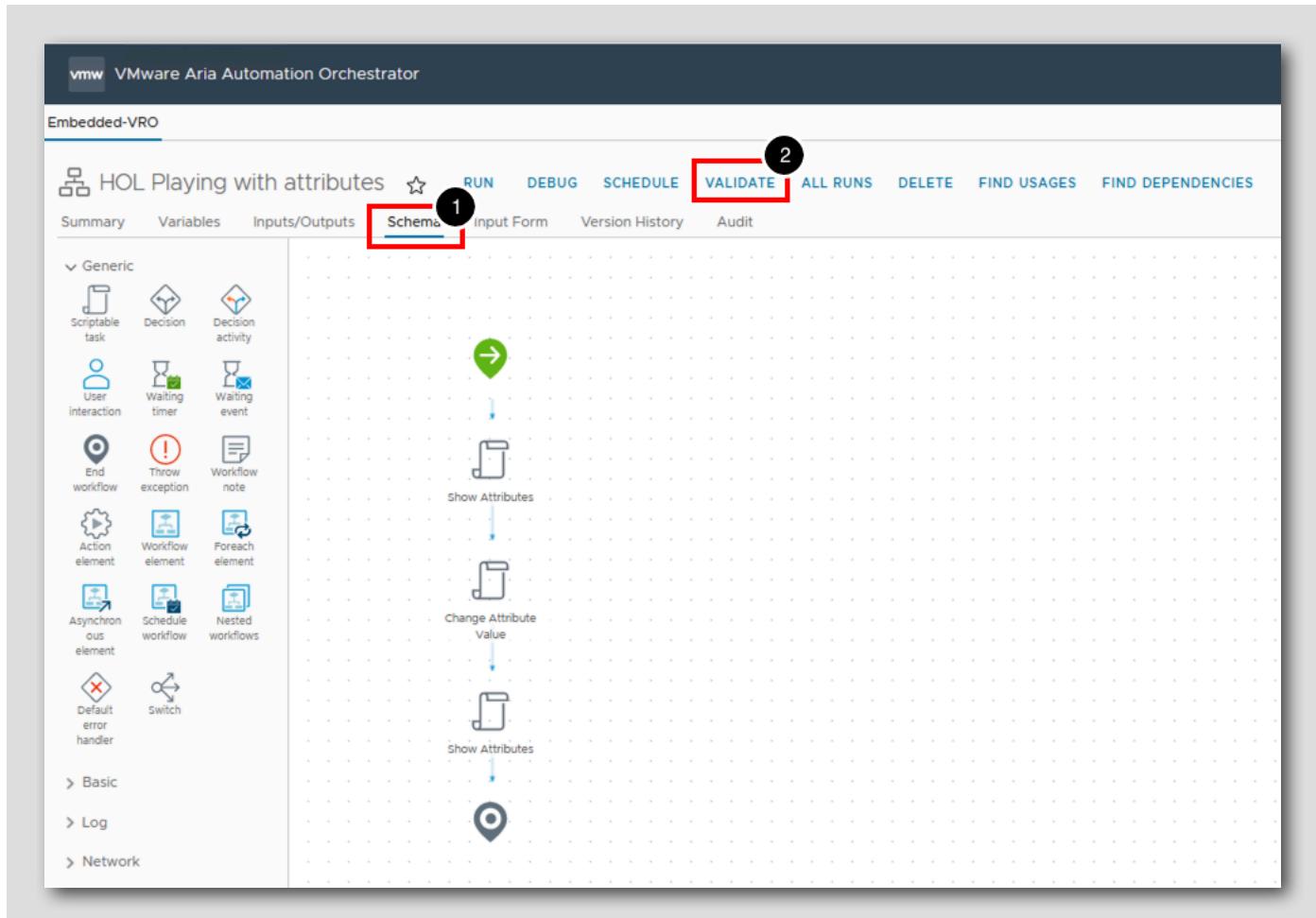
## Assign default value



1. Enter for the Value: my value 1
2. Click the SAVE button
- 3.(Not shown) Repeat this process to set the value of attribute2 to locked value

This has set the attribute attribute1's default value to **my value 1** and attribute2's to **locked value**. They will remain read/write and can be used by any workflow element within this workflow as local Input and/or local Output.

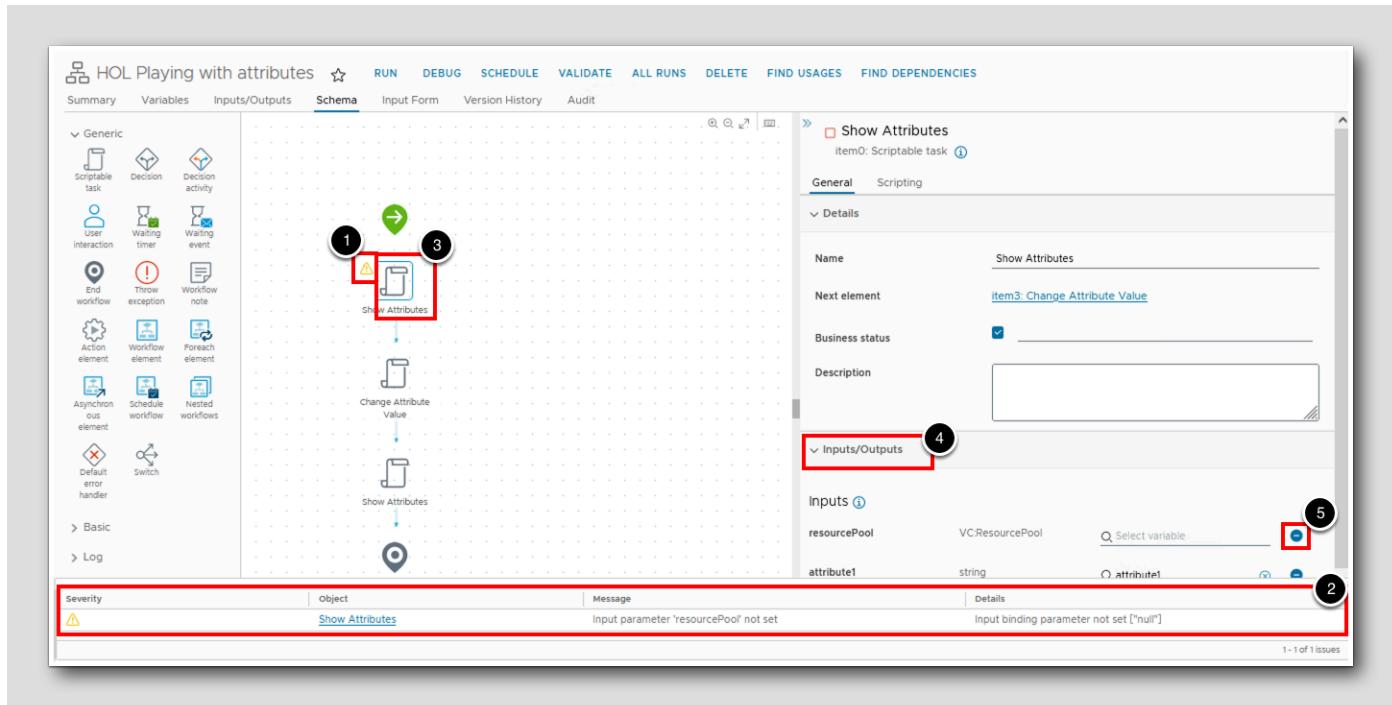
## Validate the workflow



Since we have made changes to our workflow, it is a good idea to validate the workflow to see if there are any issues as a result of our changes.

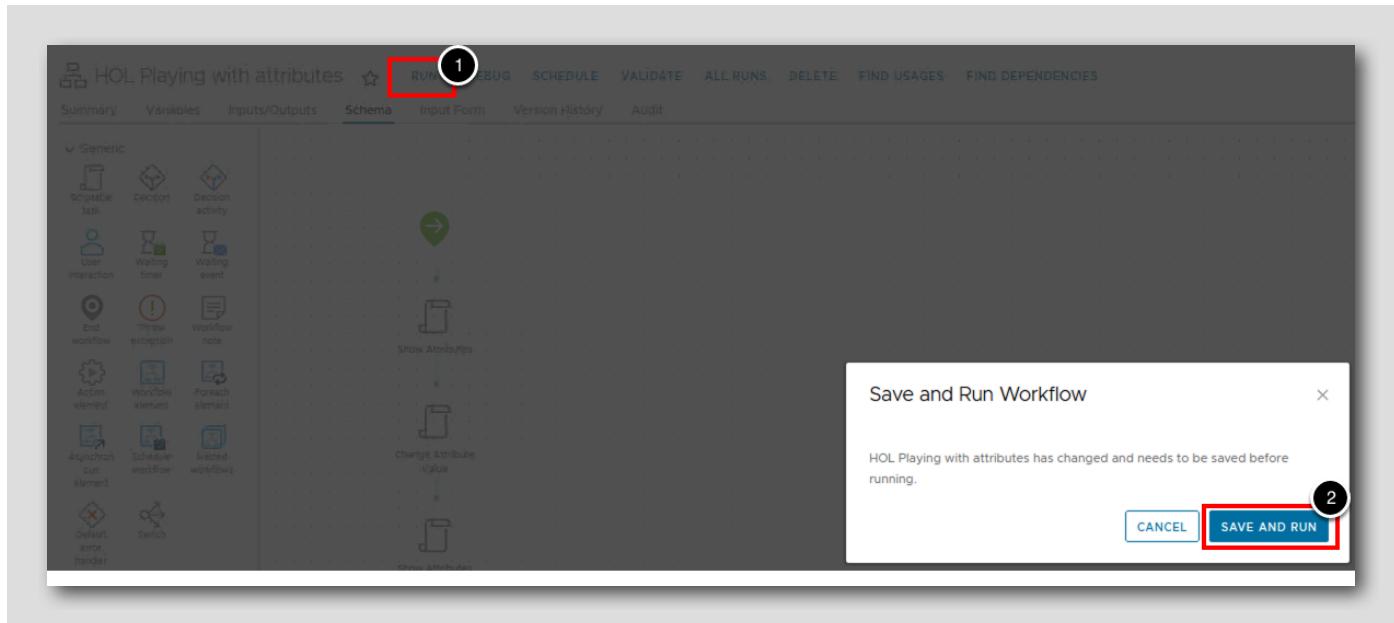
1. Click the Schema button
2. Click the VALIDATE button

## Validation results



1. After a few seconds the validation will complete and show a warning (yellow exclamation mark) against the Show Attributes scripting element.
2. A yellow severity message is also shown at the bottom of the screen with an explanation of the warning. This is expected as we have deleted the resourcePool variable that is linked to this scripting element. Let's tidy up the workflow and clear the validation warning.
3. Click on the Show Attributes scripting element
4. Expand the Inputs/Outputs section
5. Click on the - sign to the right of the resourcePool input to delete it

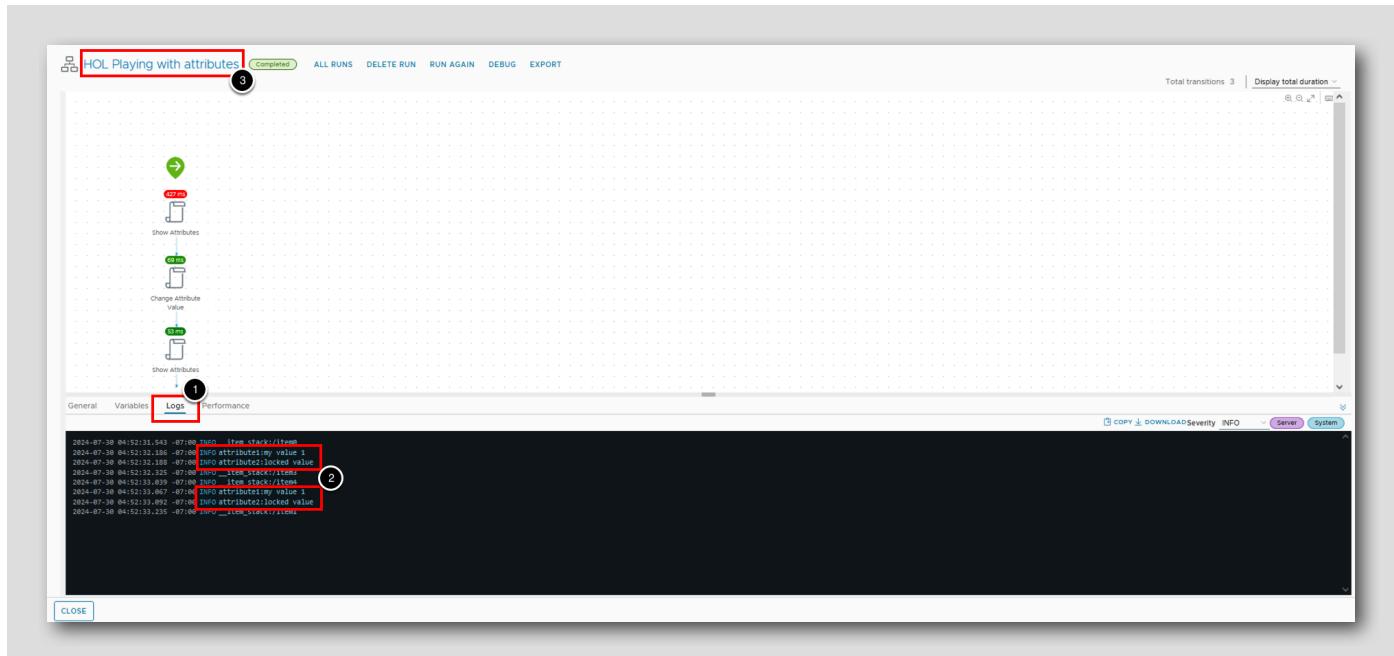
Look at the value of the Variables



With the validation warning resolved, and no validation errors we can proceed to run our workflow.

1. Click the RUN button
2. Click the SAVE AND RUN button

## Observe the workflow logs

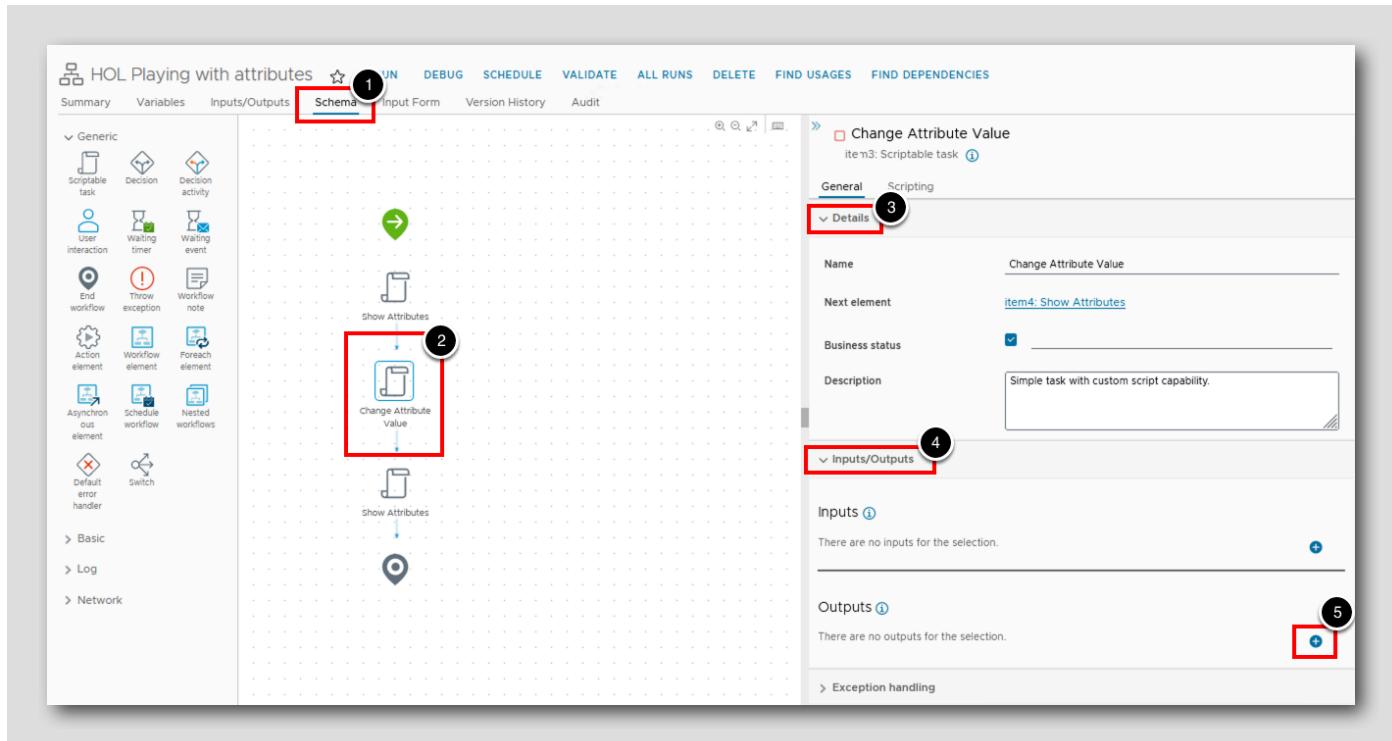


1. Once the workflow completes, we can look at the logs by clicking the Logs tab
2. The log output shows the attribute values twice, because the workflow contains two "show attribute" scriptable tasks that write the variable values to the system log.

Note: If necessary, you can click to the top of the Logs window and drag to expand it.

3. Click the workflow name to return to Edit mode

## Change the value of an attribute in a workflow element



While the workflow completed successfully, we simply printed the default values of our attributes to the logs. Now let's edit the workflow and amend the code so that it updates the value of one of the attributes when the workflow is run.

1. Select the Schema tab
2. Select the scriptable task change attribute value
3. Collapse the Details tab
4. Expand Inputs/Outputs tab
5. Under the Outputs Section, click the + button to bind an output

## Assign an output parameter

The screenshot shows the 'Inputs/Outputs' section of the VCF-L interface. The 'Outputs' tab is active, showing a list of two items: 'attribute1' and 'attribute2'. The 'attribute1' item is highlighted with a red box and has a black circle with the number '2' next to it. Below the list is a search bar labeled 'Select variable' with a red box around it and a black circle with the number '1' next to it.

We now need to assign the new output parameter to one of our attributes. This will pass the value assigned to the attribute out from the scripting element ready for it to be used within a later script as part of the workflow.

1. Click the Select variable field
2. Select attribute1 from the dropdown list

## Update the script

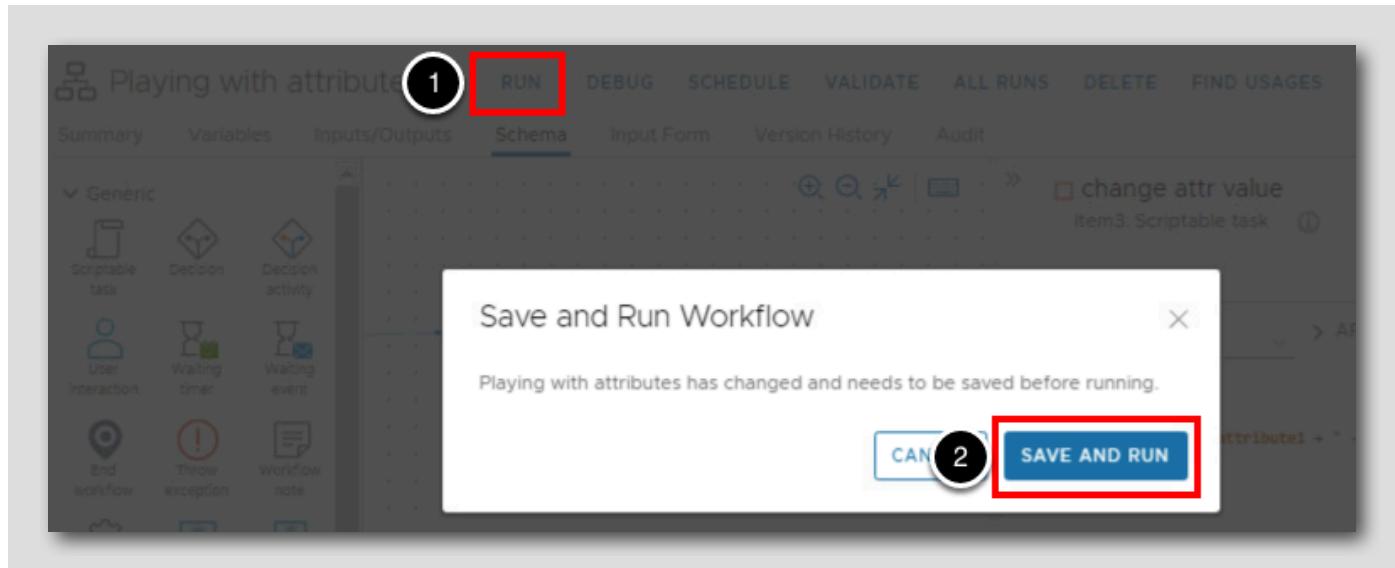


With the output parameter assigned to attribute1, we can now edit the code within our scriptable task to update the value assigned to attribute1 when the workflow runs. We are going to append a small amount of text to the end of the default value of attribute1.

1. Select the **Scripting** tab
2. Enter the following code, below the existing line:

```
attribute1 = attribute1 + " - my new value";
```

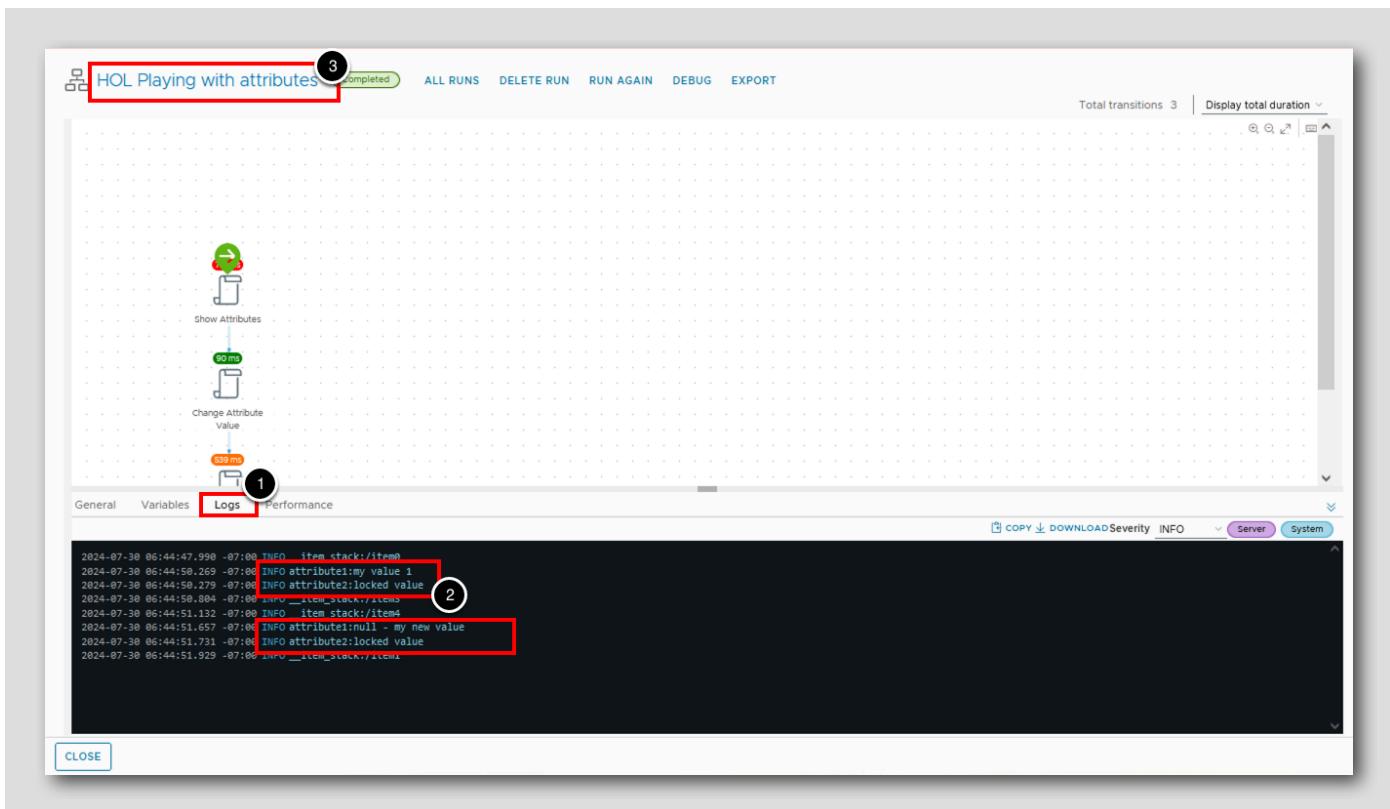
## Run the workflow



Let's save our changes and run the workflow to see how the amended code changes our output values.

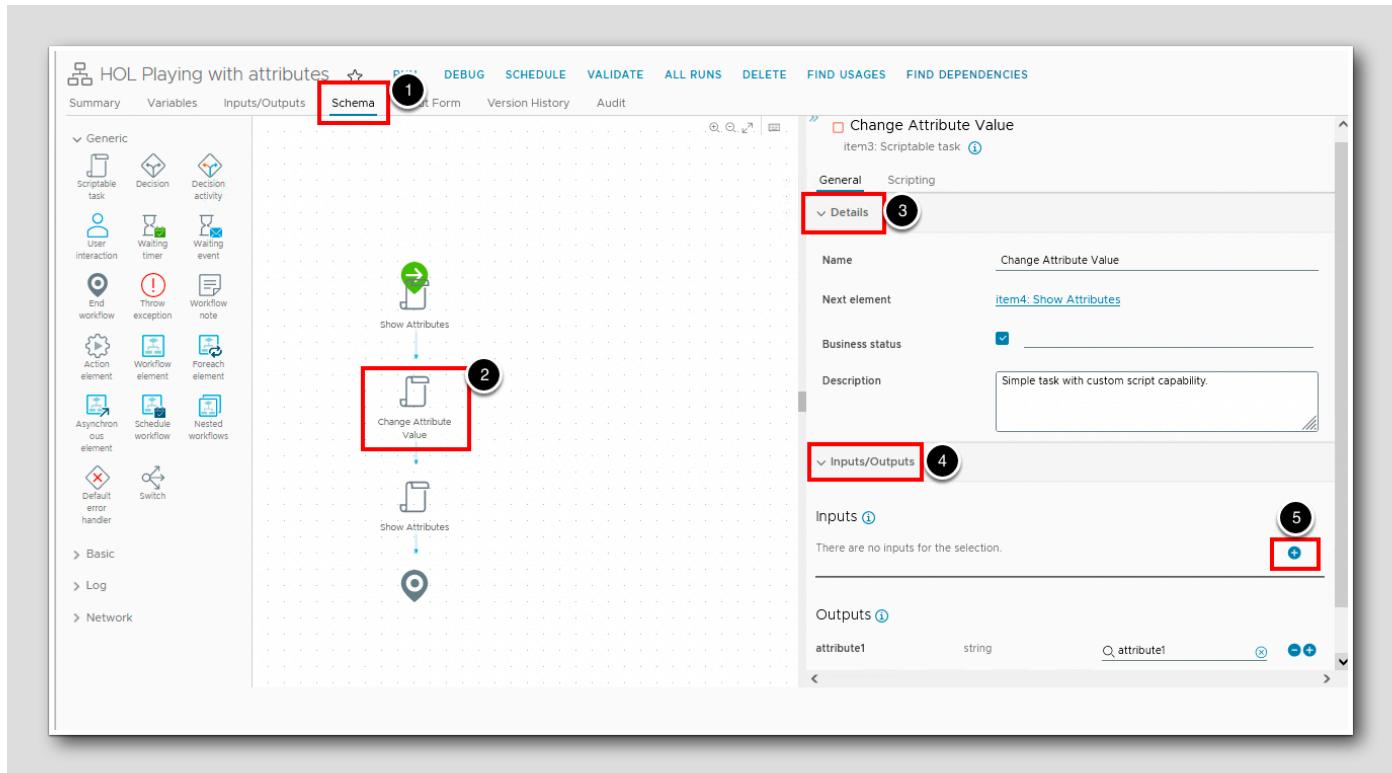
1. Click the RUN button
2. Click the SAVE AND RUN button

## Observe the logs



1. Once the workflow completes, look at the logs by clicking the Logs tab
2. Notice the second time the value of attribute1 is shown, "my new value" is visible but with null at the beginning.  
This does not seem to be the expected behavior - in the next steps we'll look at how to fix that.
3. Click the workflow name to return to Edit mode

## Make additional changes to the attributes in a workflow element



In the previous step, a null value was found in the logs. It happened because the variable **attribute1**, despite existing in the overall workflow (as a variable) and bound as output, has not been bound as input to this specific scriptable task "change attribute value". Let's resolve this so that the existing value of attribute1 is passed into the change attribute value scripting element.

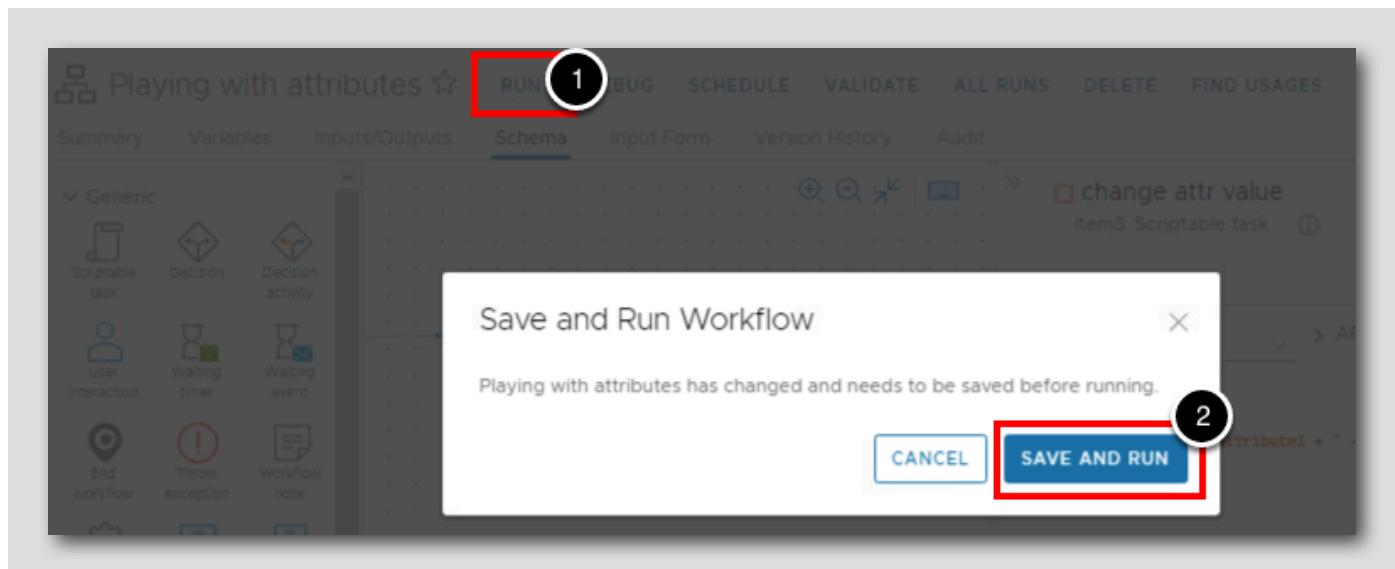
1. Select the Schema tab
2. Select the scriptable task change attribute value
3. Collapse the Details tab
4. Expand Inputs/Outputs tab
5. Under the Inputs Section, click the + button to bind an input.

## Assign an input parameter

The screenshot shows the configuration interface for a 'Change Attribute Value' scripting element. The 'Inputs' section is highlighted, showing a dropdown menu with 'attribute1' selected. A red box surrounds the 'attribute1' entry, and a black circle with the number '2' is placed over the 'attribute1' entry in the list. Another black circle with the number '1' is placed over the 'Select variable' field.

1. Click in the Select variable field
2. Select attribute1 from the list that appears. The attribute1 string variable will now be passed into the Change Attribute Value scripting element as an input, and out of the scripting element as an output. This allows its existing value to be read, updated and then shared with other scripting elements in the workflow.

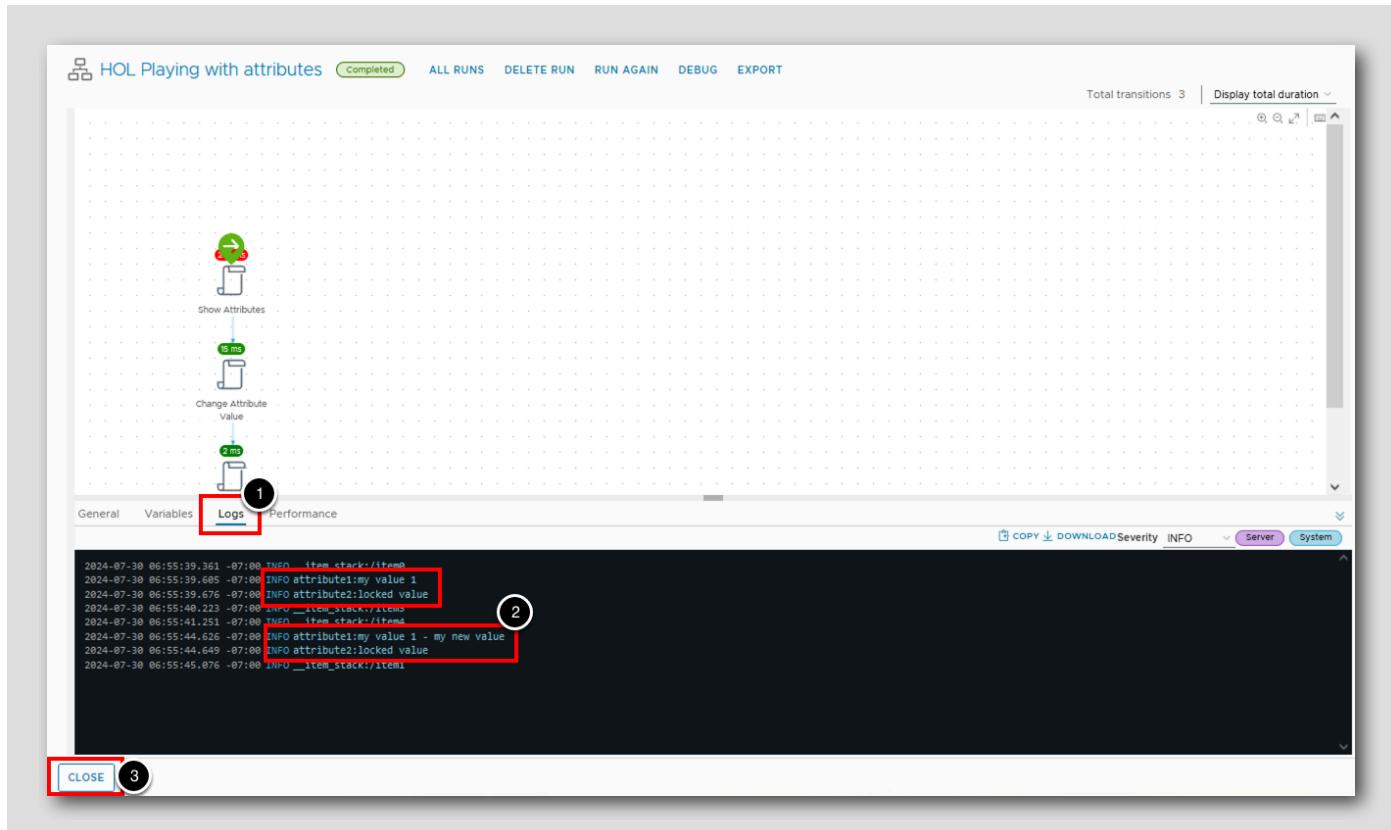
## Run the updated workflow



With our changes made, let's run our workflow again and see how the outcome has changed.

1. Click the RUN button
2. Click the SAVE AND RUN button

## Observe the logs



- Once the workflow completes, look at the logs by clicking the Logs tab
- Notice the second time the value of attribute1 is shown, "my new value 1" has been concatenated with " - my new value" to give a resultant value of my value 1 - my new value.

This is the expected result. Now that the `attribute1` variable has been added as an input, its value is available in the scriptable task and properly concatenated.

- Click the CLOSE button

## Using Scripting Objects

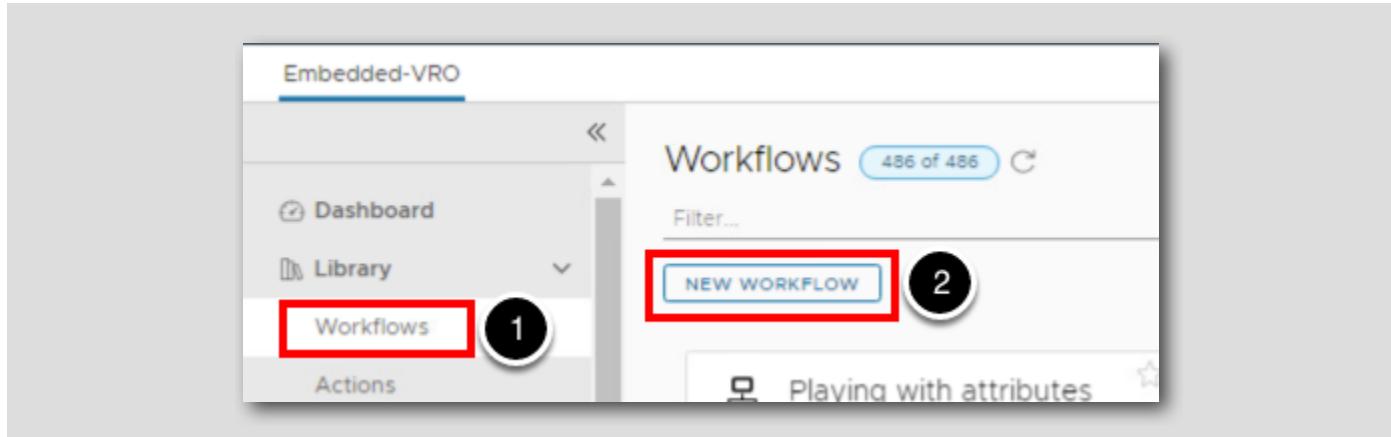
Scripting objects are JavaScript classes that can be used in any scriptable task and/or action. They can be browsed from the API explorer and are related to Aria Automation Orchestrator core components (like logs, workflows, etc.). They also come with plugins like VirtualMachine for vCenter, CloudTemplate for Aria Automation, etc.

Each of these JavaScript classes, like in any other object-oriented languages, has properties, constructors and methods associated with it.

To illustrate this, let's create a workflow to retrieve the IP address of a virtual machine (this information can be found in the VirtualMachine scripting object).

## Create a new workflow

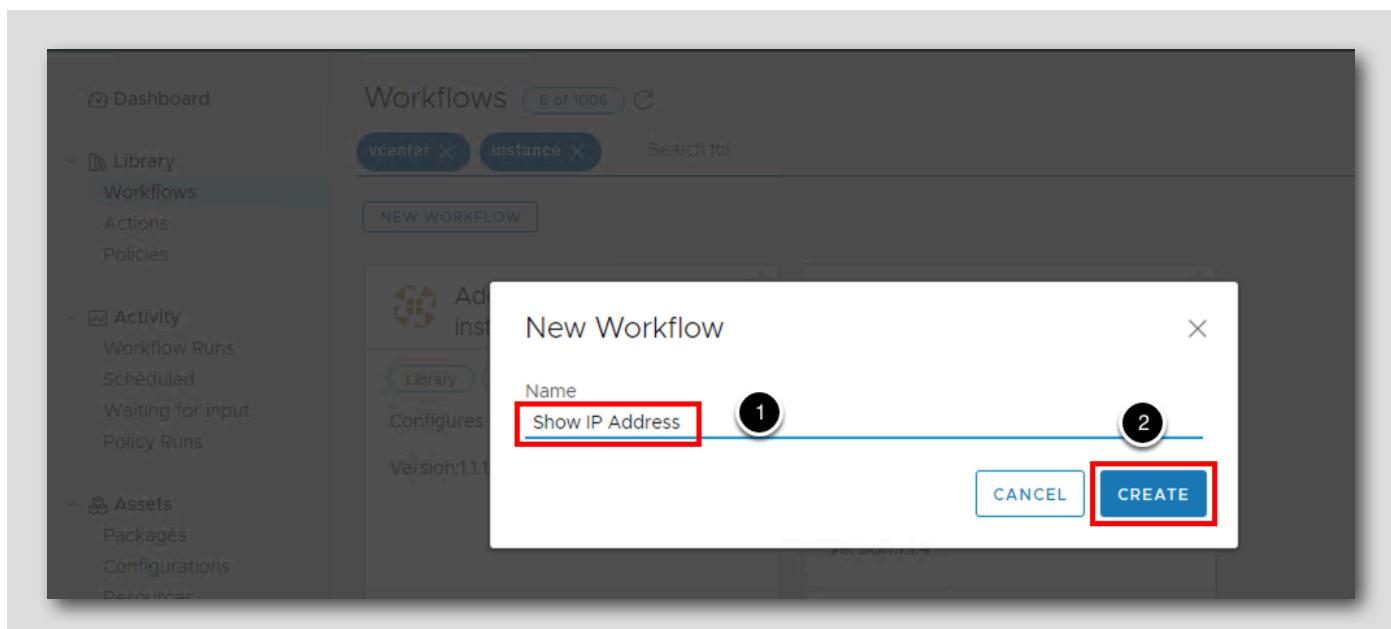
[102]



1. Select the **Workflows** tab
2. Click the **NEW WORKFLOW** button

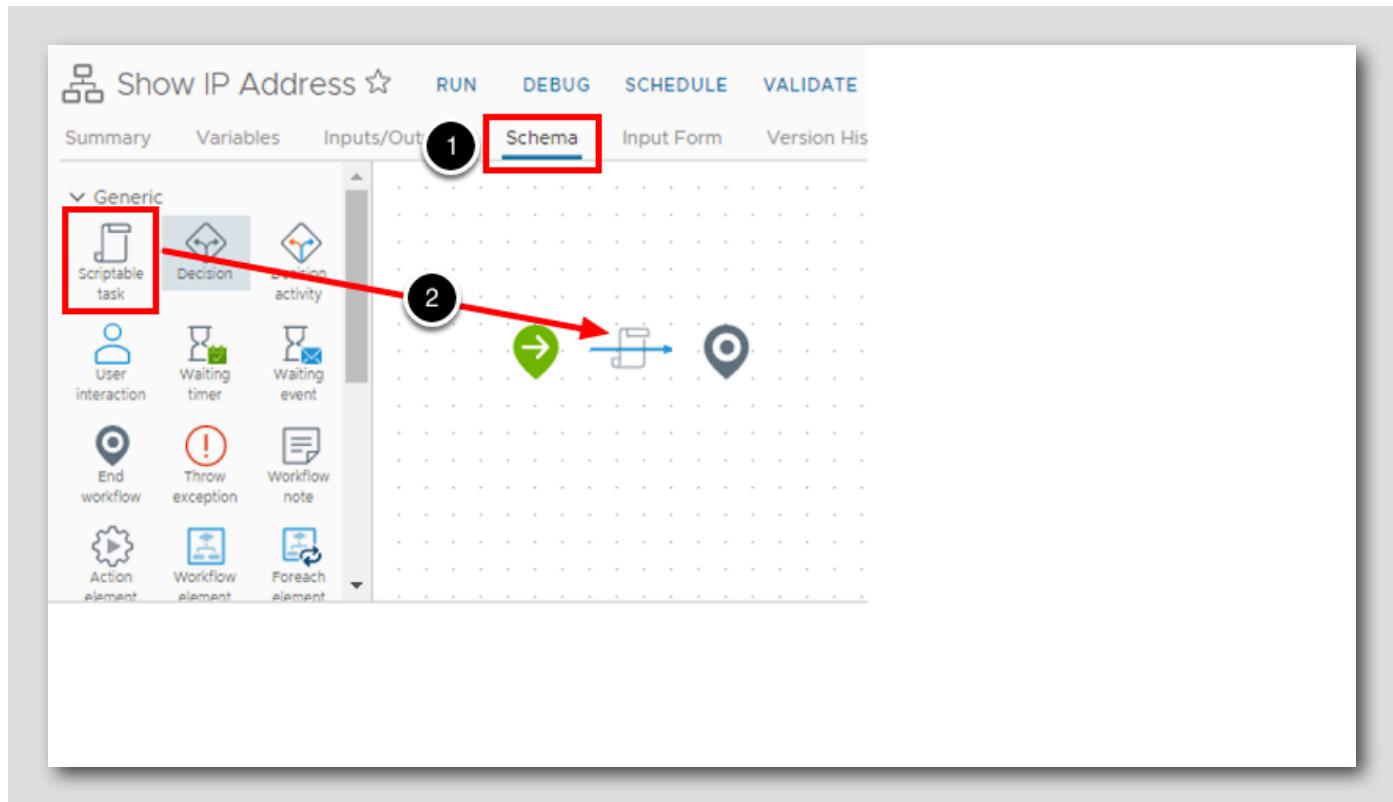
## Enter the workflow name

[103]



1. In the Name field, enter Show IP Address
2. Click the CREATE button

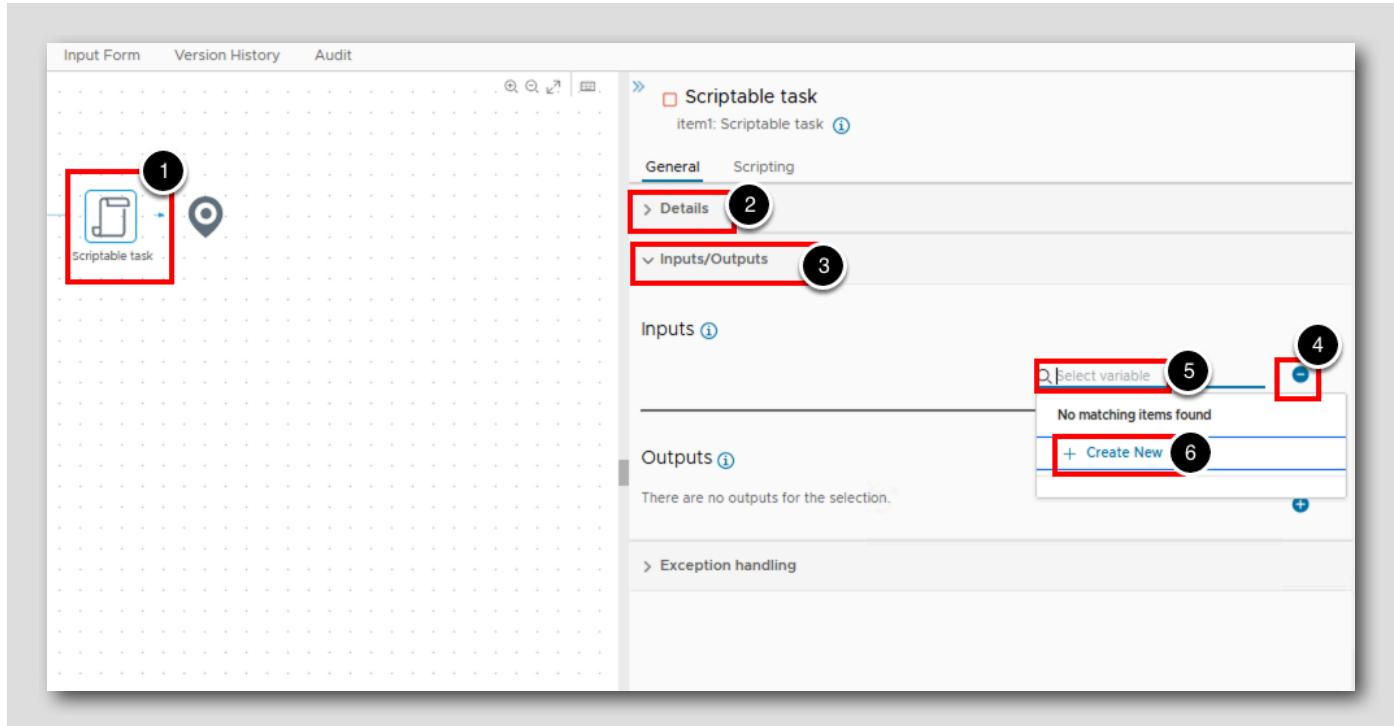
Add a scriptable task



Our workflow will start with some custom code, so we will add a scriptable task to the workflow schema.

1. Select the Schema tab
2. Drag the Scriptable task element into the schema

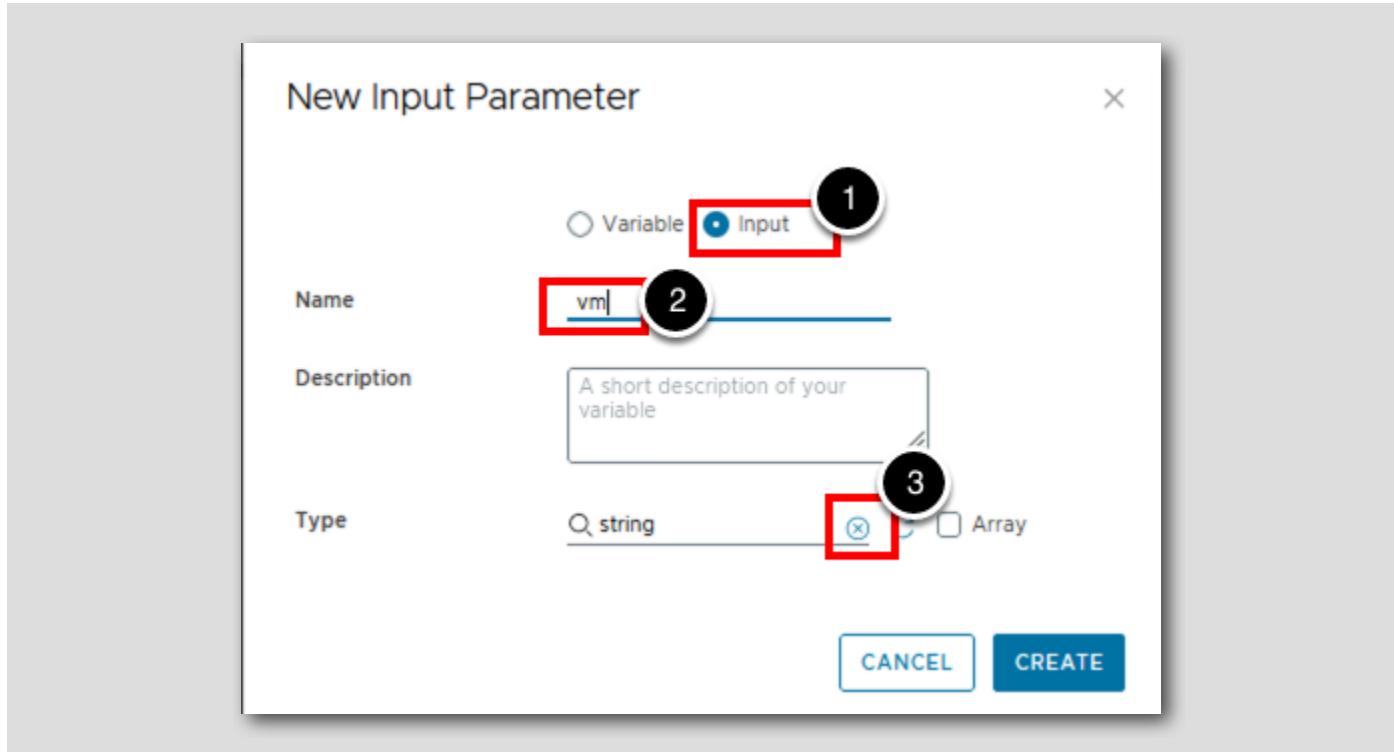
## Add an input to the scriptable task



We are going to need to collect some information from the user at the time the workflow runs, let's add an input parameter to the scriptable task.

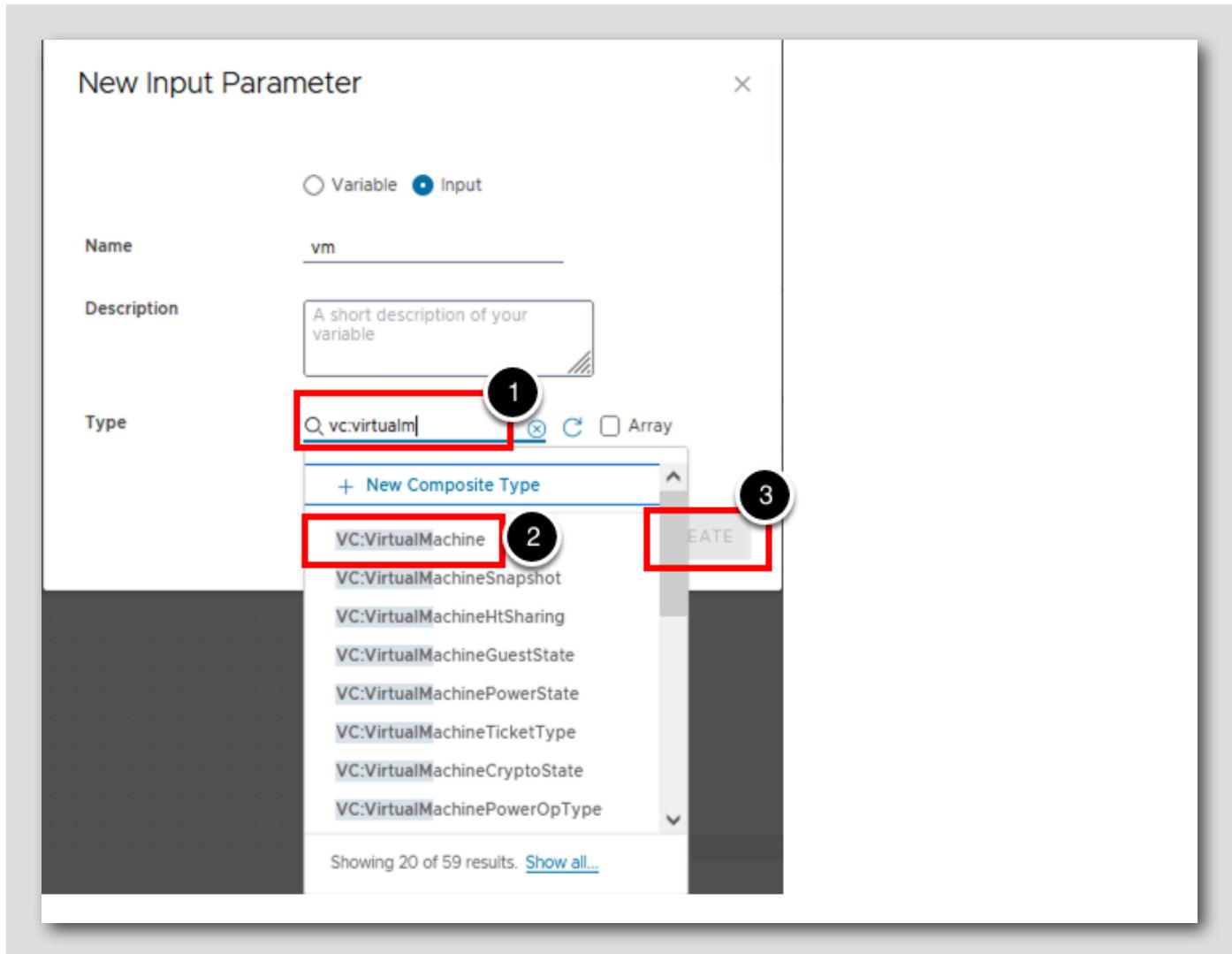
1. Select the **Scriptable task** element
2. Collapse the **Details** pane
3. Expand the **Inputs/Outputs** pane
4. Click the **+** button to add an **input**
5. Click the **Select variable** field
6. Click the **+ Create New** link

## Add a new input parameter



1. Select Input
2. Enter the Name vm
3. Remove the existing Type by clicking on the x icon

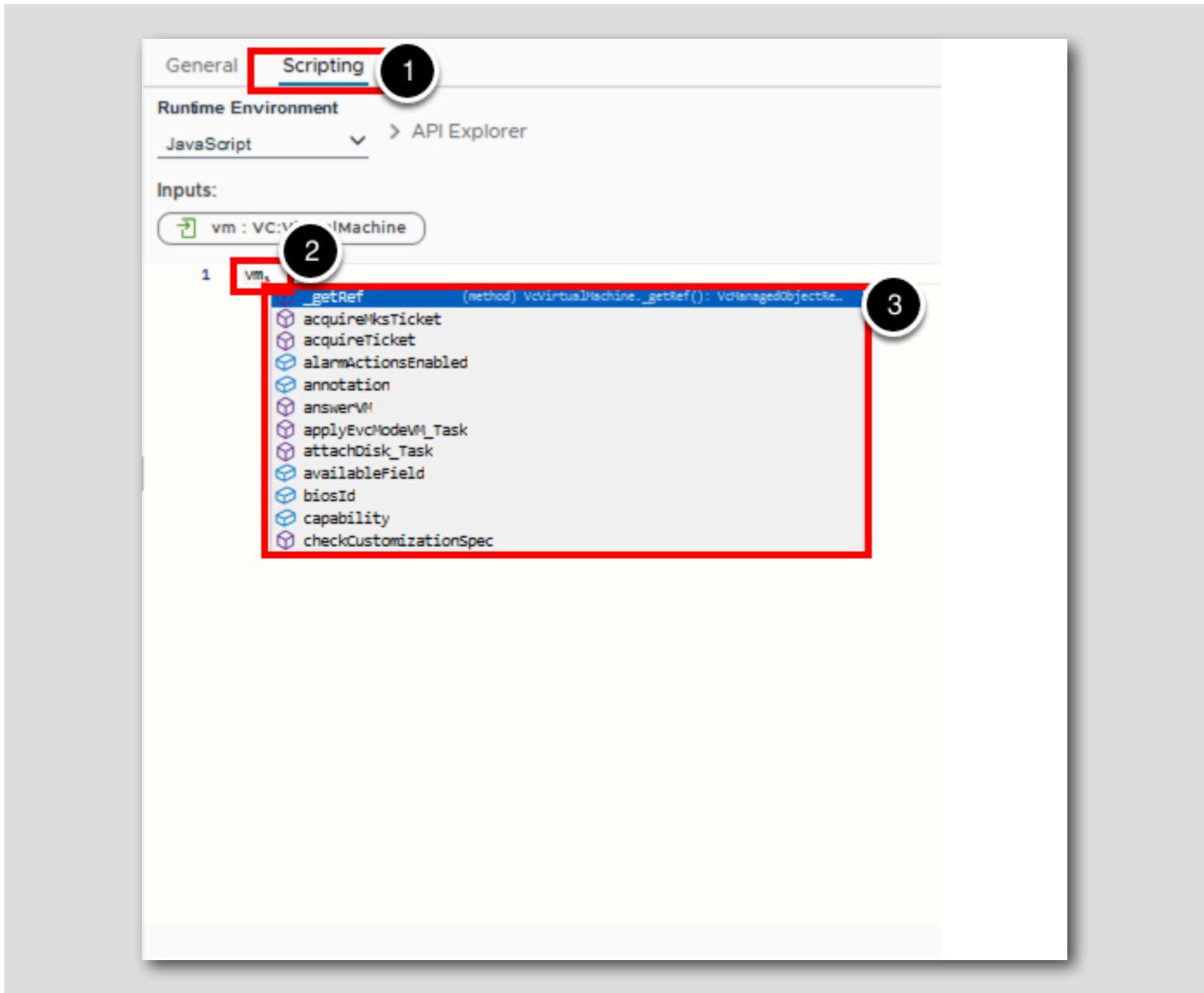
Select the type of the variable



As the workflow will need to retrieve the IP address of a Virtual Machine we need to tell Orchestrator which Virtual Machine it will be working with. We could ask the user to provide the name of the Virtual Machine when the workflow runs, and then use this name to search the vCenter inventory. This approach would not scale well if we had multiple vCenter instances, or if there was a possibility of having more than one Virtual Machine with an identical name. Instead we will set the type to be the Virtual Machine object that Orchestrator can read from the vCenter inventory for simplicity.

1. Enter **vc:virtualm** into the Type field
2. Select **VC:VirtualMachine** from the list
3. Click the **CREATE** button

Browse the available object properties



As we start to type our javascript code, Orchestrator will provide suggestions on the properties or methods of an object that we may wish to use. This is a form of auto completion and is available for object types that exist within the Orchestrator library e.g. Virtual Machines, Clusters, Datastores etc. Let's see how it works for our Virtual Machine object.

1. Select the Scripting tab
2. Enter `vm.` ("vm dot")
3. Observe the list of available properties and methods displayed for this object. This is the same information that is displayed for the object within the API explorer page of Orchestrator.

Search for the ipAddress property

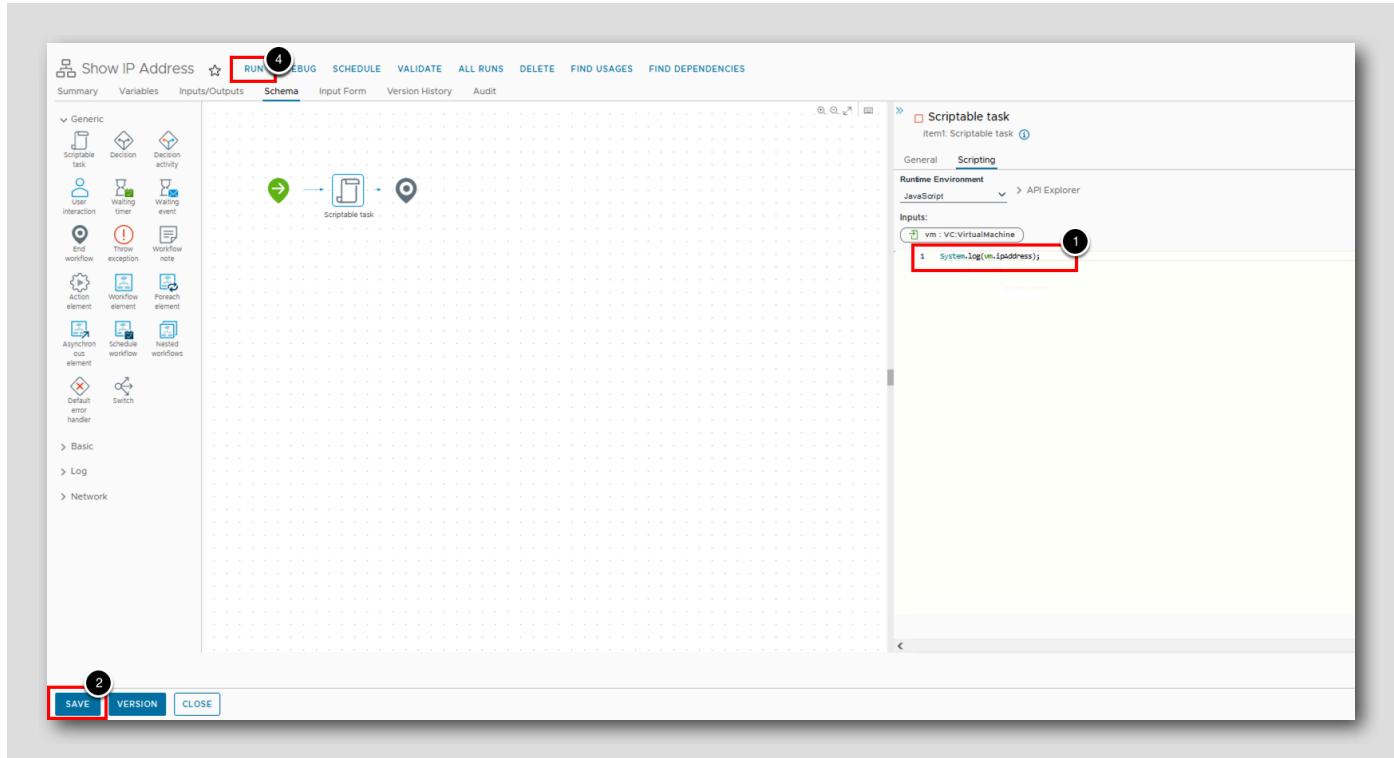


1. Immediately after the "vm." we typed in the previous step, type **ip**

It automatically narrows down the list.

2. Click the **ipAddress** property

## Log the IP address



As a summary, it is possible to start with a given input with a given type (VC:VirtualMachine in this case) and navigate through all its properties and methods directly from the Scripting panel.

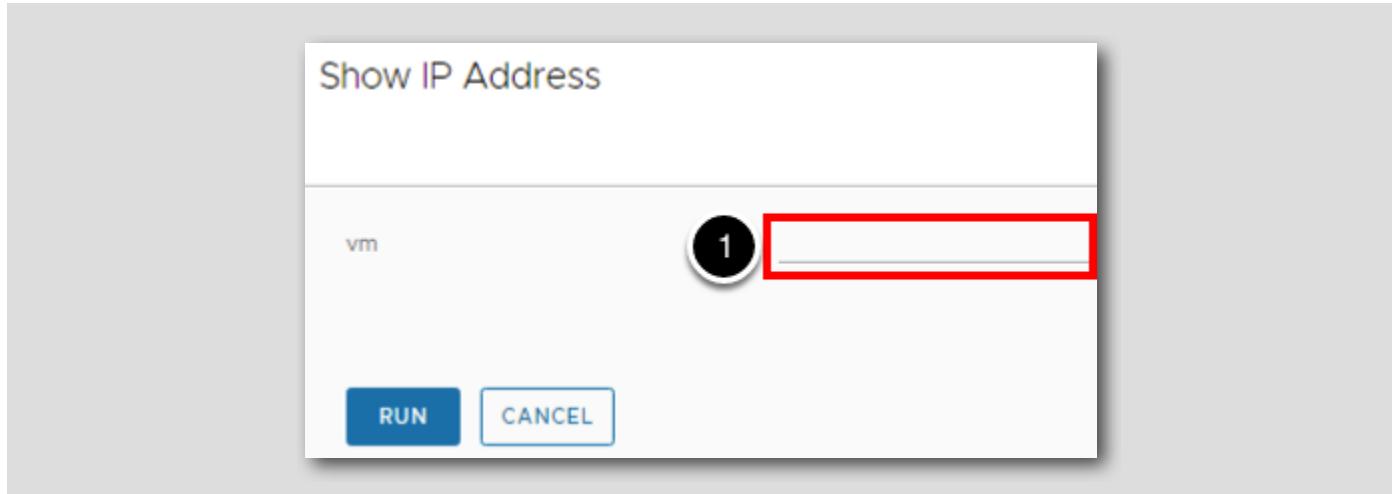
In order to obtain the IP through javascript, the following syntax must be used: `vm.ipAddress`

1. In order to output the IP of the VM in the log, replace the `vm.ipAddress` that we entered in the previous step with the following code:

```
System.log(vm.ipAddress);
```

2. Click the **SAVE** button,
3. In the confirmation prompt, click the **SAVE** button (not shown)
4. Click the **RUN** button

## Set the Virtual Machine input



Our workflow has a single input parameter, named **vm** representing the Virtual Machine object we want to retrieve the IP address for. Let's set the value of this input and run the workflow.

1. Double-click the **vm** field

## Select Virtual Machine

Select VC:VirtualMachine

The screenshot shows the 'Select VC:VirtualMachine' dialog. On the left, a tree view shows 'vSphere vCenter Server' expanded, with 'https://vcenter-mgmt.vcf.sddc.lab:443/sdk (VirtualCenter-8.0.2.0)' selected. Under 'Datacenters', 'mgmt-datacenter-01' is selected, showing 'host', 'vm', 'Development', 'Discovered virtual machine', 'Templates', and 'Workloads'. Under 'Workloads', 'ubuntu-000308' is selected (indicated by a checked checkbox and a red box). Other options like 'ubuntu-001202' and 'windows-000906' are shown but not selected. On the right, a table provides detailed information about the selected VM:

|  |                                      |
|--|--------------------------------------|
| SDK Type                                 | VirtualMachine                       |
| DNS Name                                 | ubuntu-000308                        |
| VMware Tools Status                      | guestToolsRunning                    |
| Consumed Host Memory                     | 0 MB                                 |
| Display name of this virtual machine     | ubuntu-000308                        |
| Non-shared Storage                       | 12708 MB                             |
| configStatus                             | green                                |
| type                                     | VirtualMachine                       |
| 128-bit SMBIOS UUID of a virtual machine | 422b96d7-a74e-1886-df9d-4954d8513b96 |
| Provisioned Storage                      | 52680 MB                             |
| instanceId                               | 502b6de2-811c-3464-22cf-f3b3cef96216 |
| Memory                                   | 1024 MB                              |
| Power State                              | poweredOn                            |
| id                                       | vm-11009                             |
| Orchestrator unique ID                   | vm-11009                             |
| Annotation                               |                                      |
| VM Connection State                      | connected                            |
| Consumed Host CPU                        | 39 MHz                               |
| guestHeartbeatStatus                     | green                                |
| IP address                               | 10.64.12.10                          |

At the bottom right, there are 'CANCEL' and 'SELECT' buttons. The 'SELECT' button is highlighted with a red box and a circled number '2'.

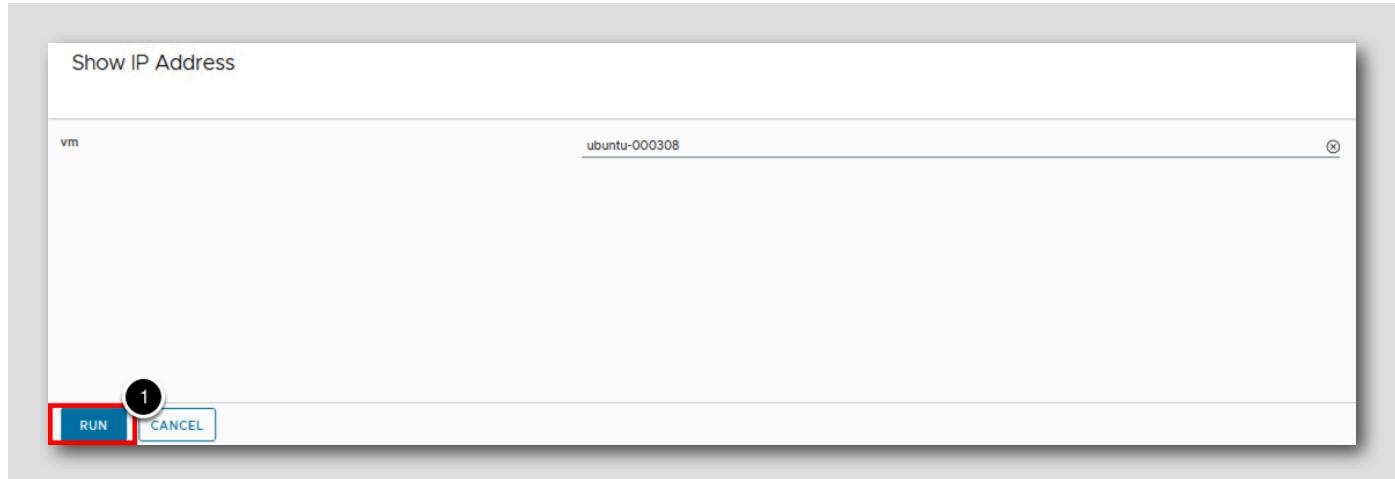
1. Navigate to vSphere vCenter Plugin > <https://vcenter-mgmt.vcf.sddc.lab:443/sdk> > Datacenters > mgmt-datacenter-01 > vm

> Workloads > ubuntu-000308

2. Click SELECT

The name of the Virtual Machine within your lab may differ from the screenshot. Select any Virtual Machine with a name in the format ubuntu-000xxx.

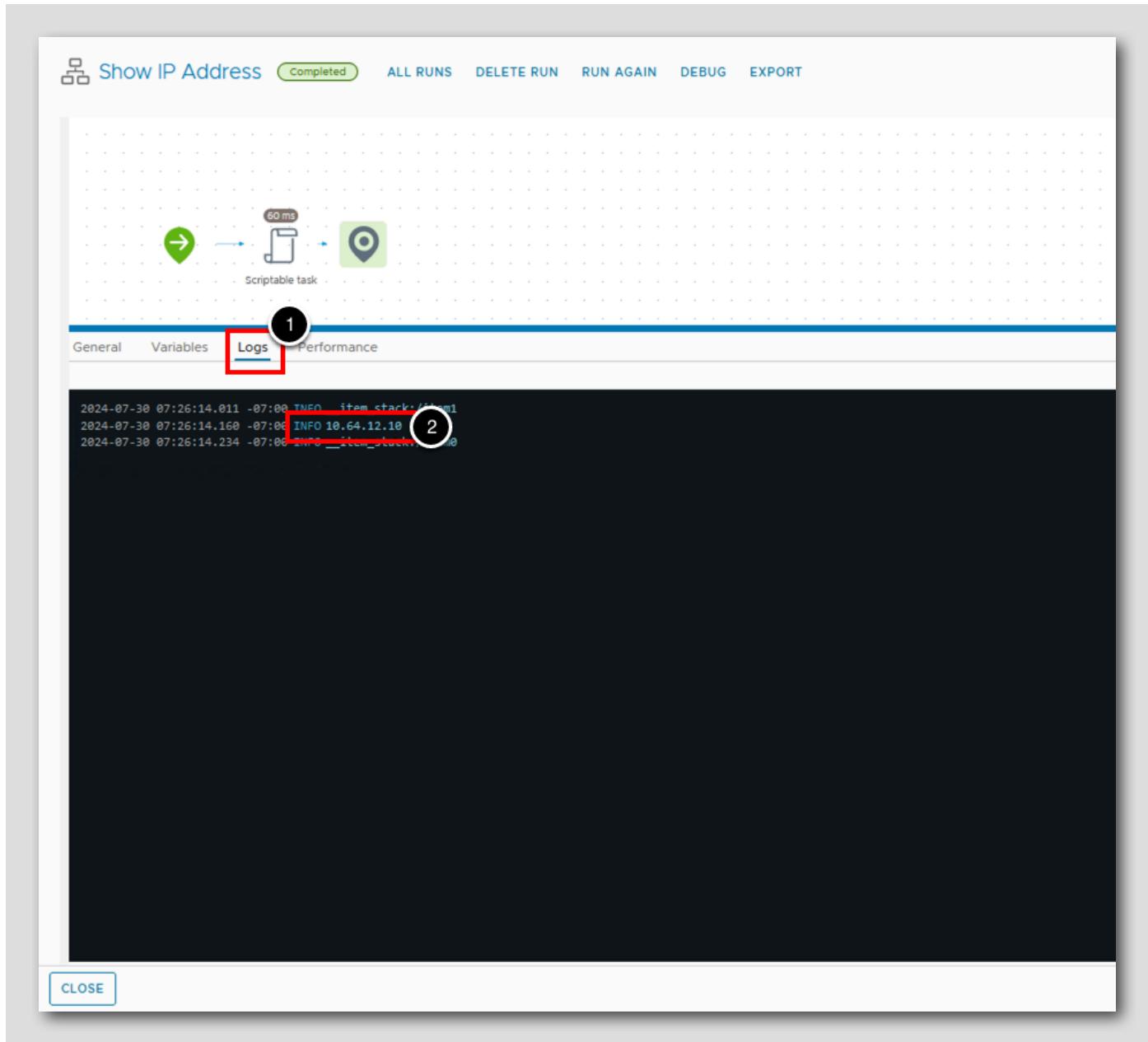
## Run the workflow



The machine selected in the previous step should be visible in the vm field and we are now ready to run the workflow.

1. Click the RUN button

Find the IP address in the logs

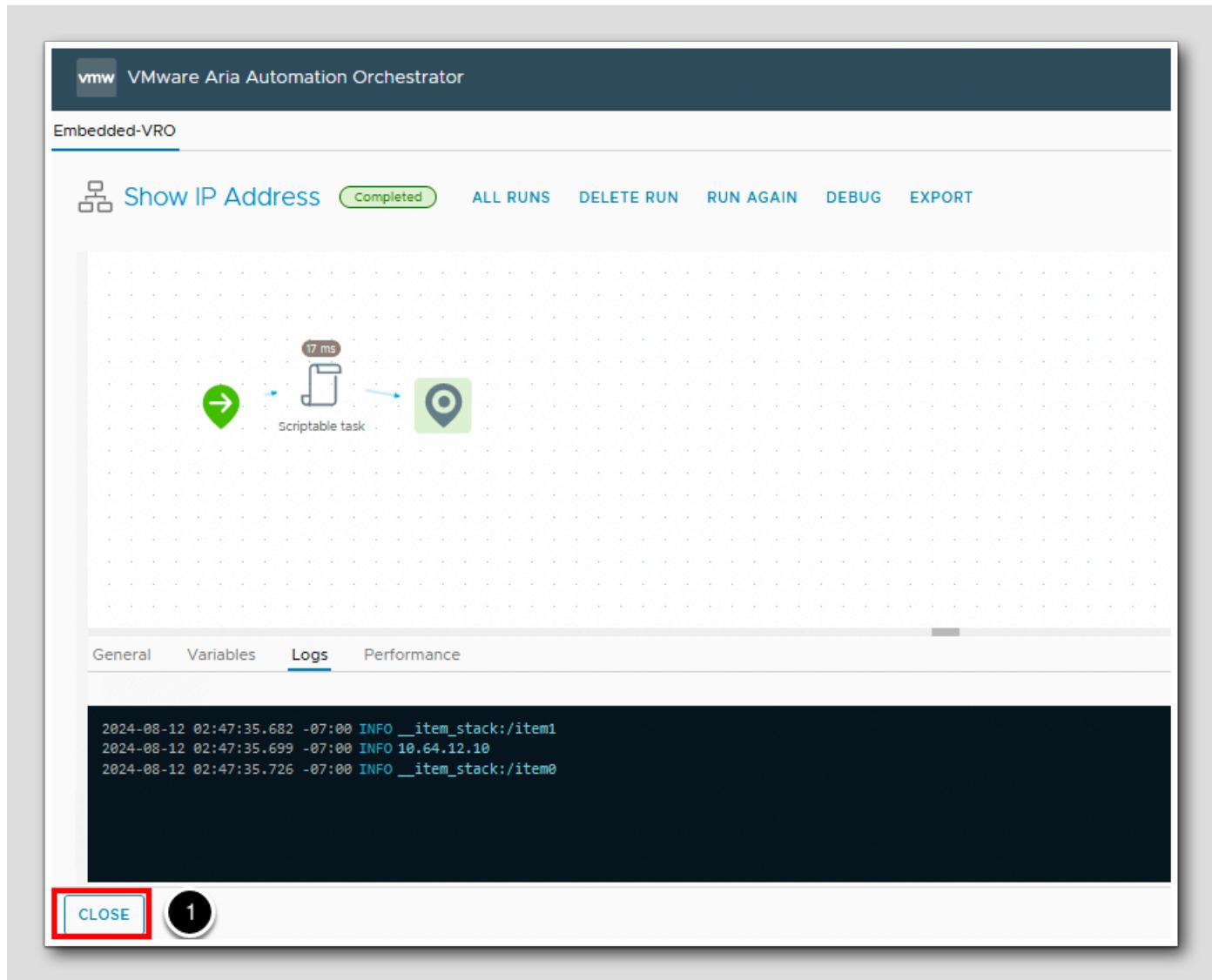


Once the workflow completes we can review the logs to see if Orchestrator was able to retrieve the IP address of our Virtual Machine.

1. Select the **Logs** tab
2. Look at the IP address in the logs (the value might not match the screenshot)

The IP address of the VM selected when we ran the workflow is shown in the logs.

Close the RUN window



1. Click on CLOSE to close the run window.

## Close the workflow window

The screenshot shows the VMware Aria Automation Orchestrator interface. At the top, it says "vmw VMware Aria Automation Orchestrator" and "Embedded-VRO". Below that is a toolbar with "Show IP Address" and icons for RUN, DEBUG, SCHEDULE, VALIDATE, ALL RUNS, DELETE, FIND USAGES, and FIND DEPENDENCIES. The "Summary" tab is selected. The main area displays workflow details:

- Workflow name:** Show IP Address
- ID:** be0755e9-5224-40a7-92da-fbad41cf72bd
- Version:** 0.0.0
- Tags:** web-root (with a placeholder "Enter a new tag")
- Folder:** /web-root (with a "SELECT FOLDER" button)
- Server restart behavior:** Resume workflow run
- Resume workflow from failed behavior:** System default
- Description:** Add description (with a rich text editor placeholder)

At the bottom, there are buttons for "SAVE", "VERSION", and "CLOSE". The "CLOSE" button is highlighted with a red box. To its right is a circular icon with the number "1".

1. Click on CLOSE to close the workflow window.

## Managing Actions

We can modify our Aria Automation Orchestrator workflows by adding action scripts.

Actions represent individual functions that you use as building blocks in workflows.

Actions are JavaScript functions. Actions can take multiple input parameters, just like workflows. However, actions can only have a single return value. Actions can call on any object in the Aria Automation Orchestrator API, or objects in any API that you import into Aria Automation Orchestrator by using a plug-in.

When a workflow runs, an action takes input parameters from the workflow's variables. These variables can be either the workflow's

initial input parameters, or variables that other elements in the workflow set when they run.

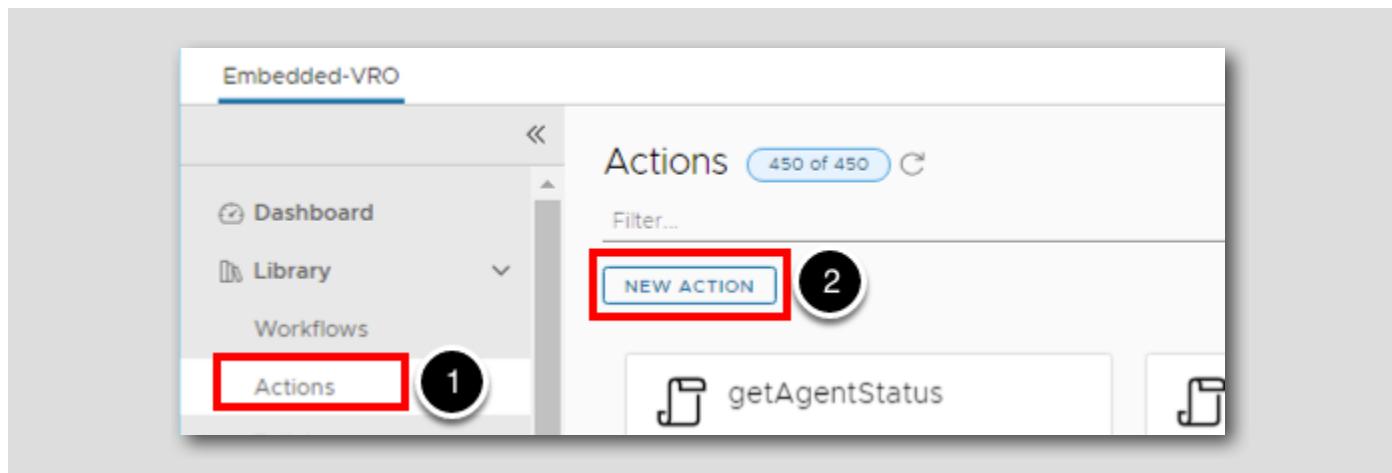
The Aria Automation Orchestrator client provides libraries of predefined actions and an action editor for custom action scripts.

The action editor includes an autocomplete feature for scripts and an API Explorer featuring the available scripting types and their documentation.

To illustrate this, let's create an action to return a boolean value of true if a Virtual Machine is marked as a template.

## Create a new action

[118]



Let's start by creating a new action that we can use with our custom code.

1. Select Actions tab
2. Click the NEW ACTION button

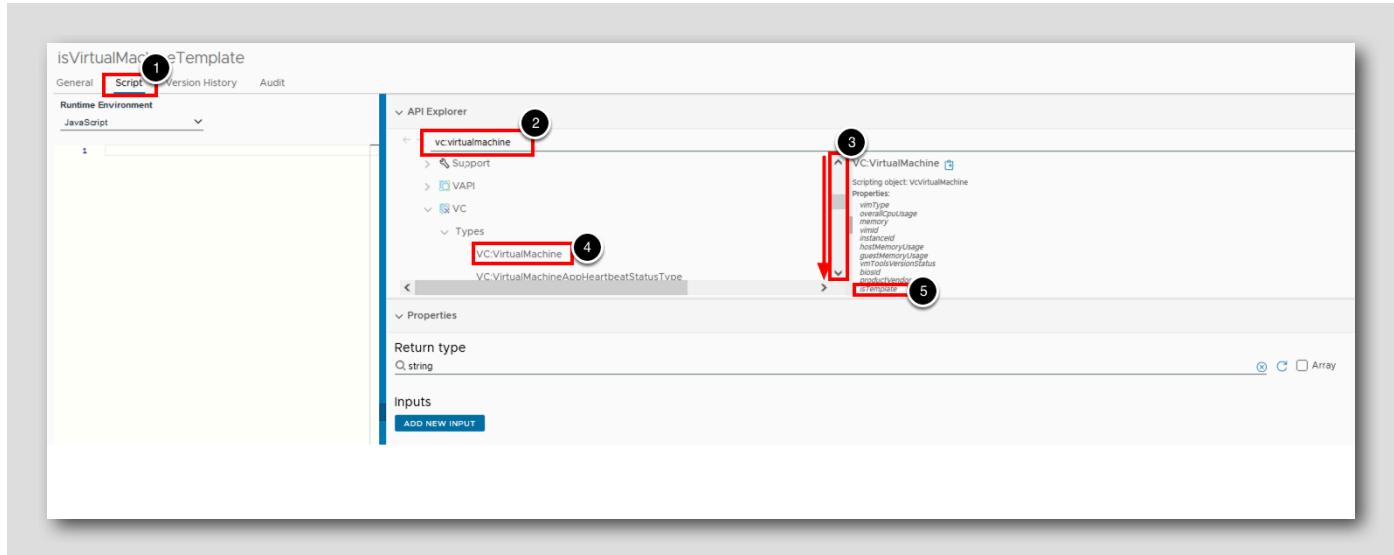
Enter the action information

The screenshot shows the 'General' tab of the 'isVirtualMachineTemplate' action configuration. The 'Name' field is set to 'isVirtualMachineTemplate'. In the 'Module' field, 'com.broadcom.hol' is typed, and a dropdown menu lists 'com.broadcom.hol' as the only result. The 'Version' field is set to '0.0.0'. The 'Tags' field has the placeholder 'Enter a new tag'. The 'Groups' field is a dropdown menu.

In the same way as when we create a workflow, we must provide some basic information for our new action.

1. In the Name field enter **isVirtualMachineTemplate**
2. In the Module field, type **com.broadcom.hol**
3. From the list that appears, select **com.broadcom.hol**

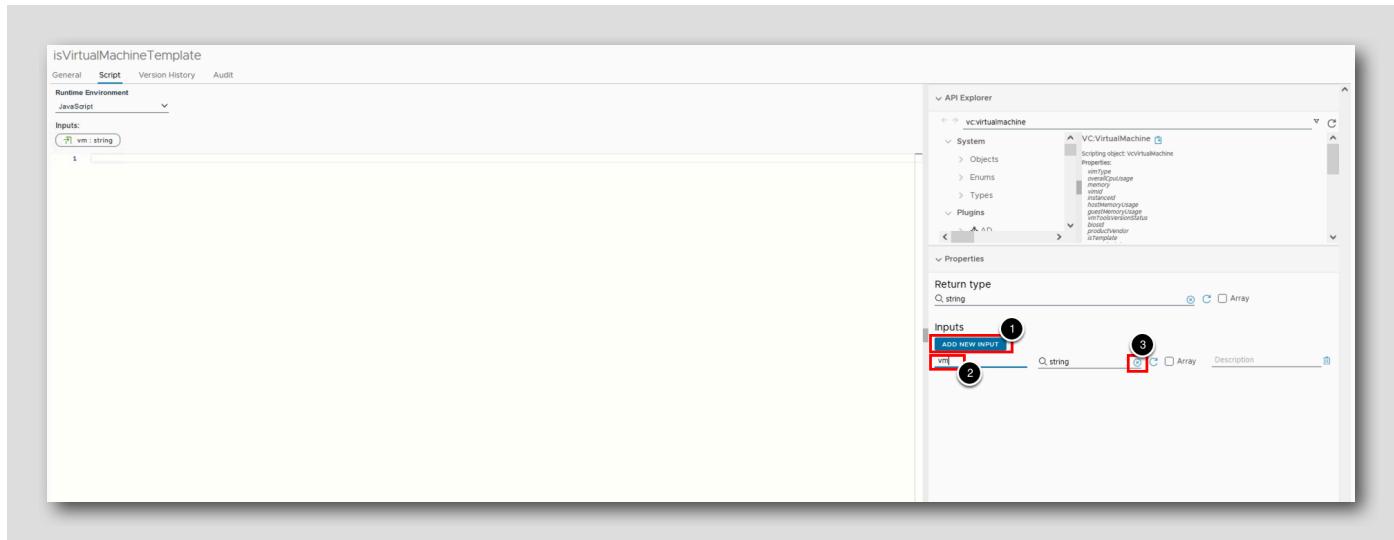
## Try the API Explorer



We can use the API Explorer to determine the appropriate object type(s) our action needs to take in as input in order to accomplish a task.

1. Select the Script tab
- 2.Under the API Explorer section, type `vc:virtualmachine`
- 3.Use the scroll bar to scroll down until you can see VC listed under the Plugins section
- 4.In the tree below, select Plugins > VC > Types > VC:VirtualMachine
- 5.The pane to the right shows the VC:VirtualMachine object type's properties, including one called `isTemplate`

## Add an input to the action



Our action is going to check if a Virtual Machine is marked as a template. Let's add an input to pass the Virtual Machine details we want Orchestrator to check into our action.

1. Click the ADD NEW INPUT button
2. For the input name, enter **vm**
3. Remove the string type

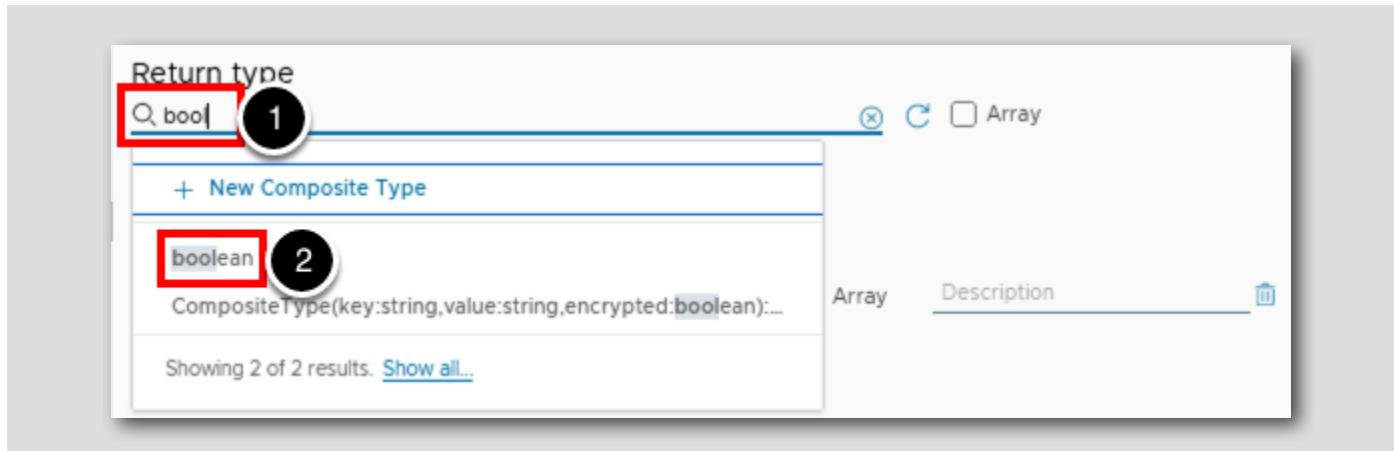
## Select the input type

The screenshot shows the 'Properties' dialog for an orchestrator step. In the 'Inputs' section, there is a field labeled 'vm' with the placeholder 'vc:virtualm'. A red box highlights this input field, and a black circle with the number '1' is placed above it. To the right of the input field is a dropdown menu containing various vCenter object types. A red box highlights the first item in the list, 'VC:VirtualMachine', and a black circle with the number '2' is placed above it. At the top of the properties dialog, there is a 'Return type' section where the 'string' type is selected. To the right of this, there are two checkboxes: one for 'Array' (unchecked) and one for 'Object' (unchecked). A black circle with the number '3' is placed above the 'Object' checkbox.

By default Orchestrator using an input type of String. We want to work with the Virtual Machine object so let's change the parameter type to be a vCenter Virtual Machine object.

1. Where it says Select Type, enter **vc:virtualm**
2. Select **VC:VirtualMachine** from the filtered list of types
- 3.Under Return type, remove the string type

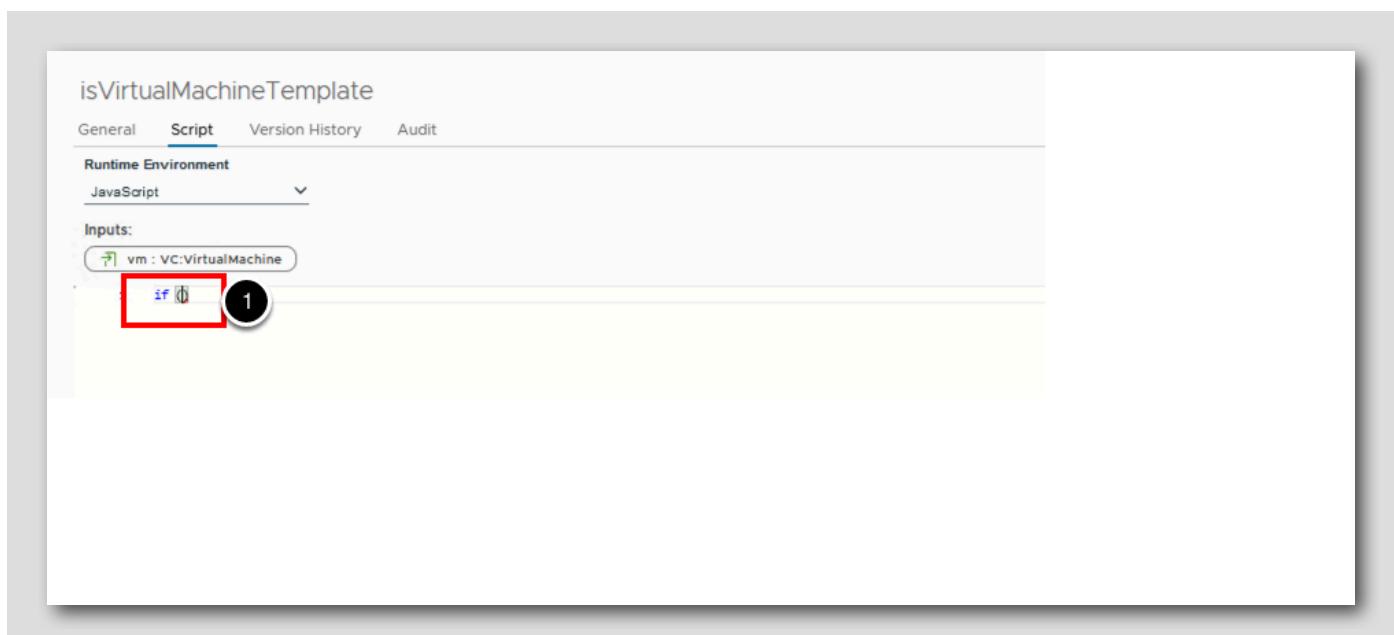
## Change return type



Just like the default type for inputs, Orchestrator uses a default return type of string as well. We want to return a TRUE or FALSE value, or boolean type value from our action so let's change the return type.

1. Where it says Select Type, enter bool
2. Select boolean from the filtered list of types

## Start entering the JavaScript code



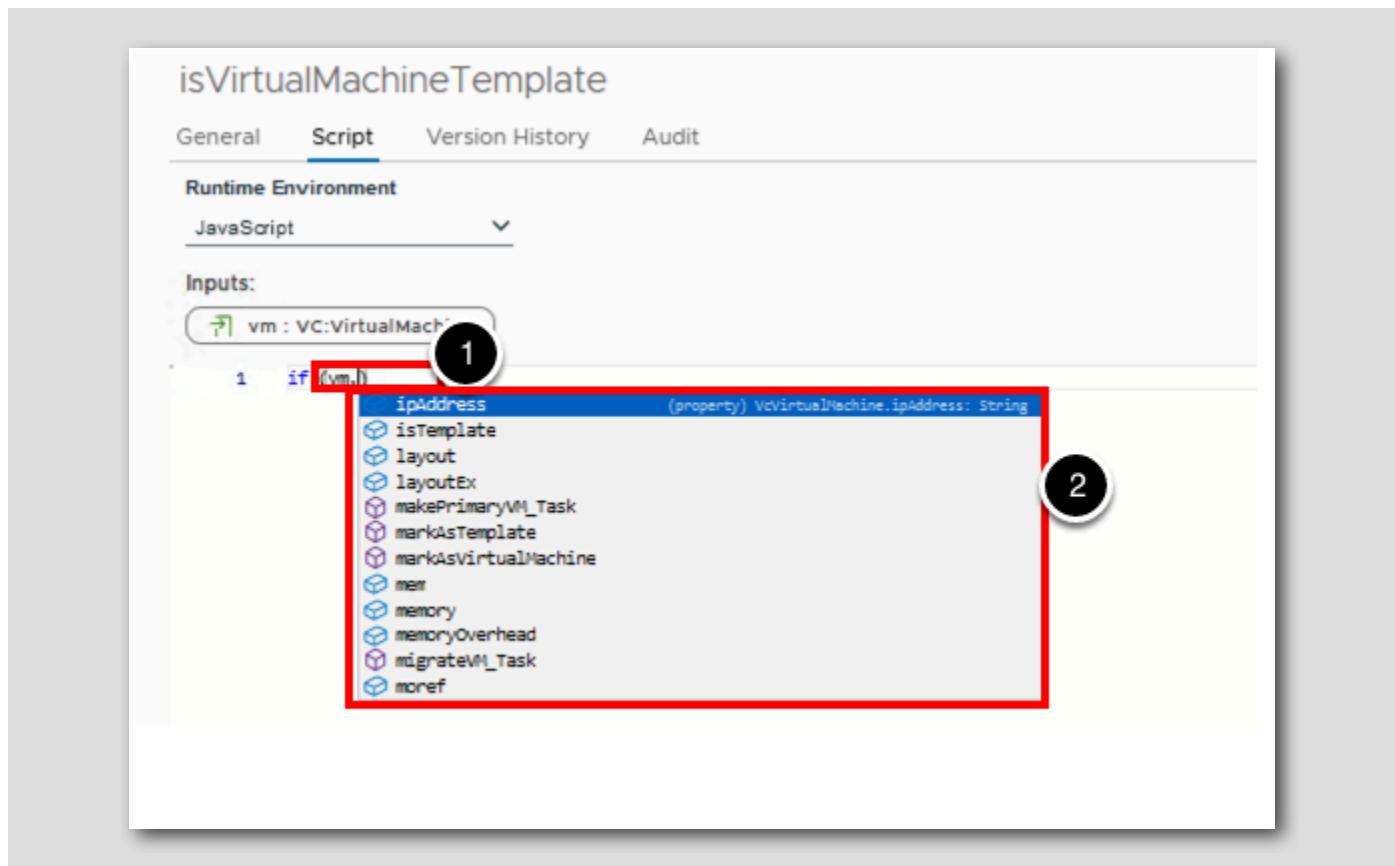
With the correct input and return types set, we can now start to enter the Javascript code we need to check if a Virtual Machine is a template.

1. In the left-hand scripting pane, enter `if (`

Notice that the Aria Automation Orchestrator client automatically added the closed-parenthesis to the end of line 1.

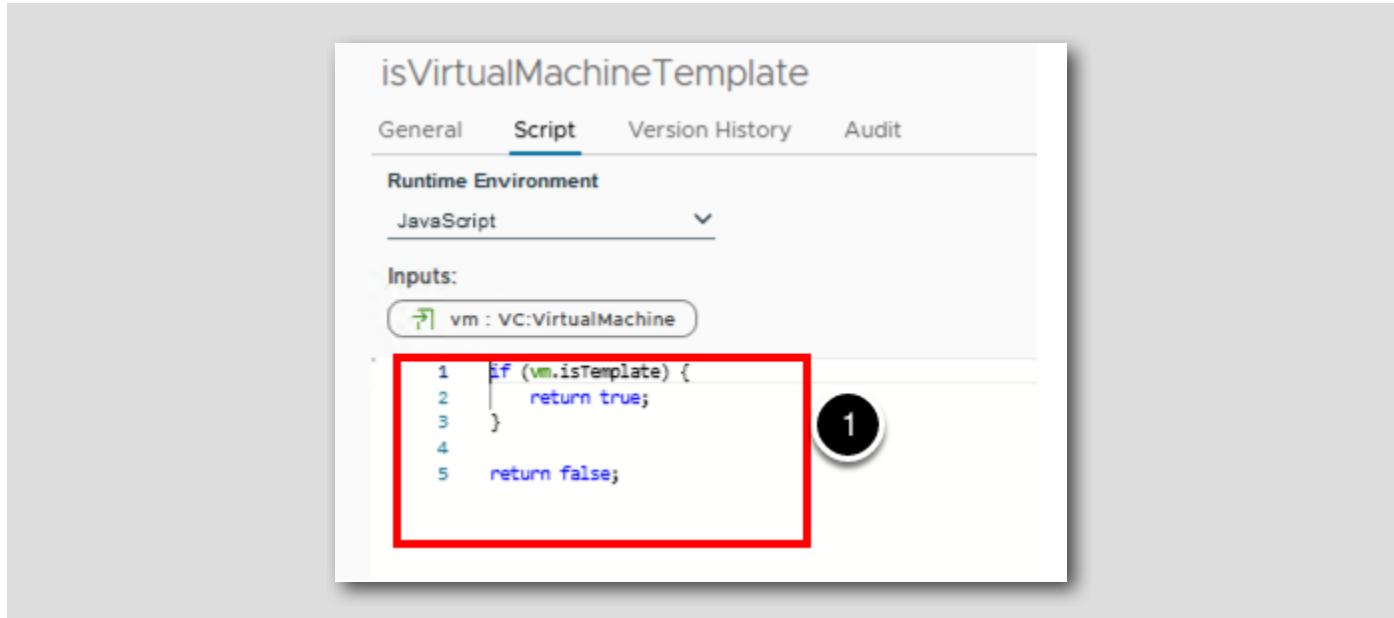
Browse the VM object options

[125]



1. With the cursor still inside the parentheses, type `vm.` (vm dot)
2. Notice that all the properties of the `vm` input object are displayed, including `isTemplate`, which we found in the API Explorer

Complete the action code



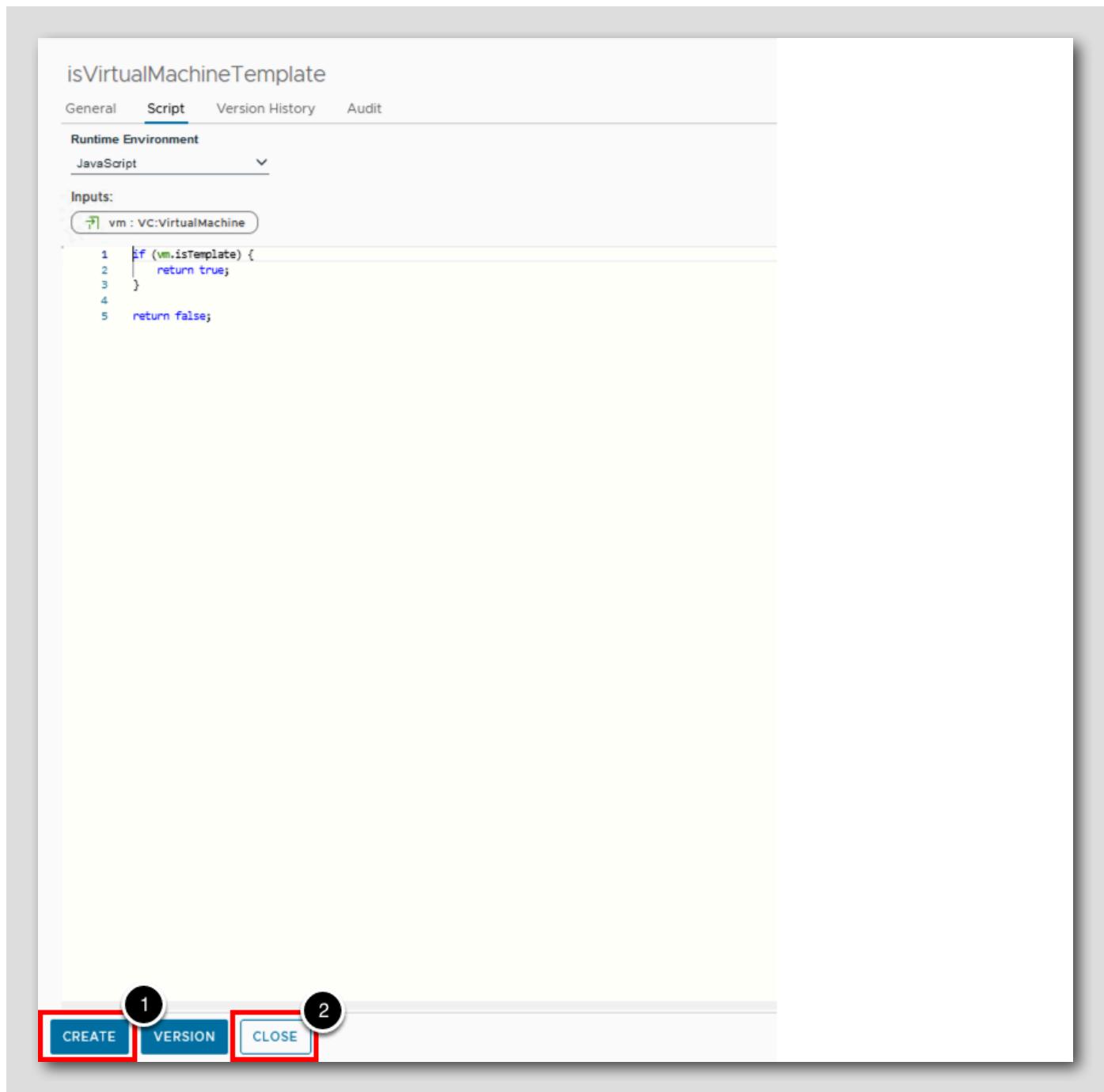
While we can manually type out the remaining code we need to perform the template check inside our action, to save time we are going to copy and paste the code directly into the action.

1. Replace all code we've written so far inside the action with the following:

```
if (vm.isTemplate) {  
    return true;  
}  
  
return false;
```

This just tells Orchestrator to return the value of true if the Virtual Machine is a template, and false otherwise.

Save the new action

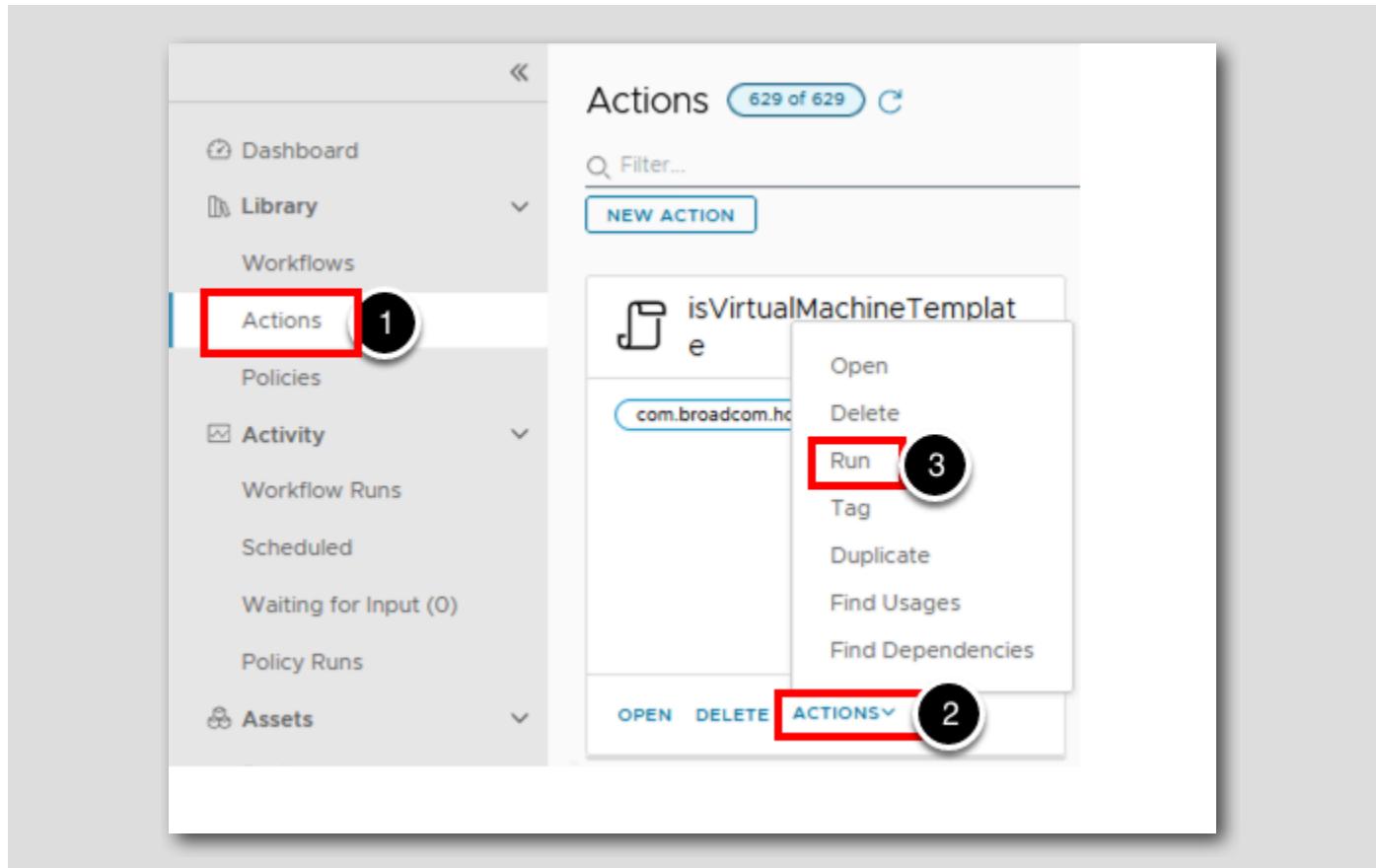


1. Click CREATE to save the action
2. Click CLOSE to return to the Actions page

We've created an action. We can make this action even more useful by calling it from within a workflow.

## Run the new action

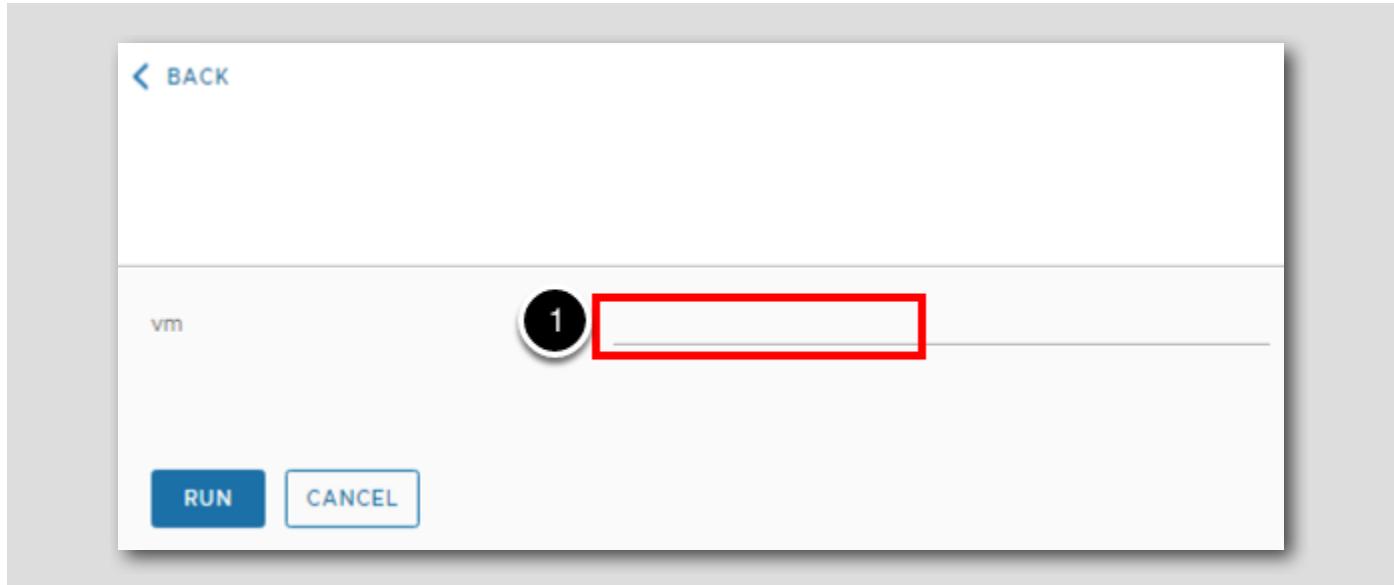
[128]



Before we use our action from within a workflow schema, let's test out our new action to make sure it does what we expect.

1. Return to the Actions tab
2. For the isVirtualMachineTemplate action, click the ACTIONS menu
3. Click Run

Set the Virtual Machine input value



1. Click the vm field

## Select Virtual Machine

The screenshot shows the "Select VC:VirtualMachine" dialog box. On the left is a tree view of the vSphere vCenter Server structure, with the path selected: "https://vcenter-mgmt.vcf.sddc.lab:443/sdk (VirtualCenter-8.0.2.0) > Datacenters > mgmt-datacenter-01 > vm". A red box highlights the item "ubuntu-000308" under "Workloads", which is checked. Number 1 is placed over this highlighted item.

The main area is a table of VM properties. A vertical red box on the right side of the table highlights the "VM Template" row, which has a value of "false". Number 2 is placed above the top of this red box, and number 3 is placed directly next to the "false" value in the "VM Template" row.

At the bottom right of the dialog are two buttons: "CANCEL" and "SELECT". A red box highlights the "SELECT" button, and number 4 is placed to its right.

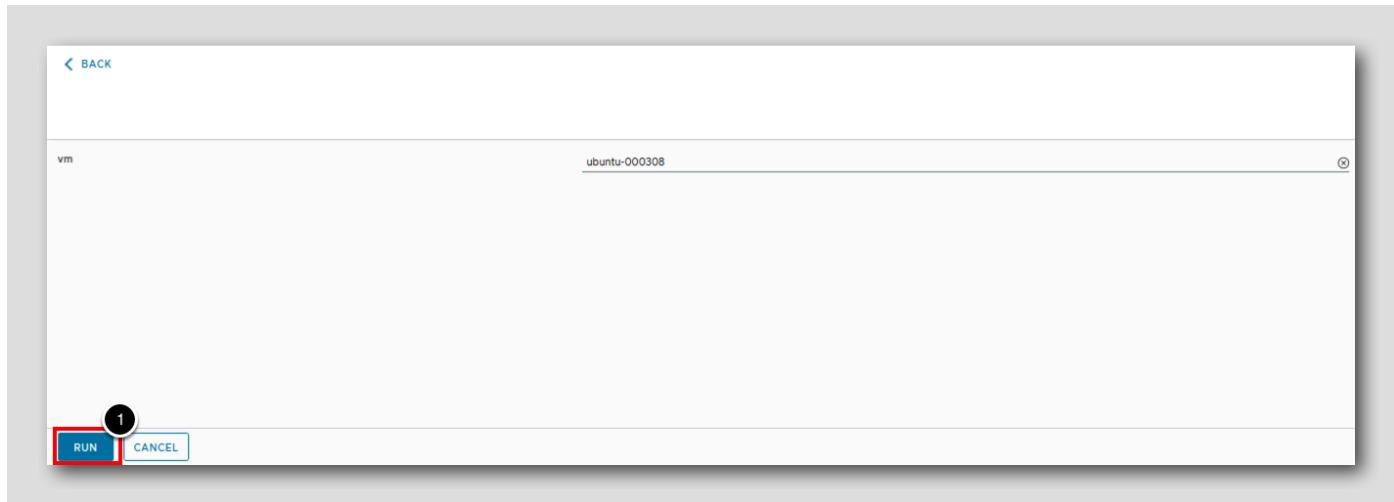
| id                          | vm-11009                              |
|-----------------------------|---------------------------------------|
| Orchestrator unique ID      | vm-11009                              |
| Annotation                  |                                       |
| VM Connection State         | connected                             |
| Consumed Host CPU           | 59 MHz                                |
| guestHeartbeatStatus        | green                                 |
| IP address                  | 10.64.12.10                           |
| Active Guest Memory         | 0 MB                                  |
| CPU                         | 1 vCPUs                               |
| Product Vendor              |                                       |
| Memory Overhead             |                                       |
| Guest OS                    | Ubuntu Linux (64-bit)                 |
| dunesId                     | vcenter-mgmt.vcf.sddc.lab.id.vm-11009 |
| <b>VM Template</b>          | <b>false</b>                          |
| sdkId                       | vcenter-mgmt.vcf.sddc.lab/vm-11009    |
| Used Storage                | 13200 MB                              |
| name                        | ubuntu-000308                         |
| VMware Tools Version Status | guestToolsUnmanaged                   |
| VM Version                  | vmx-21                                |
| alarmActionsEnabled         | true                                  |
| overallStatus               | green                                 |

Let's first try this with a Virtual Machine that we know is not a template.

1. Navigate to vSphere vCenter Plugin > <https://vcenter-mgmt.vcf.sddc.lab:443/sdk> > Datacenters > mgmt-datacenter-01 > vm > Workloads > ubuntu-000308
2. Scroll down to the VM Template property
3. Confirm the VM Template property value is set to false
4. Click SELECT

The name of the Virtual Machine within your lab instance may differ from the screenshot. You can choose any Virtual Machine with a name in the format ubuntu-000xxx which has the VM Template property set to false.

## Run the action



The machine selected in the previous step should be visible in the vm field.

1. Click the RUN button
- 2.(Not shown) wait until the Action completes and its Status shows as Completed in a green bubble at the top of the screen.

## Find the action run results

The screenshot shows the 'isVirtualMachineTemplate' script details in the vSphere Client. The 'RUN' button is highlighted with a red box and circled with number 3. The 'Result / Inputs' tab is selected and highlighted with a red box and circled with number 1. The 'Action Result' row is highlighted with a red box and circled with number 2.

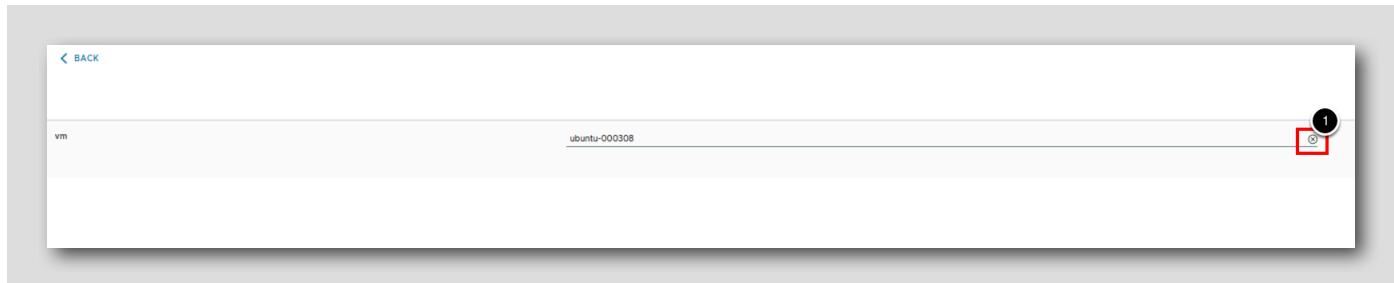
| Name          | Type              |
|---------------|-------------------|
| vm            | VC:VirtualMachine |
| Action Result | boolean           |

Once the action run has completed:

1. Look at the Results / Inputs tab
2. The Action Result is **false**, which is the expected result, because we know this Virtual Machine is not a template
3. Click RUN to test out our action with a Virtual Machine that **is** a template

Remove the previous Virtual Machine input

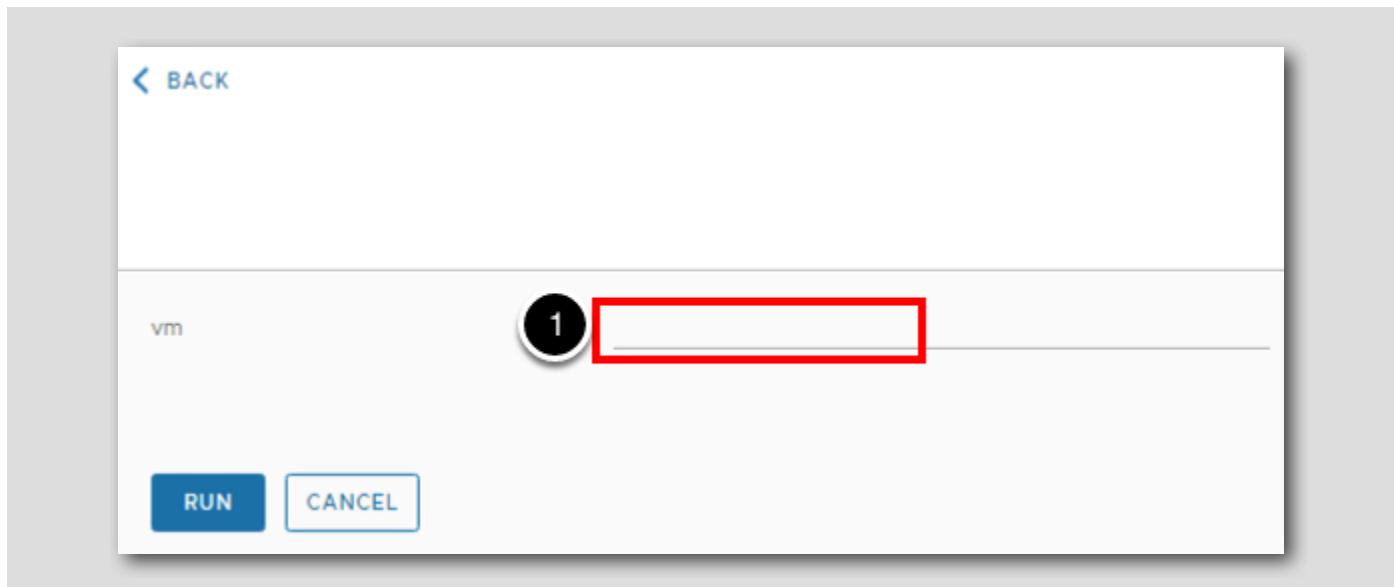
[133]



1. Remove the vm value from the previous action run

Set the new Virtual Machine input

[134]



1. Click the vm field

## Select Virtual Machine

Select VC:VirtualMachine

|                             |                                      |
|-----------------------------|--------------------------------------|
| Power State                 | poweredOff                           |
| id                          | vm-6002                              |
| Orchestrator unique ID      | vm-6002                              |
| Annotation                  |                                      |
| VM Connection State         | connected                            |
| Consumed Host CPU           | 0 MHz                                |
| guestHeartbeatStatus        | gray                                 |
| Active Guest Memory         | 0 MB                                 |
| CPU                         | 2 vCPUs                              |
| Product Vendor              |                                      |
| Memory Overhead             |                                      |
| Guest OS                    | Ubuntu Linux (64-bit)                |
| dunesId                     | vcenter-mgmt.vcf.sddc.lab:id:vm-6002 |
| <b>VM Template</b>          | true                                 |
| sdkId                       | vcenter-mgmt.vcf.sddc.lab/vm-6002    |
| Used Storage                | 12568 MB                             |
| name                        | ubuntu22                             |
| VMware Tools Version Status | guestToolsUnmanaged                  |
| VM Version                  | vmx-21                               |
| alarmActionsEnabled         | true                                 |
| overallStatus               | green                                |

**CANCEL** **SELECT**

This time, we'll select a Virtual Machine that we know is a template.

1. Navigate to vSphere vCenter Plugin > <https://vcsa-01a.corp.local:443/sdk> > Datacenters > RegionA01 > vm > Templates > ubuntu22
2. Scroll down to the VM Template property
3. Confirm the VM Template property is set to true
4. Click SELECT

## Run the action



The machine selected in the previous step should be visible in the vm field.

1. Click the RUN button

## Find the action run results

[137]

The screenshot shows the 'isVirtualMachineTemplate' action run results in the vSphere Client. The 'Result / Inputs' tab is selected (circled with 1). The table contains one row with the following data:

| Name          | Type    |
|---------------|---------|
| Action Result | boolean |

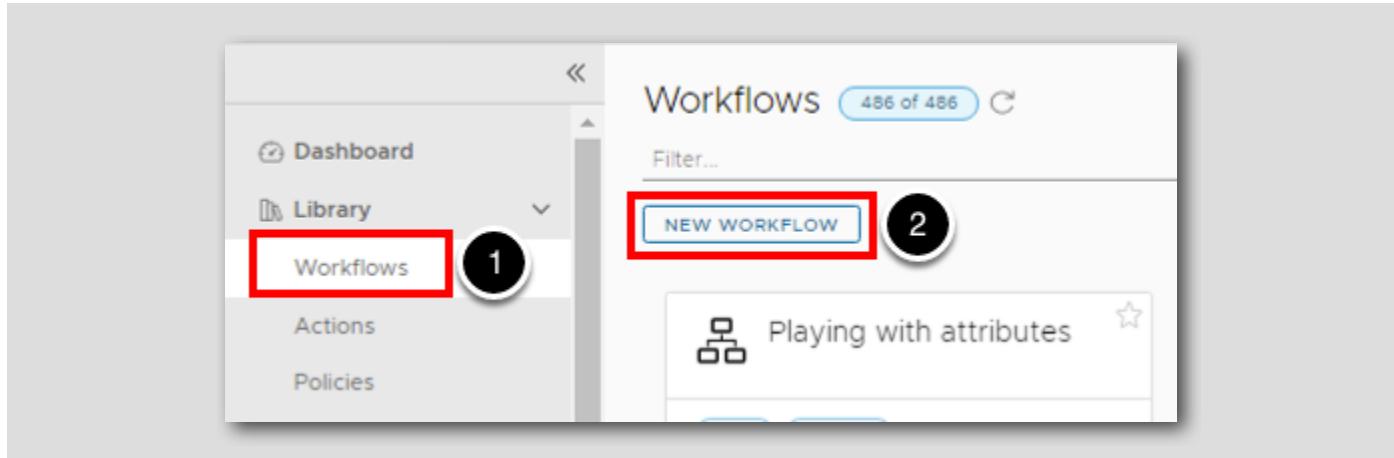
The 'Value' column shows '[object VC:VirtualMachine]'. The 'Logs' tab is also visible. At the bottom, the 'CLOSE' button is highlighted with a red box and a circled number 3.

Once the action run has completed:

1. Look at the Results / Inputs tab
2. The Action Result is true, which is the expected result, because we know this Virtual Machine is a template
3. Click the CLOSE button

Now let's finally call this action from a workflow.

## Create a workflow

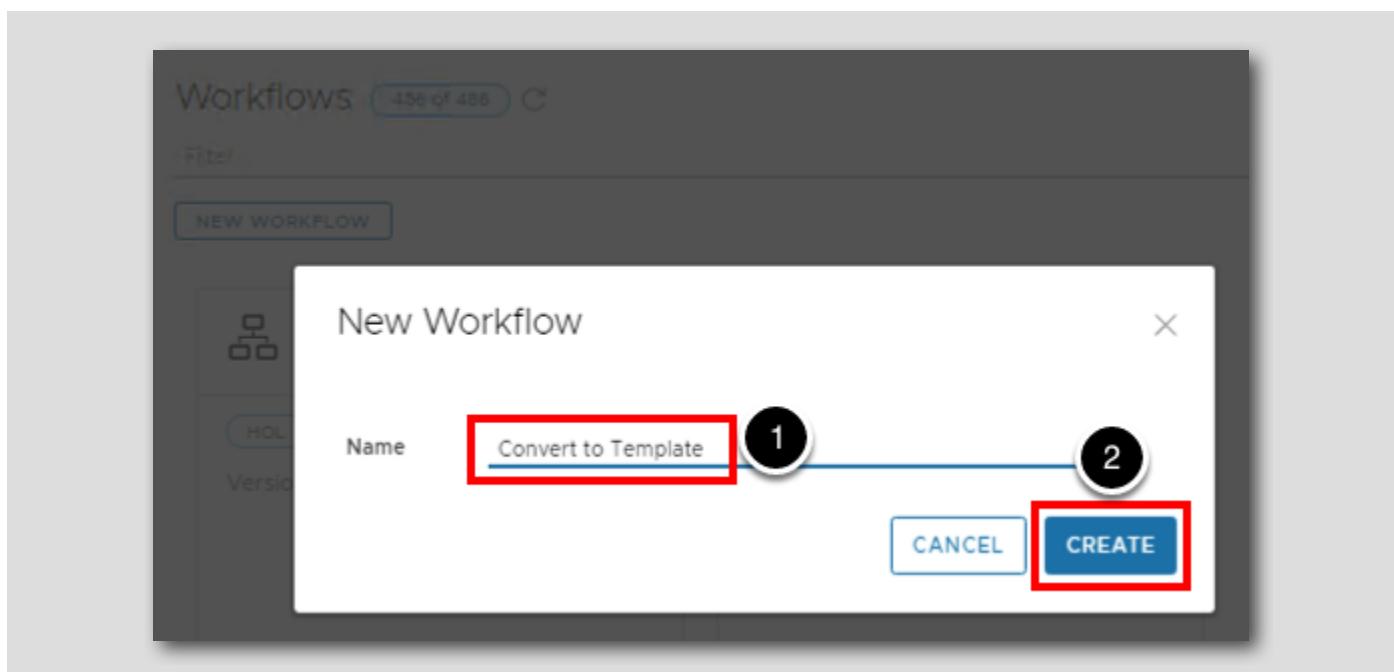


Let's create a workflow that will convert a Virtual Machine to a Virtual Machine template if it is not already marked as a template:

1. Navigate to the **Workflows** tab
2. Click the **NEW WORKFLOW** button

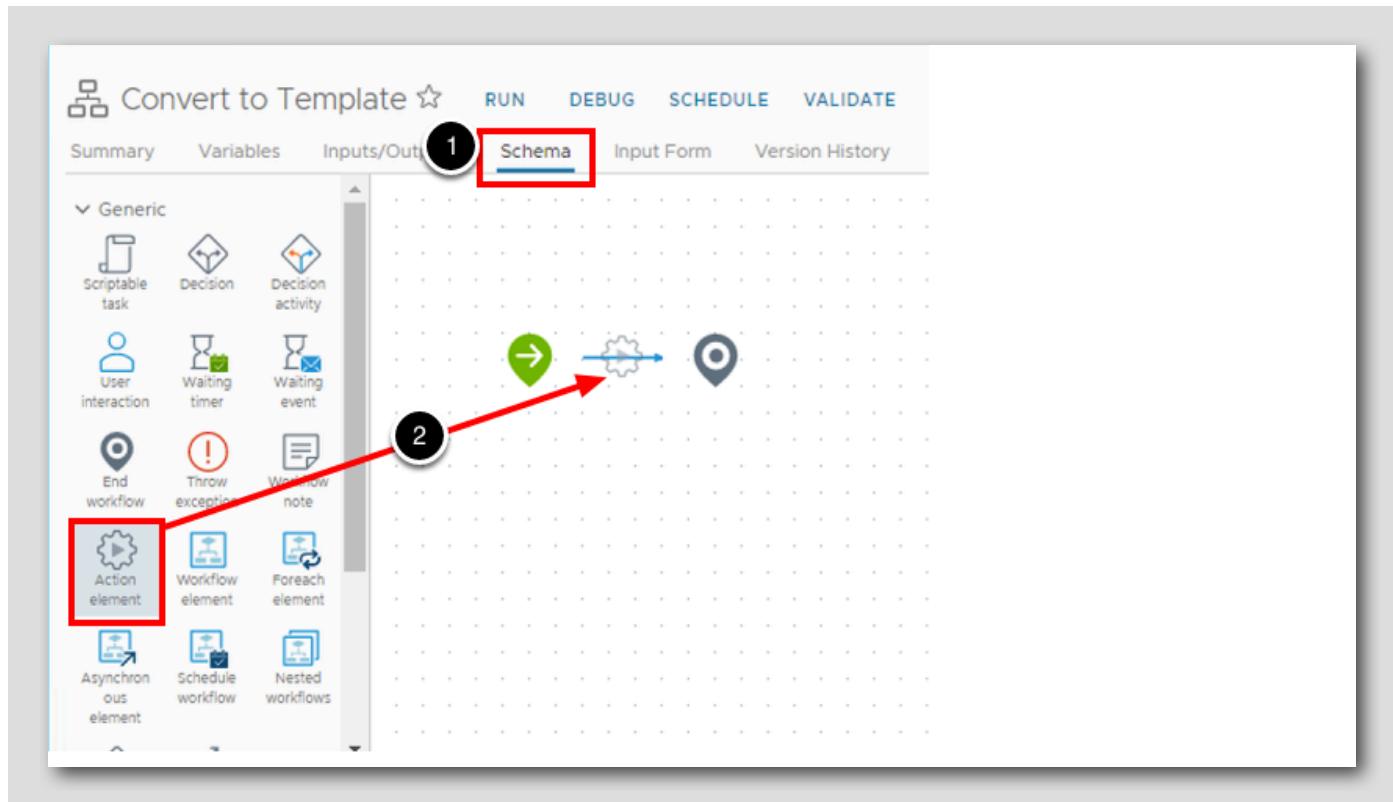
Enter the workflow name

[139]



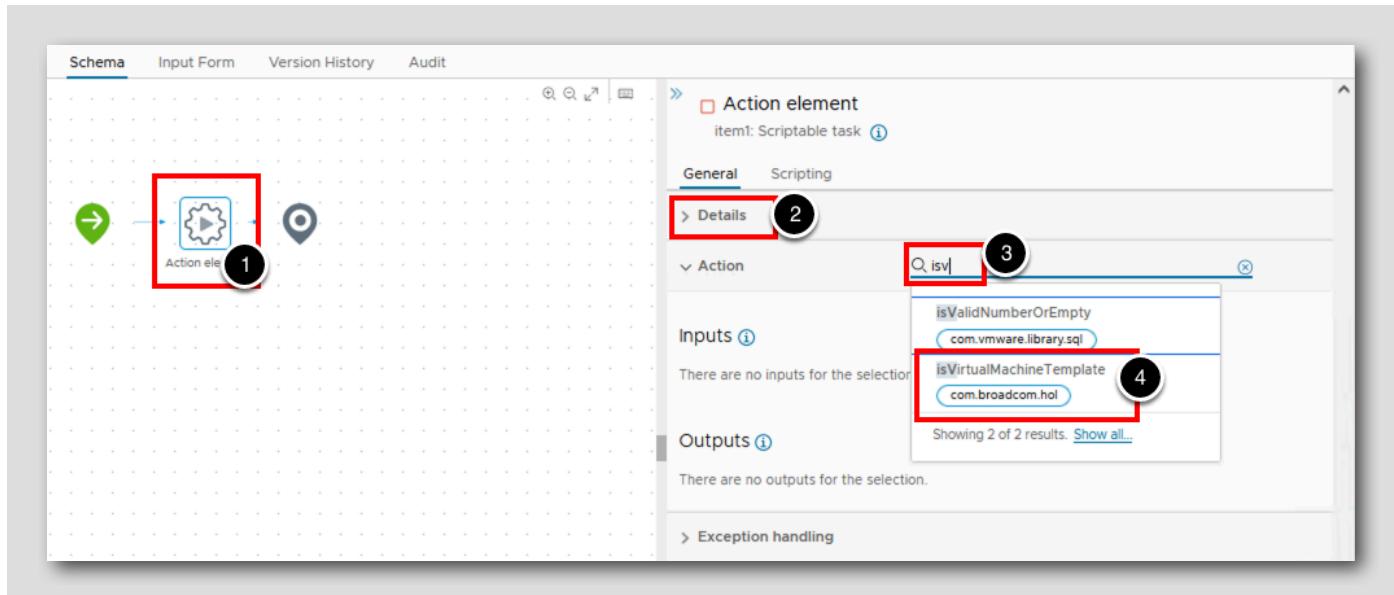
1. In the Name field enter Convert to Template
2. Click the CREATE button

Add an action element



1. Select the Schema tab
2. Drag an Action element into the schema

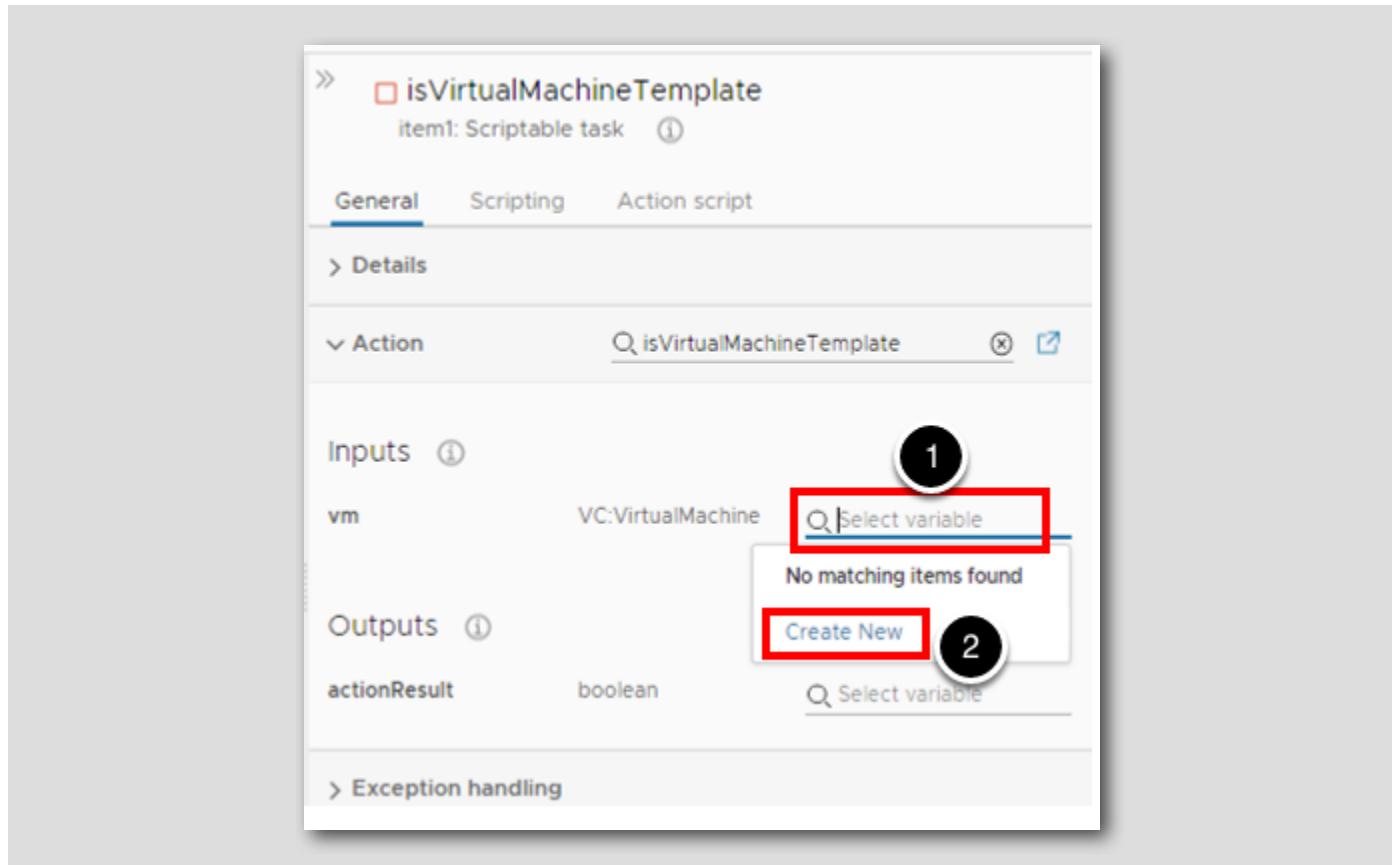
## Select our custom action



We now want to tell Orchestrator that this action element should call the action we created in the previous steps.

1. Select the **Action element**
2. Collapse the **Details** pane
3. In the Action pane, where it says *Select action*, type **isv**
4. Select our newly created action **isVirtualMachineTemplate**

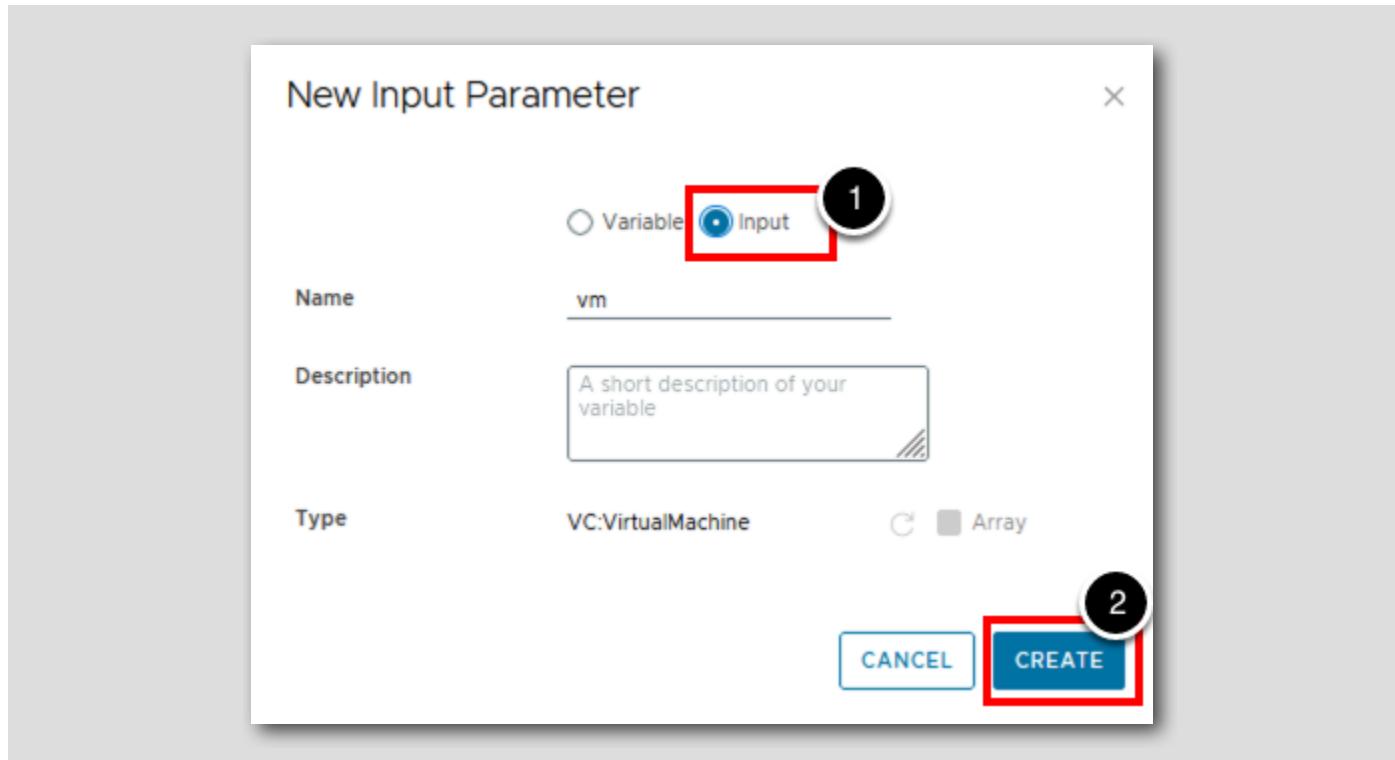
## Add a new input parameter



Our action expects an input parameter for the Virtual Machine object. We need to map this input to a variable, or input parameter within the workflow.

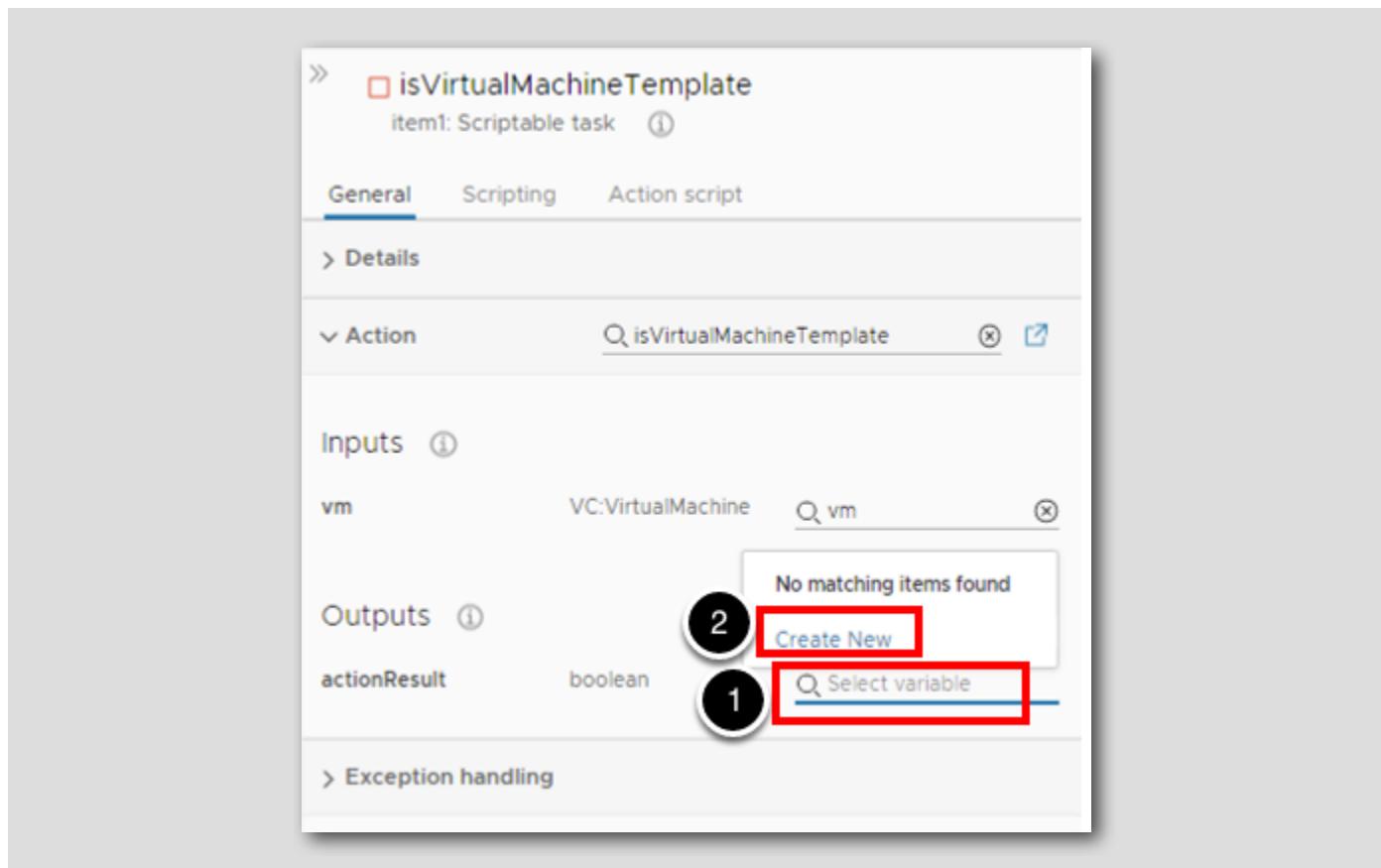
1. Under Inputs, click the vm input where it says *Select variable*
2. Click Create New

Create the new input parameter



1. Select the **Input** radio picker
2. Click the **CREATE** button

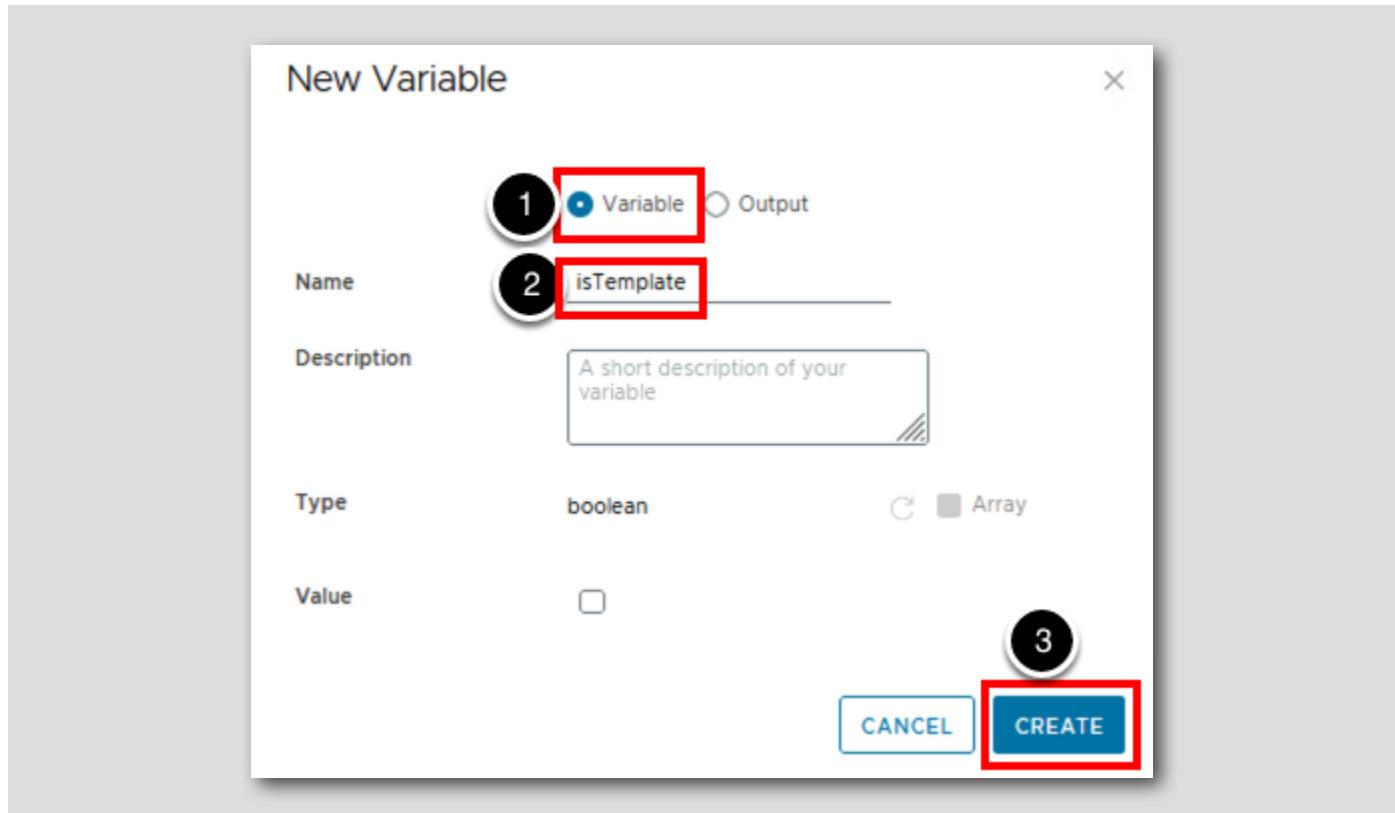
## Add a new variable



Our action also provides an output, with a TRUE/FALSE value based on whether the Virtual Machine is a template. Let's map the output to a variable within the workflow so we can see the end result of the action at the workflow level.

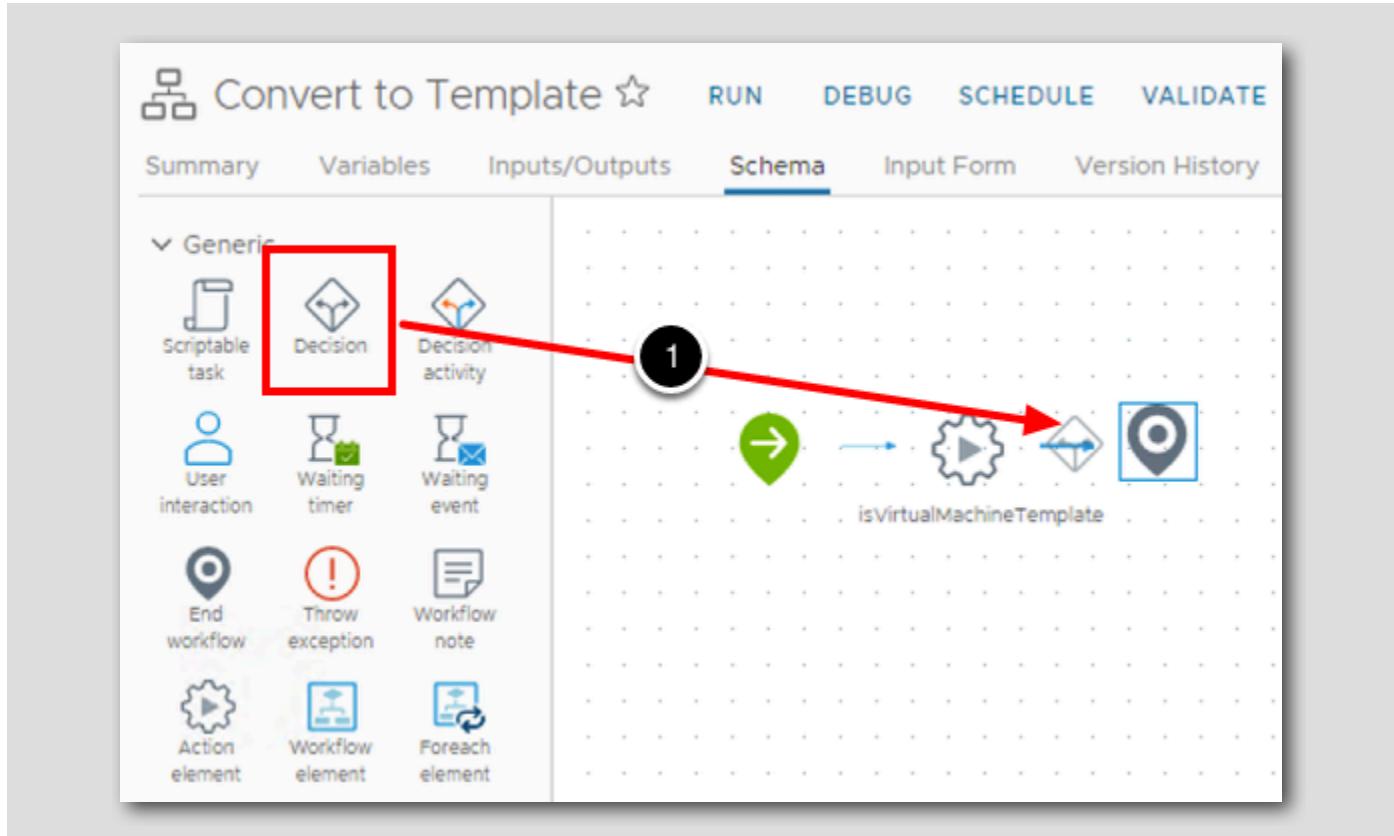
1. Under Outputs, click the `actionResult` output where it says *Select variable*
2. Click *Create New*

Create the new variable



1. Select Variable radio picker
2. For Name, enter isTemplate
3. Click the CREATE button

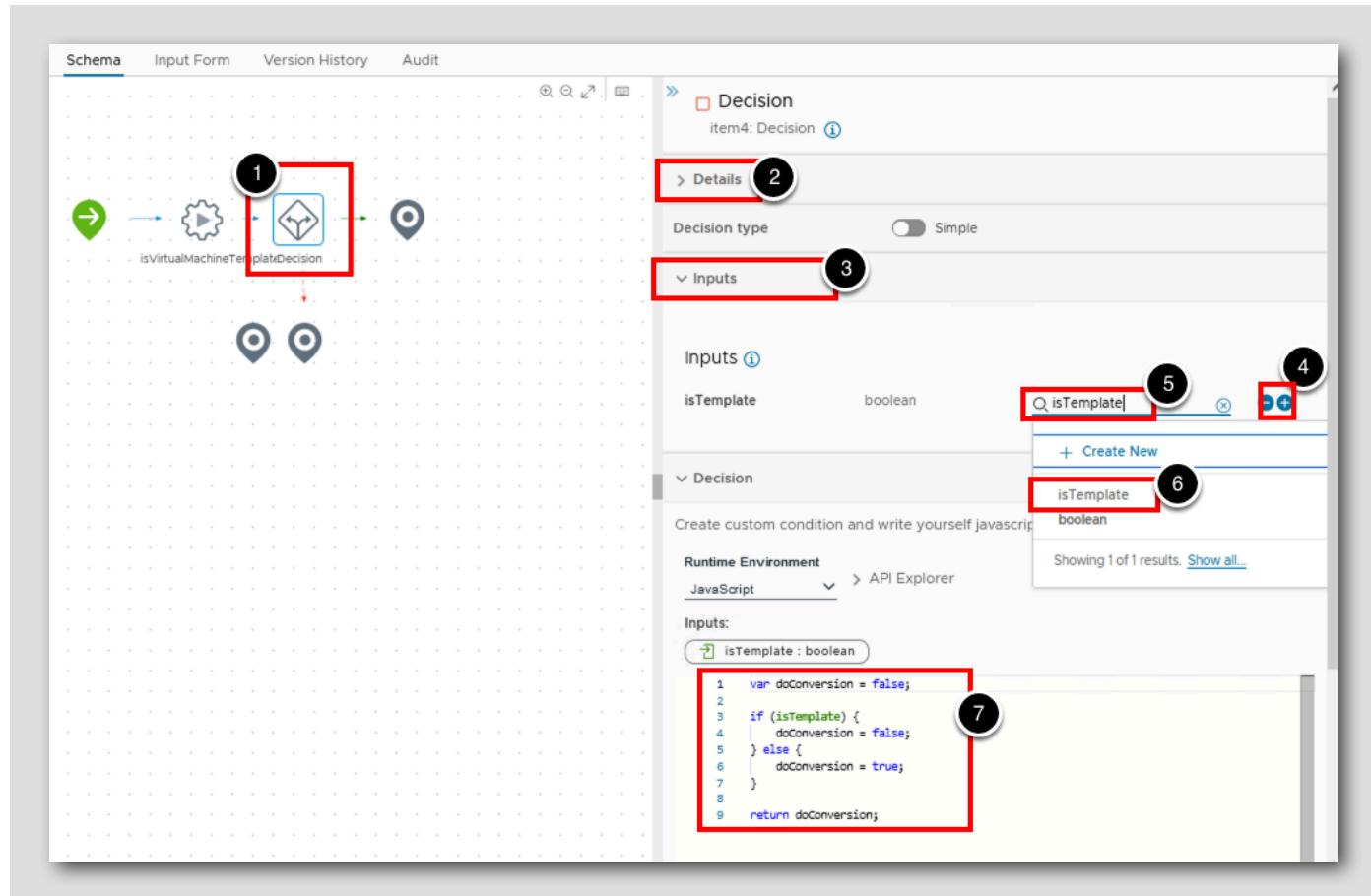
## Add a decision



Since our action can return two possible values (TRUE/FALSE), we can tell Orchestrator to perform different steps based on the value returned. So if our action returns a value of FALSE, meaning the Virtual Machine is not a template we want to power off the Virtual Machine and convert it to a template. If the action returns TRUE, we already have a template and don't need to convert it. Let's add a decision element to our workflow to create two different branches for Orchestrator to follow based on the value returned by our action.

1. Drag a Decision element into the schema after the isVirtualMachineTemplate action element

## Define decision details



1. Select the **Decision** element
2. Collapse the **Details** pane
3. Expand the **Inputs** pane
4. In the **Inputs** pane, click the **+** button to add an input
5. In the **Inputs** pane, click the field that says *Select variable*
6. Select the **isTemplate** variable
7. In the **Decision** pane, add the following code:

```

var doConversion = false;

if (isTemplate) {
    doConversion = false;
} else {
    doConversion = true;
}

return doConversion;

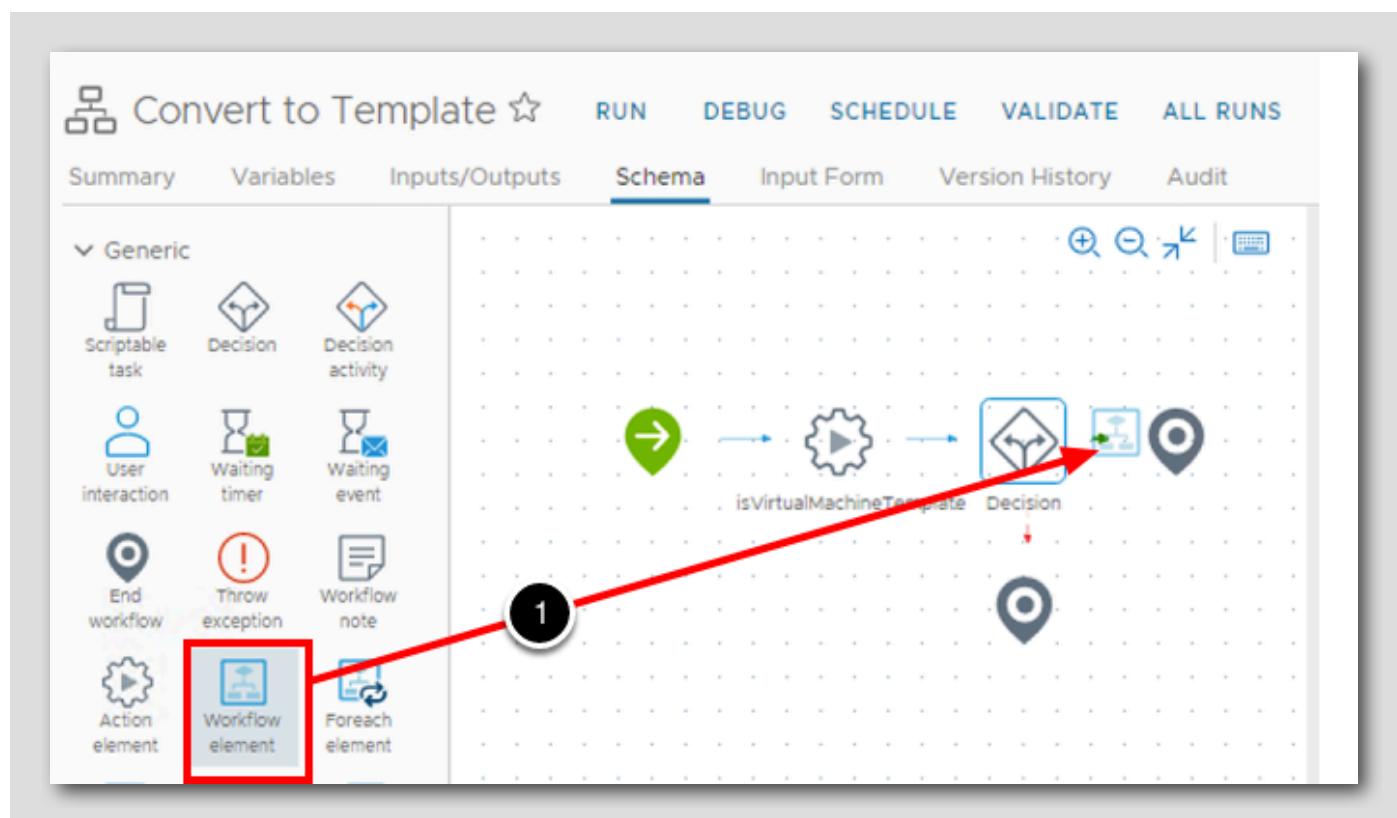
```

Here, we've defined the decision to return true and go down the green path if the Virtual Machine is not a template, where we will convert the Virtual Machine to a template.

If the Virtual Machine is already a template, it will return false and go down the red path, which will skip the Virtual Machine template conversion.

## Add a workflow element

[148]



A Virtual Machine cannot be marked as a template until it is powered off, so let's add the workflow to power off the machine first. We could create this workflow ourselves, however the vCenter plugin provided with Orchestrator has a workflow that can perform this step for us. We will therefore reuse the existing workflow to save effort and complexity.

1. Drag a Workflow element into the schema after the Decision element

## Define the workflow details

[149]

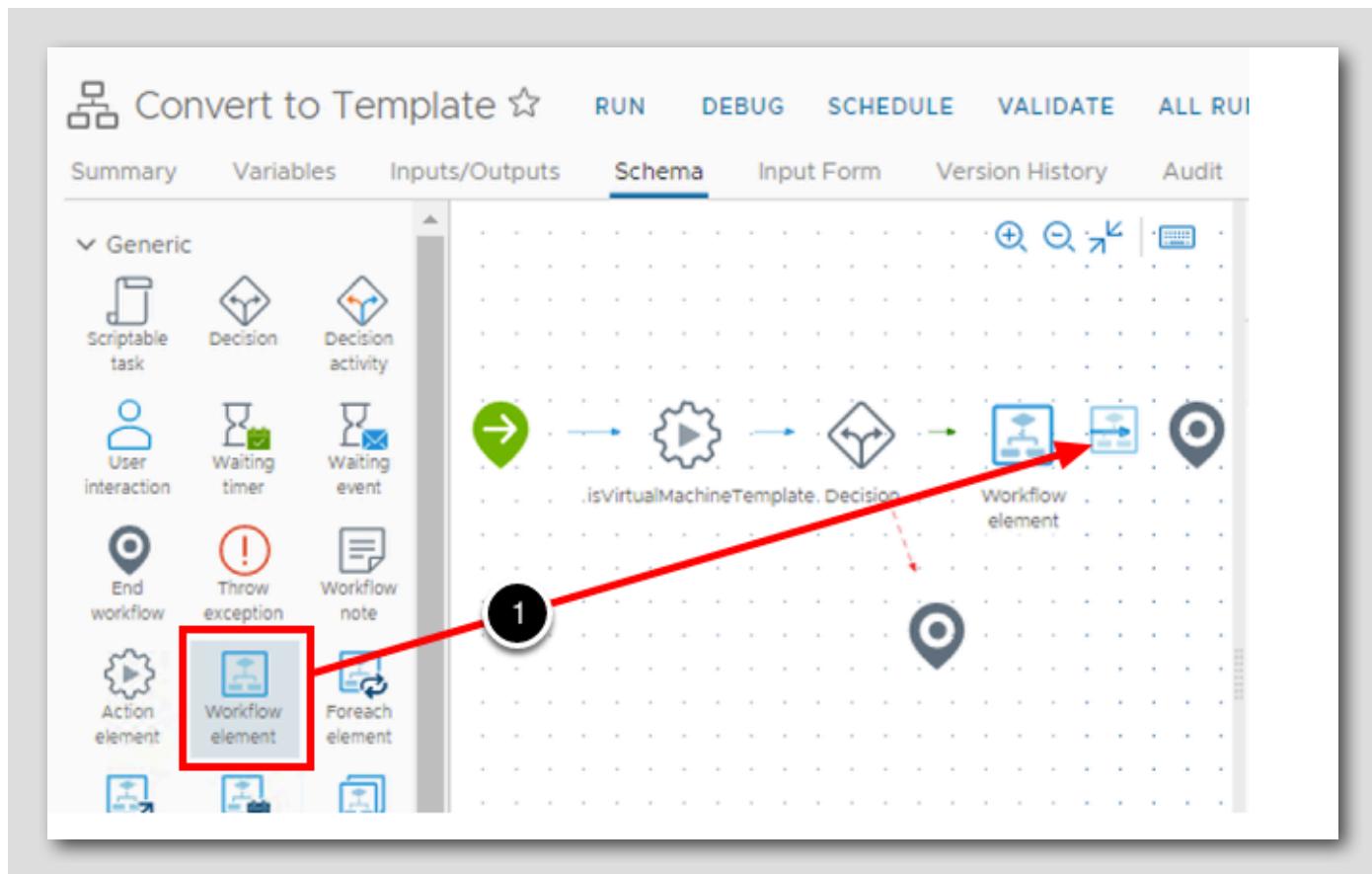
The screenshot shows the VMware Orchestrator Schema editor. On the left, the 'Schema' tab is active, displaying a workflow diagram. The diagram starts with a green arrow, followed by a gear icon, then a decision diamond labeled 'isVirtualMachineTemplateDecision'. After the decision, there is a red box around a blue square icon labeled 'Workflow element', which is connected to a grey location pin icon. Step 1 is indicated by a black circle with the number 1. Below the schema, the 'Workflow element' details pane is open, showing:

- Name:** Workflow element
- Next element:** item0: End
- Business status:** checked
- Description:** (empty)

Step 2 is indicated by a black circle with the number 2, pointing to the search bar in the 'Workflow' section where 'power off' is typed. Step 3 is indicated by a black circle with the number 3, pointing to the selected workflow item 'Power off virtual machine and wait' in the search results list. The list also includes 'vRO Power Off VM' and other items. The search results pane shows 'Showing 2 of 2 results. [Show all...](#)'.

1. Select the Workflow element
2. In the Workflow pane, where it says *Select workflow*, type **power off**
3. Select the Power off Virtual Machine and wait workflow

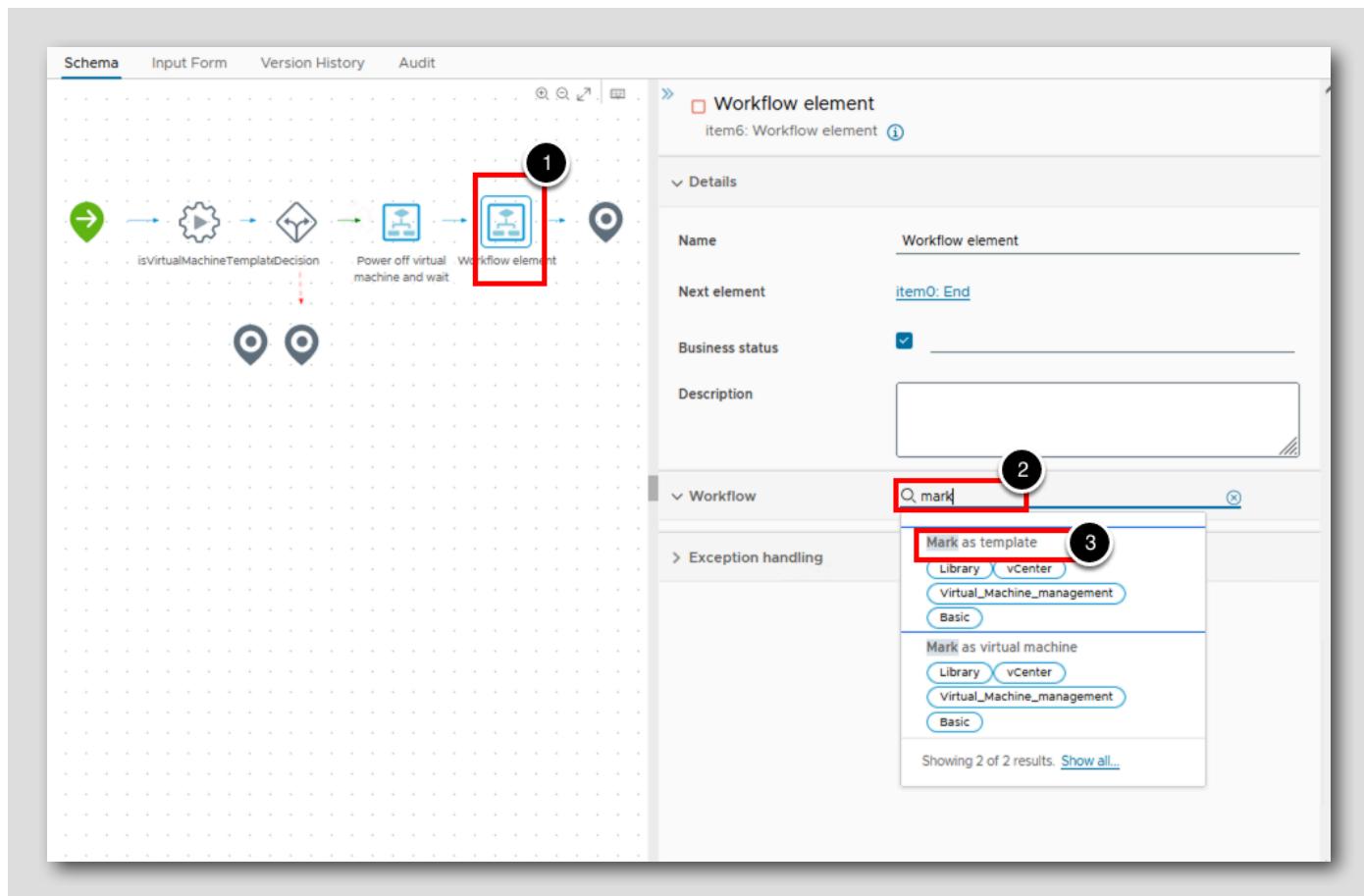
## Add a workflow element



Lastly, let's add the workflow to convert the Virtual Machine to a template. We will again reuse an existing workflow provided by the vCenter plugin for simplicity.

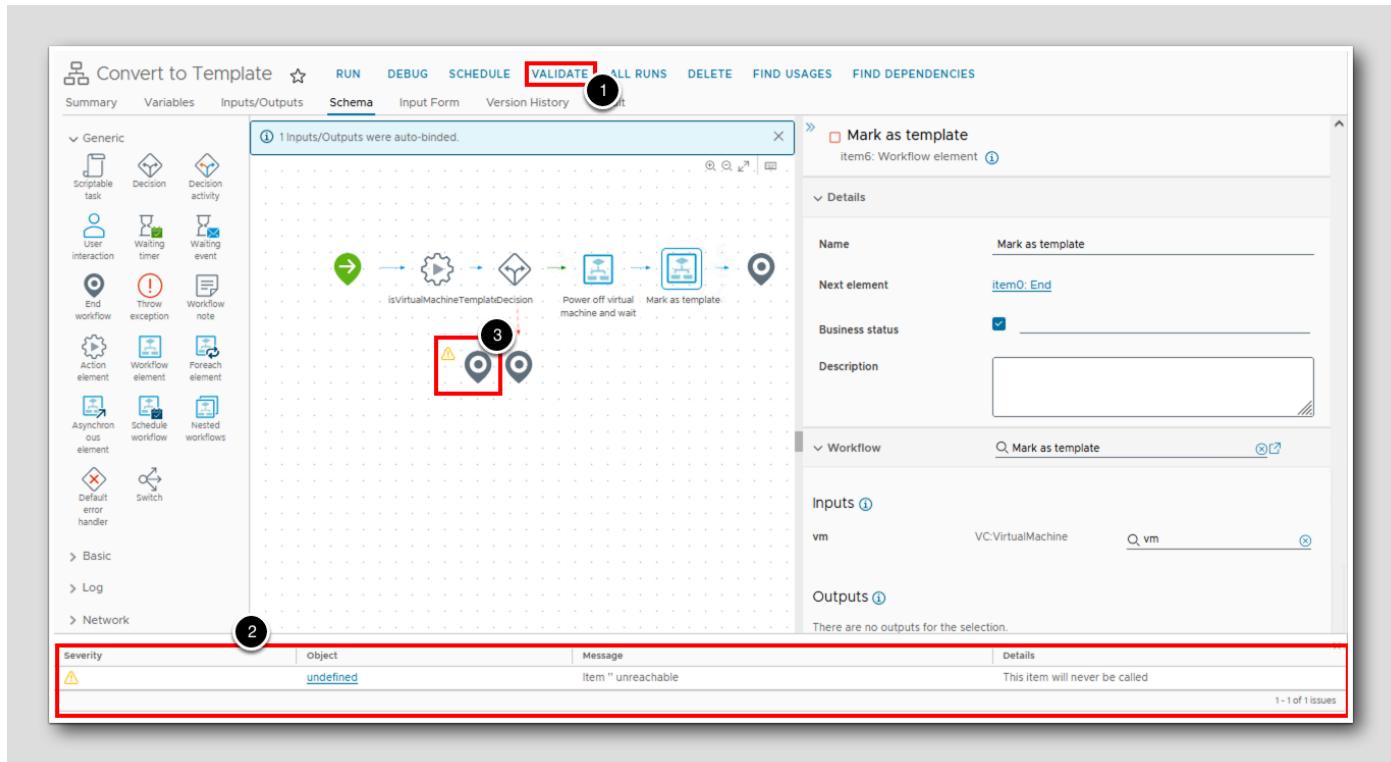
1. Drag a Workflow element into the schema after the previous Workflow element

## Define the workflow details



1. Select the **Workflow element**
2. In the **Workflow** pane, where it says *Select workflow*, type **mark**
3. Select the **Mark as template** workflow

## Validate the workflow

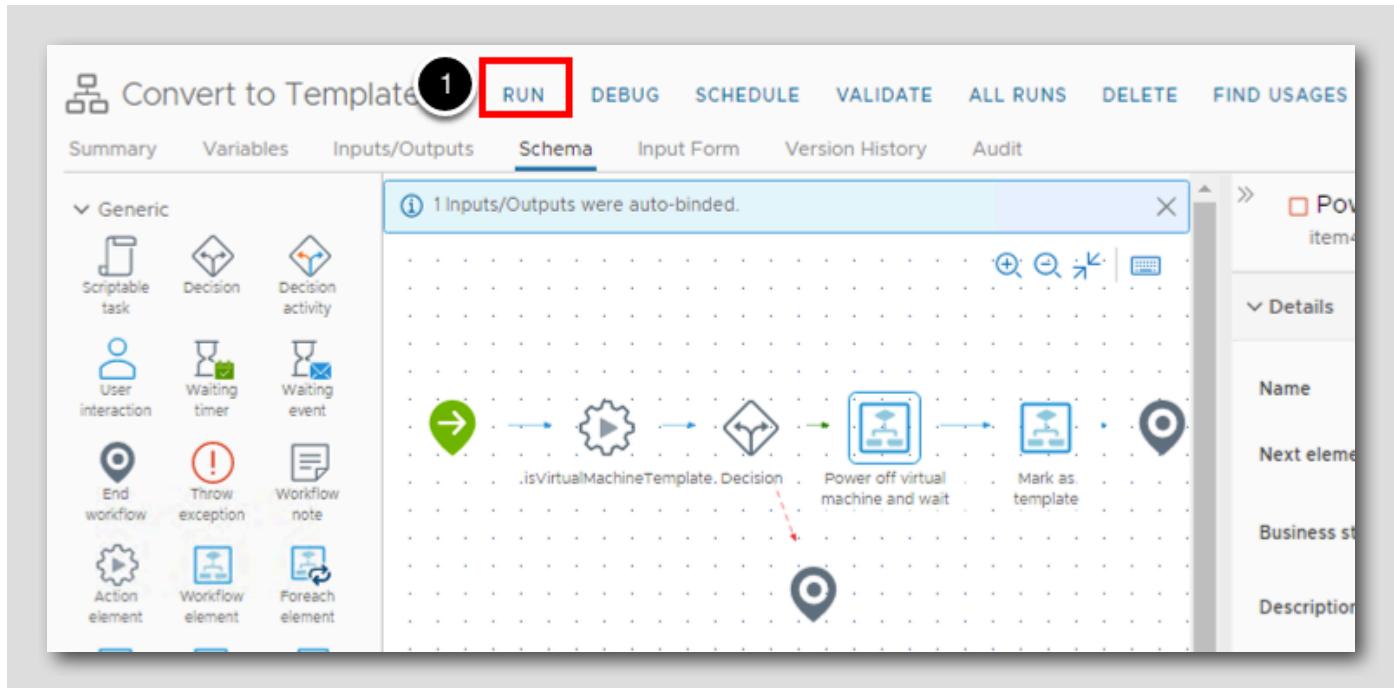


Before we run our workflow its always a good idea to validate the workflow to check for any unintended mistakes or issues.

1. Click **VALIDATE**
2. After a short pause while the validation completes, the validation results will display at the bottom of the screen. If you receive a prompt box stating validation completed successfully you can move on to the Review Workflow and Run It section.
3. We have one validation warning relating to the unused End Workflow element in our schema. Click on the element and select the red X that appears in the top right corner of the element (not shown) to remove it.

If you receive a prompt box stating "validation is successful" you can move on to the Review the workflow and run it section of this lab manual without performing step 3 above.

Review the workflow and run it

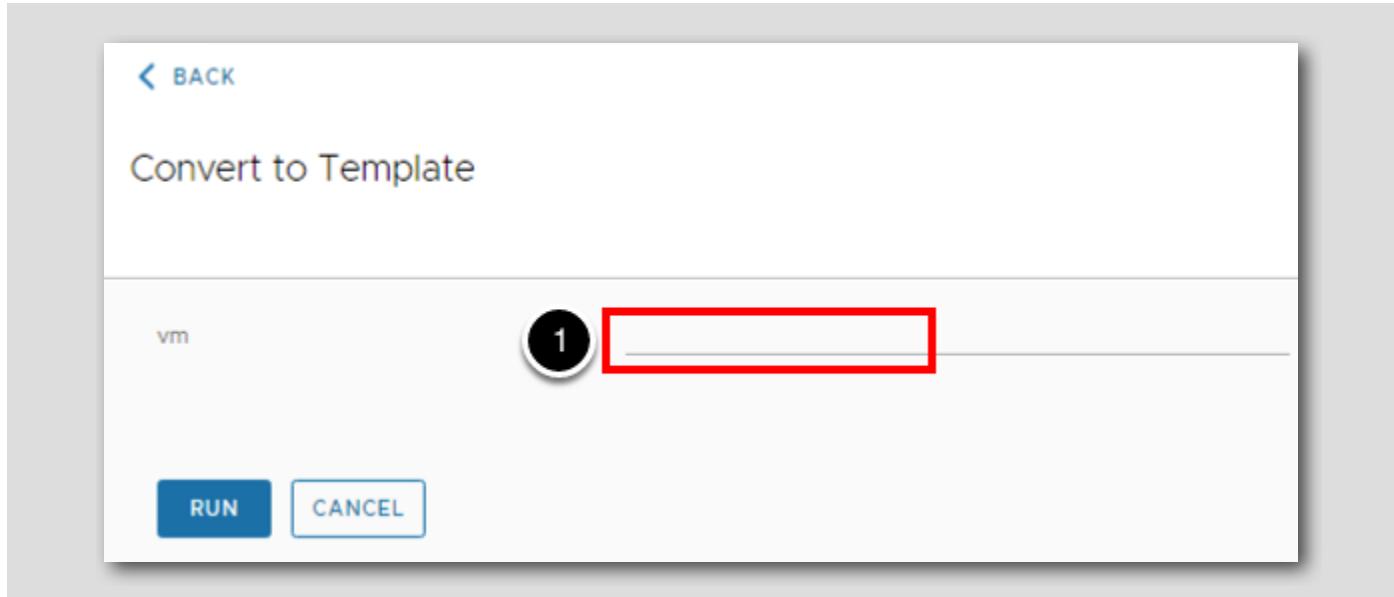


Let's finally run the workflow and confirm that it performs as intended.

1. Click RUN to test the workflow
2. When prompted, click SAVE AND RUN (not shown)

If you noticed the Info messages that appeared regarding auto-bound parameters after we added each of the workflow elements, in this case that refers to the fact that the Power off Virtual Machine and wait workflow and the Mark as template workflow each have a vm input parameter that was automatically set to the vm input parameter of the Convert to Template workflow because they share the exact same name.

Set the Virtual Machine input value



1. Click the **vm** field

## Select the Virtual Machine

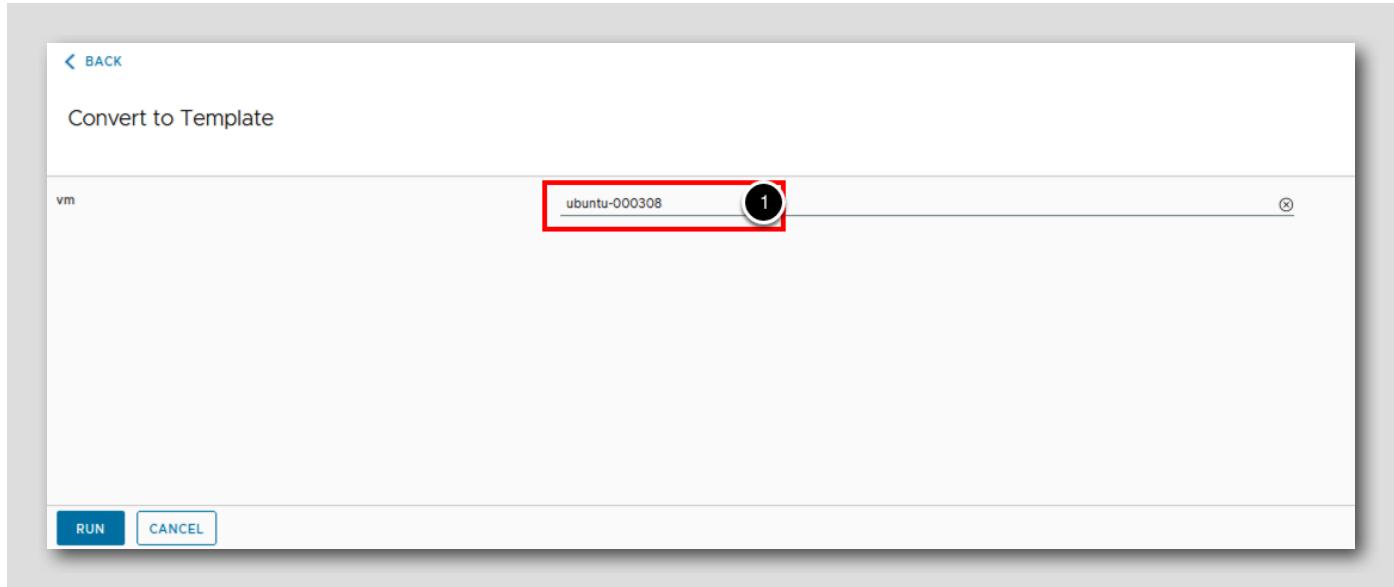
|                             |                                       |
|-----------------------------|---------------------------------------|
| <b>id</b>                   | vm-11009                              |
| Orchestrator unique ID      | vm-11009                              |
| Annotation                  |                                       |
| VM Connection State         | connected                             |
| Consumed Host CPU           | 59 MHz                                |
| guestHeartbeatStatus        | green                                 |
| IP address                  | 10.64.12.10                           |
| Active Guest Memory         | 0 MB                                  |
| CPU                         | 1 vCPUs                               |
| Product Vendor              |                                       |
| Memory Overhead             |                                       |
| Guest OS                    | Ubuntu Linux (64-bit)                 |
| dunesId                     | vcenter-mgmt.vcf.sddc.lab.id.vm-11009 |
| <b>VM Template</b>          | false                                 |
| sdkId                       | vcenter-mgmt.vcf.sddc.lab/vm-11009    |
| Used Storage                | 13200 MB                              |
| name                        | ubuntu-000308                         |
| VMware Tools Version Status | guestToolsUnmanaged                   |
| VM Version                  | vmx-21                                |
| alarmActionsEnabled         | true                                  |
| overallStatus               | green                                 |

**CANCEL**    **SELECT**

We'll select a Virtual Machine that we know is not a template.

1. Navigate to vSphere vCenter Plugin > <https://vcenter-mgmt.vcf.sddc.lab:443/sdk> > Datacenters > mgmt-datacenter-01 > vm > Workloads > ubuntu-000308
2. Click SELECT

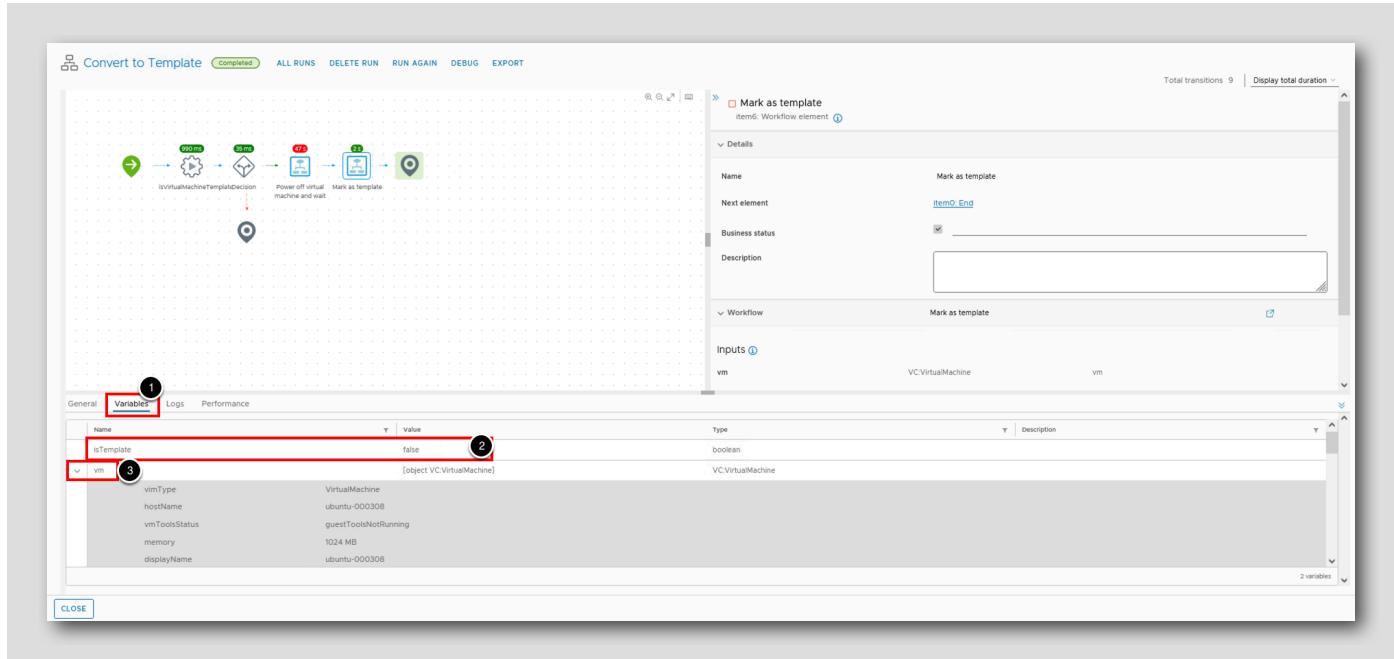
## Run the workflow



The machine selected in the previous step should be visible in the vm field.

1. Click the RUN button

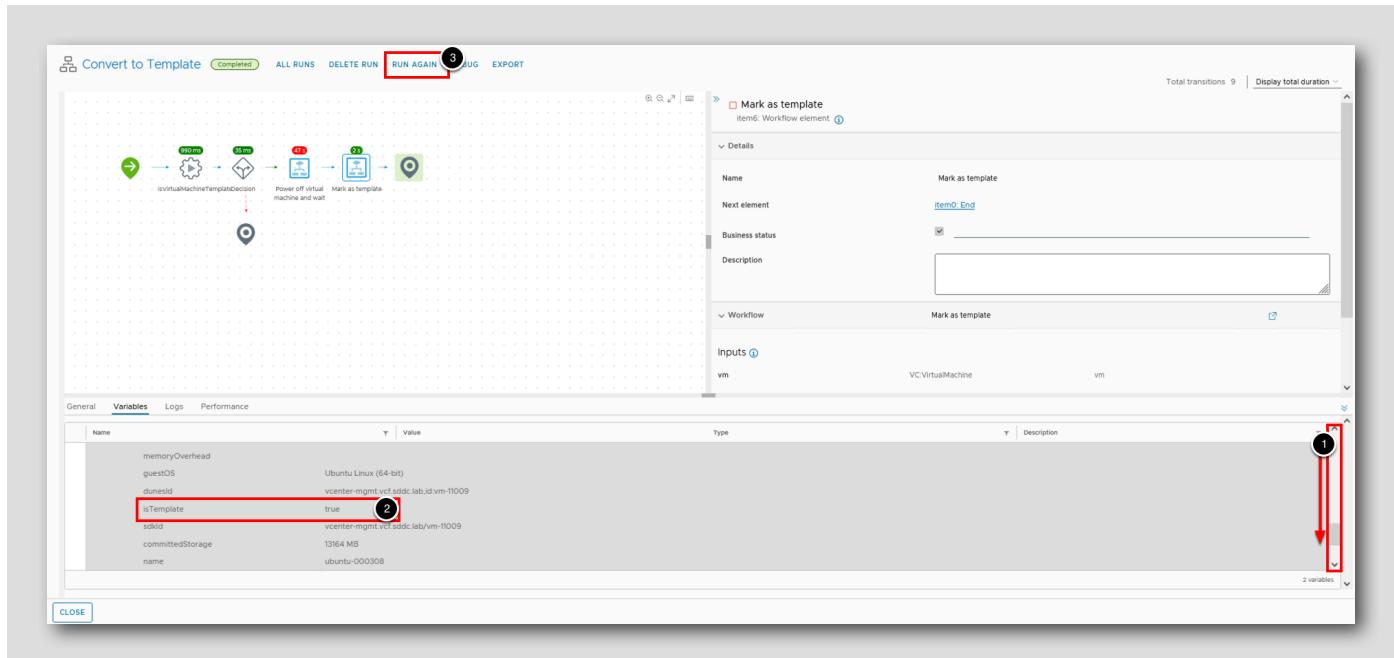
## Inspect the workflow run results



Once the workflow run has completed:

1. Click the **Variables** tab
2. The **isTemplate** variable is set to false, indicating that the Virtual Machine was not a template at the start of this workflow run
3. Expand the **vm** variable to view its properties

## Inspect the Virtual Machine properties

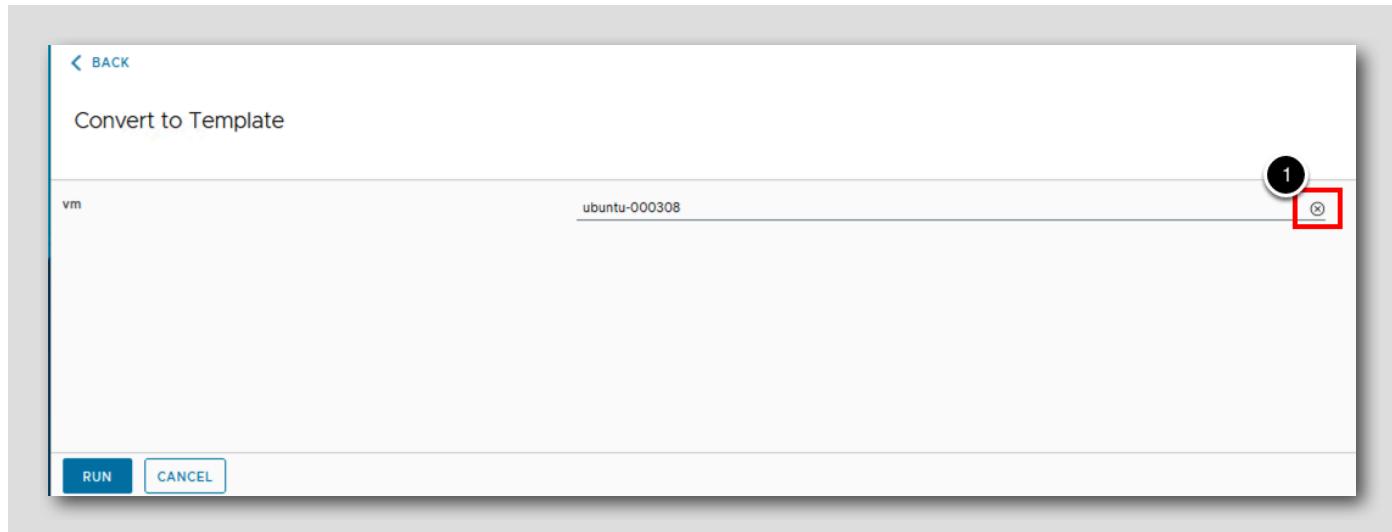


1. Scroll down to find the **vm** input parameter's **VM Template** property
2. Confirm that **isTemplate** is true, indicating that the Virtual Machine is now a template
3. Click **RUN AGAIN**

We also want to see what happens when we run the workflow on a Virtual Machine that is already marked as a template.

Remove the previous Virtual Machine input

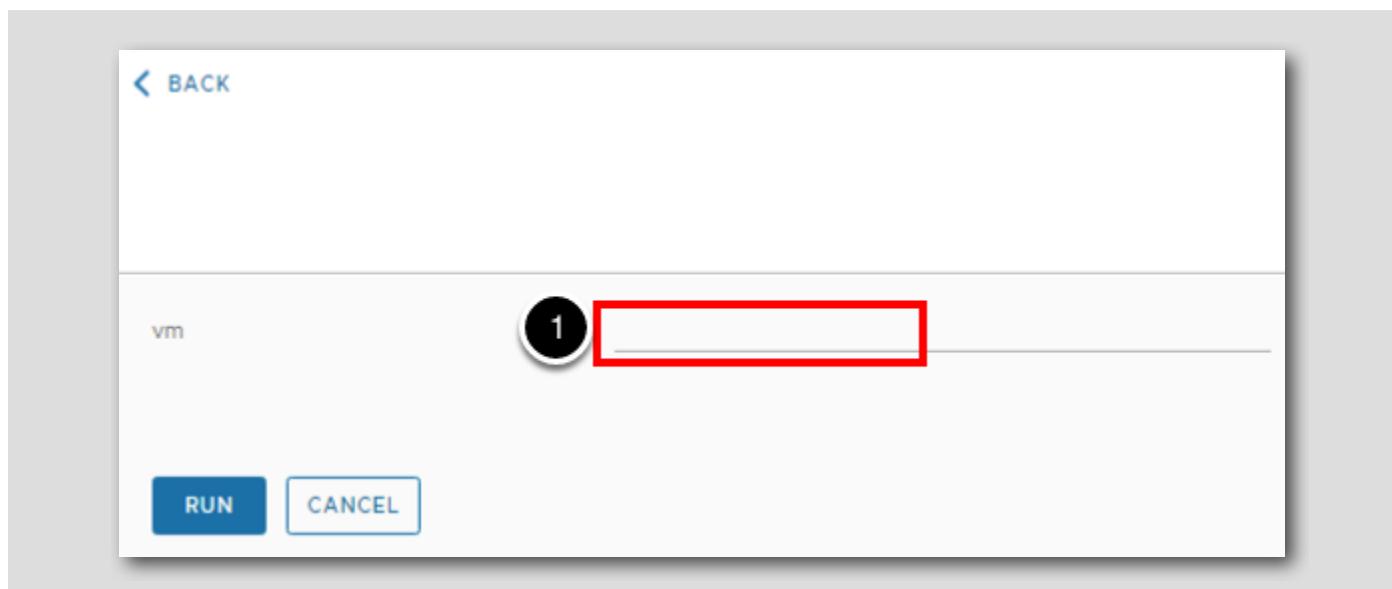
[159]



1. Remove the vm value from the previous action run

Set the Virtual Machine input

[160]



1. Click the vm field

## Select Virtual Machine

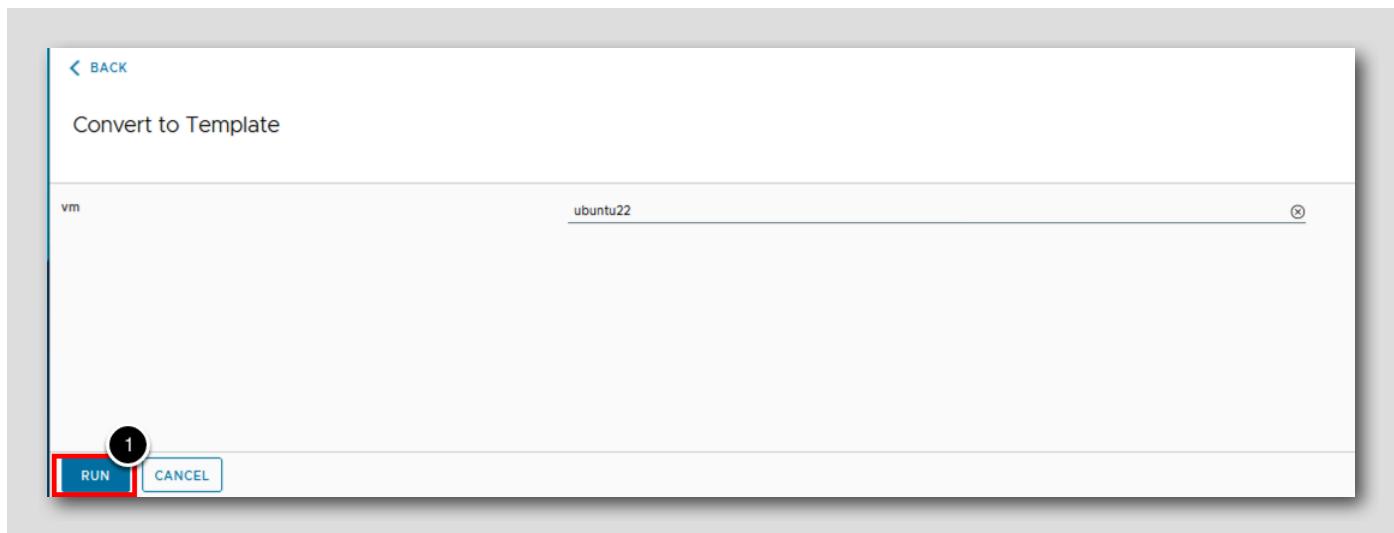
Select VC:VirtualMachine

The screenshot shows the vSphere vCenter Server interface. On the left, there is a navigation tree under 'vSphere vCenter Server'. The 'Datacenters' section is expanded, showing 'mgmt-datacenter-01' which contains 'host', 'vm', 'Templates', 'Workloads', and several management domains. Under 'Templates', 'ubuntu22' is selected (indicated by a red box and a circled '1'). Other options like 'windows2022' are also listed. On the right, a detailed properties panel for 'ubuntu22' is displayed, listing various configuration details such as SDK Type, DNS Name, VMware Tools Status, Consumed Host Memory, Display name of this virtual machine, Non-shared Storage, configStatus, type, 128-bit SMBIOS UUID, Provisioned Storage, instanceId, Memory, Power State, id, Orchestrator unique ID, Annotation, VM Connection State, Consumed Host CPU, guestHeartbeatStatus, and Active Guest Memory. At the bottom right of the properties panel, there are 'CANCEL' and 'SELECT' buttons, with 'SELECT' being highlighted by a red box and circled '2'.

|  |                                      |
|--|--------------------------------------|
| SDK Type                                 | VirtualMachine                       |
| DNS Name                                 | ubuntu22                             |
| VMware Tools Status                      | guestToolsNotRunning                 |
| Consumed Host Memory                     | 0 MB                                 |
| Display name of this virtual machine     | ubuntu22                             |
| Non-shared Storage                       | 12188 MB                             |
| configStatus                             | green                                |
| type                                     | VirtualMachine                       |
| 128-bit SMBIOS UUID of a virtual machine | 422b06f5-fd44-b82b-708b-f30504a51079 |
| Provisioned Storage                      | 53905 MB                             |
| instanceId                               | 502ba23f-1690-65af-268b-671f6dba8d02 |
| Memory                                   | 2048 MB                              |
| Power State                              | poweredOff                           |
| id                                       | vm-6002                              |
| Orchestrator unique ID                   | vm-6002                              |
| Annotation                               |                                      |
| VM Connection State                      | connected                            |
| Consumed Host CPU                        | 0 MHz                                |
| guestHeartbeatStatus                     | gray                                 |
| Active Guest Memory                      | 0 MB                                 |

1. Navigate to vSphere vCenter Plugin > <https://vcenter-mgmt.vcf.sddc.lab:443/sdk> > Datacenters > mgmt-datacenter-01 > vm > Templates > ubuntu22
2. Click SELECT

## Run the workflow



The machine selected in the previous step should be visible in the vm field.

1. Click the RUN button

## Inspect the workflow run results

Convert to Template Completed

ALL RUNS DELETE RUN RUN AGAIN DEBUG EXPORT

isVirtualMachineTemplateDecision (301 ms) → Power off virtual machine and wait (7 ms) → Mark as template

Power off virtual machine and wait  
Mark as template

1 Variables 2 3

| Name       | Type              |
|------------|-------------------|
| isTemplate | boolean           |
| vm         | VC:VirtualMachine |

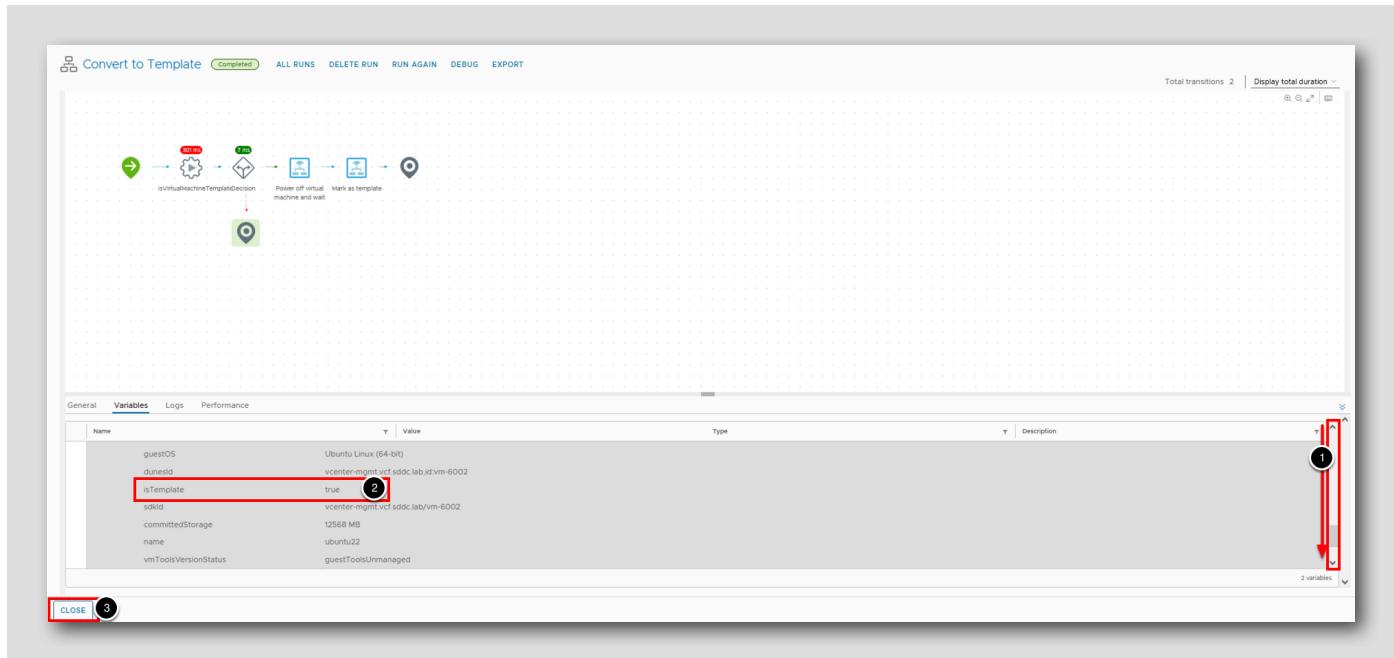
isTemplate: true [object VC:VirtualMachine]  
vm:  
  vimType: VirtualMachine  
  hostName: ubuntu22  
  vmToolsStatus: guestToolsNotRunning  
  memory: 2048 MB  
  displayName: ubuntu22

CLOSE

Once the workflow run has completed:

1. Click the **Variables** tab
2. The **isTemplate** variable is set to true, indicating that the Virtual Machine was already a template at the start of this workflow run
3. Expand the **vm** variable to view its properties

## Inspect the Virtual Machine properties



1. Scroll down to find the vm input parameter's isTemplate property
2. Confirm that isTemplate is true, indicating that the Virtual Machine is still a template
3. Click CLOSE here for the workflow run

## Close the workflow

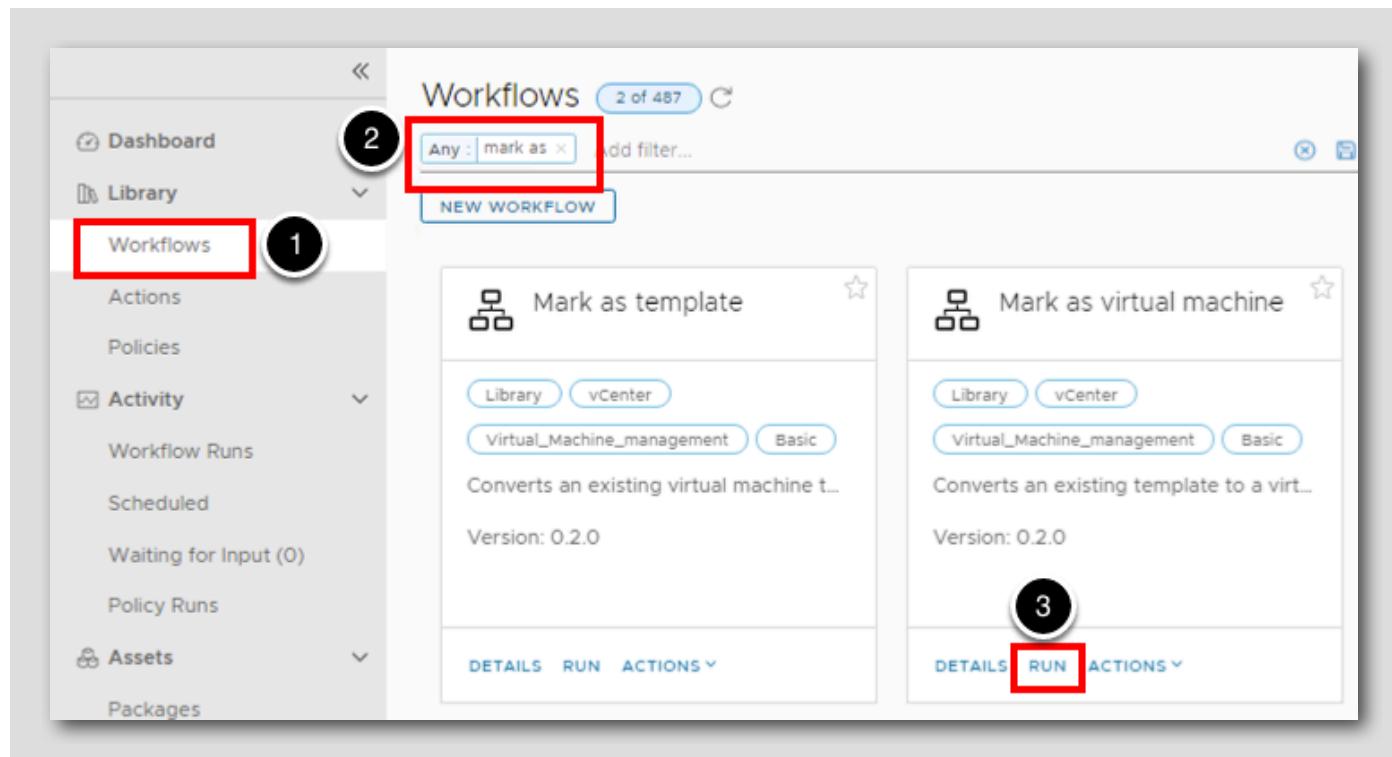
The screenshot shows the VMware Aria Automation Orchestrator interface with the following details:

- Workflow name:** Convert to Template
- ID:** 887d00dd-5262-45f9-9ed9-0d324e649bdf
- Version:** 0.0.0
- Tags:** web-root, Enter a new tag
- Folder:** /web-root, SELECT FOLDER
- Server restart behavior:** Resume workflow run
- Resume workflow from failed behavior:** System default
- Description:** Add description

At the bottom, there are three buttons: SAVE, VERSION, and CLOSE. The CLOSE button is highlighted with a red box and has a circled '1' above it.

1. Click on CLOSE to close the whole workflow.

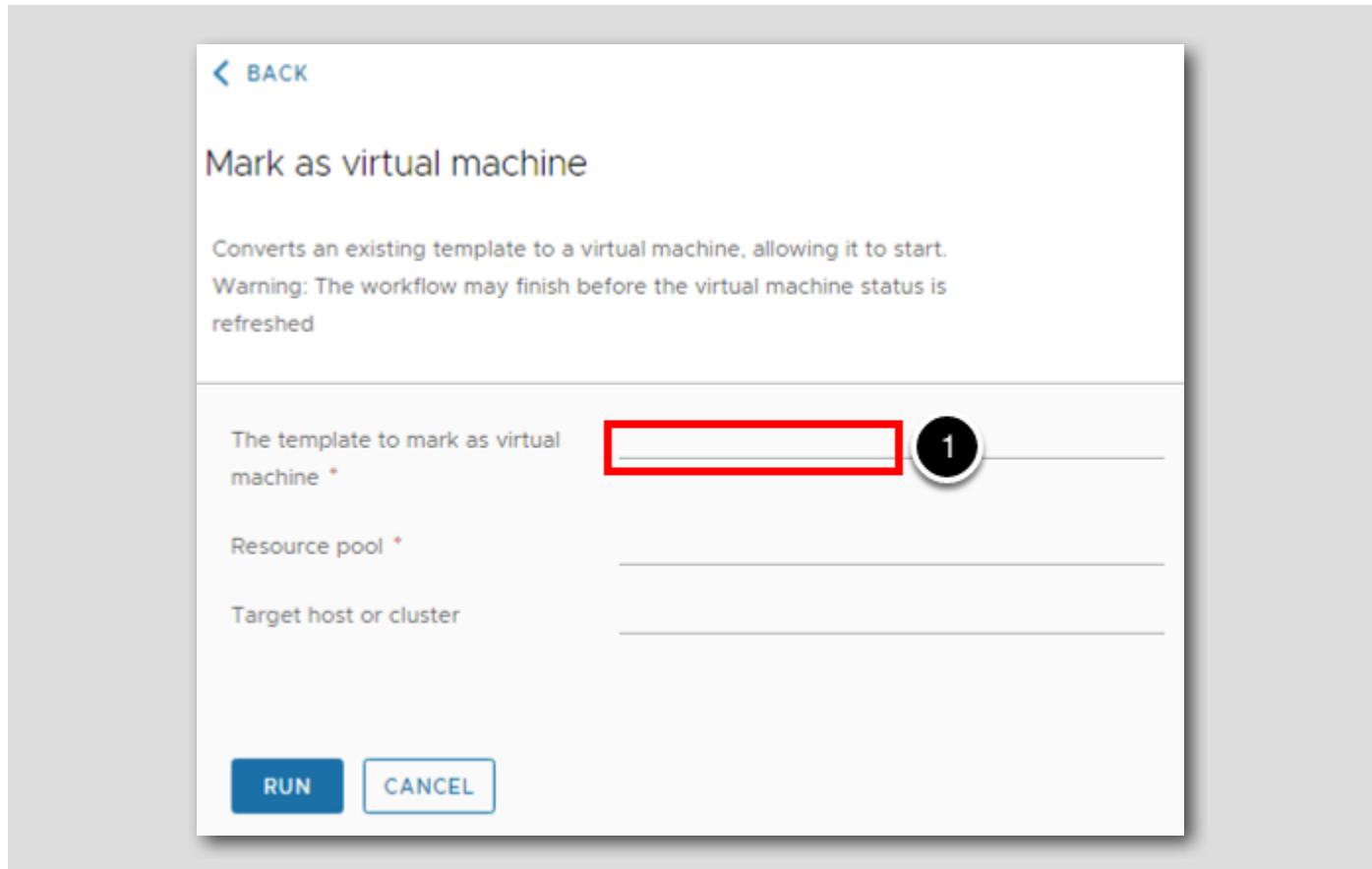
## Run the workflow



Before we proceed to the next exercise, let's revert `ubuntu-0003` to a powered on Virtual Machine again.

1. Click the **Workflows** tab
2. In the filter, enter **mark as** and press return
3. On the **Mark as Virtual Machine** workflow card, click **RUN**

Select the Virtual Machine template field



1. Click the field The template to mark as Virtual Machine

## Select Virtual Machine

Select VC:VirtualMachine

The tree view shows:

- vSphere vCenter Server
  - https://vcenter-mgmt.vcf.sddc.lab:443/sdk (VirtualCenter-8.0.2.0)
    - Datacenters
      - mgmt-datacenter-01
        - host
        - vm
          - Development
          - Discovered virtual machine
          - Templates
          - Workloads
            - ubuntu-000308 (1)
            - ubuntu-001202
            - windows-000906
        - domain-fd-edge
        - domain-fd-mcm
        - domain-fd-mgmt
        - domain-fd-nsx
      - datastore
      - network

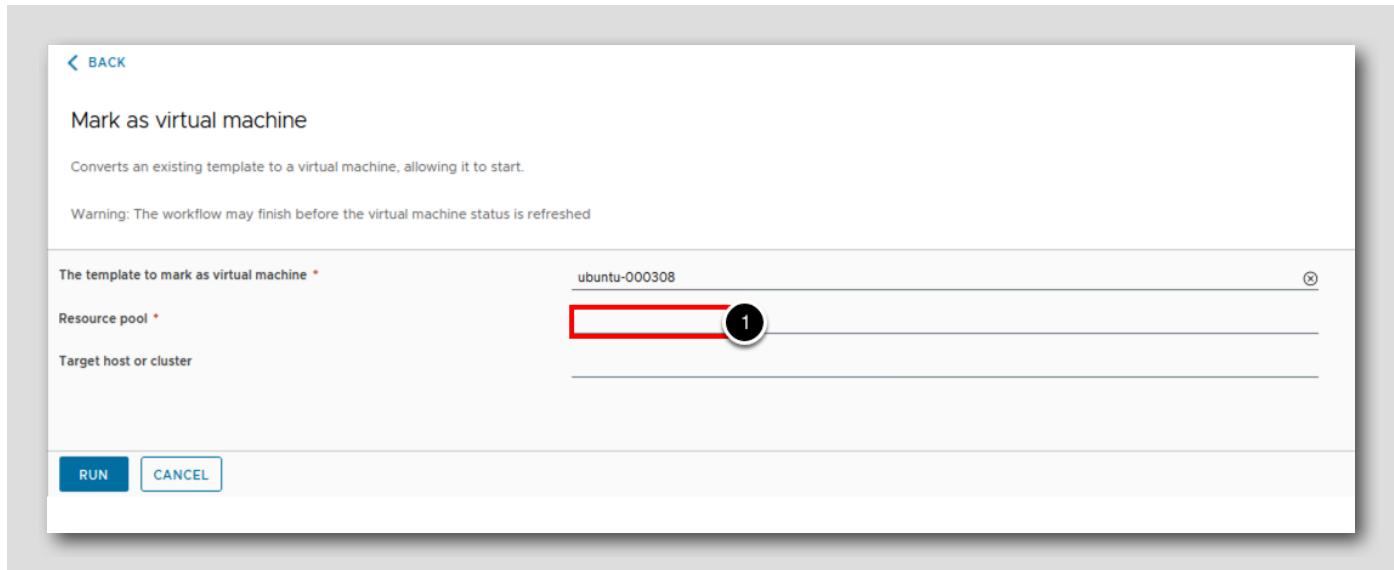
The properties table for 'ubuntu-000308' includes the following details:

|  |                                      |
|--|--------------------------------------|
| SDK Type                                 | VirtualMachine                       |
| DNS Name                                 | ubuntu-000308                        |
| VMware Tools Status                      | guestToolsRunning                    |
| Consumed Host Memory                     | 0 MB                                 |
| Display name of this virtual machine     | ubuntu-000308                        |
| Non-shared Storage                       | 12708 MB                             |
| configStatus                             | green                                |
| type                                     | VirtualMachine                       |
| 128-bit SMBIOS UUID of a virtual machine | 422b96d7-a74e-1886-d9d-4954d8513b96  |
| Provisioned Storage                      | 52680 MB                             |
| instanceId                               | 502b6de2-811c-3464-22cf-f3b3cef96216 |
| Memory                                   | 1024 MB                              |
| Power State                              | poweredOn                            |
| id                                       | vm-11009                             |
| Orchestrator unique ID                   | vm-11009                             |
| Annotation                               |                                      |
| VM Connection State                      | connected                            |
| Consumed Host CPU                        | 59 MHz                               |
| guestHeartbeatStatus                     | green                                |
| IP address                               | 10.64.12.10                          |

At the bottom right are 'CANCEL' and 'SELECT' buttons. The 'SELECT' button is highlighted with a red box and labeled '2'.

1. Navigate to vSphere vCenter Plugin > https://vcenter-mgmt.vcf.sddc.lab:443/sdk > Datacenters > mgmt-datacenter-01 > vm > Workloads > ubuntu-000308
2. Click SELECT

Select the Virtual Machine template field



1. Click the field Resource pool

## Select the Resource Pool

Select VC:ResourcePool

| Resources              |   |
|------------------------|---|
| SDK Type               | ResourcePool                                |
| CPU reservation        | 15438 MHz                                   |
| id                     | resgroup-4002                               |
| dunesId                | vccenter-mgmt.vcf.sddc.lab:id:resgroup-4002 |
| sdkId                  | vccenter-mgmt.vcf.sddc.lab/resgroup-4002    |
| name                   | Resources                                   |
| Orchestrator unique ID | resgroup-4002                               |
| type                   | ResourcePool                                |
| Memory Reservation     | 16670 MB                                    |

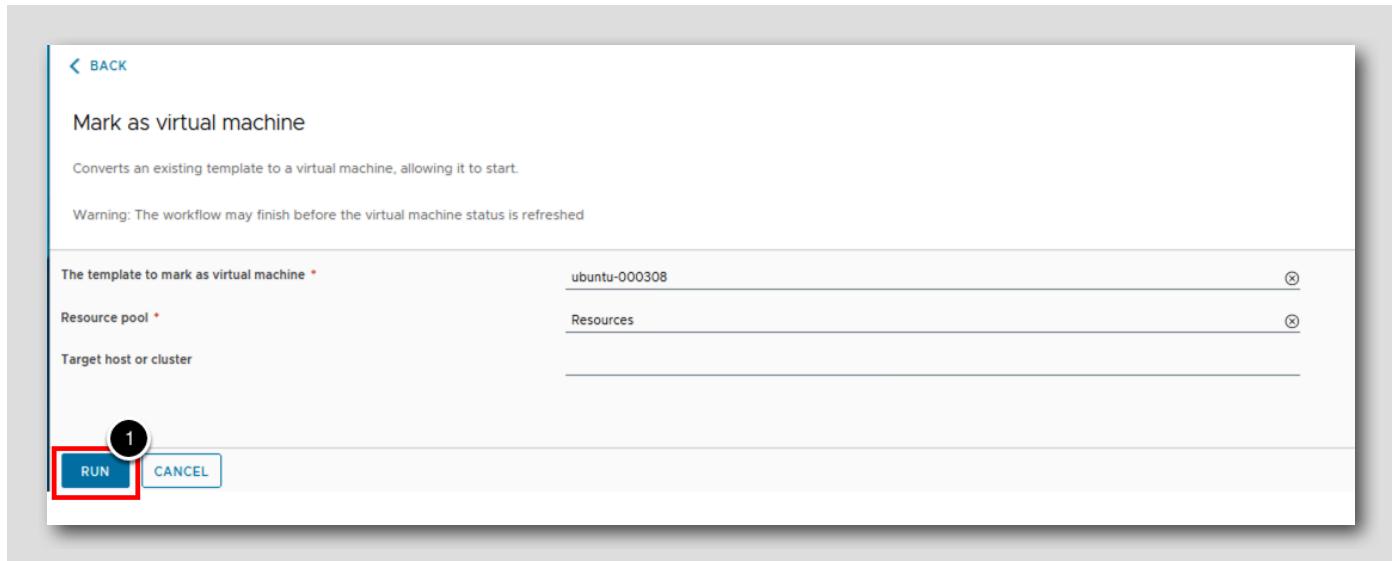
**1**

**2**

1. Navigate to vSphere vCenter Plugin > https://vcenter-mgmt.vcf.sddc.lab:443/sdk > Datacenters > mgmt-datacenter-01 > host > wld-cluster-01 > Resources
2. Click SELECT

## Run the workflow

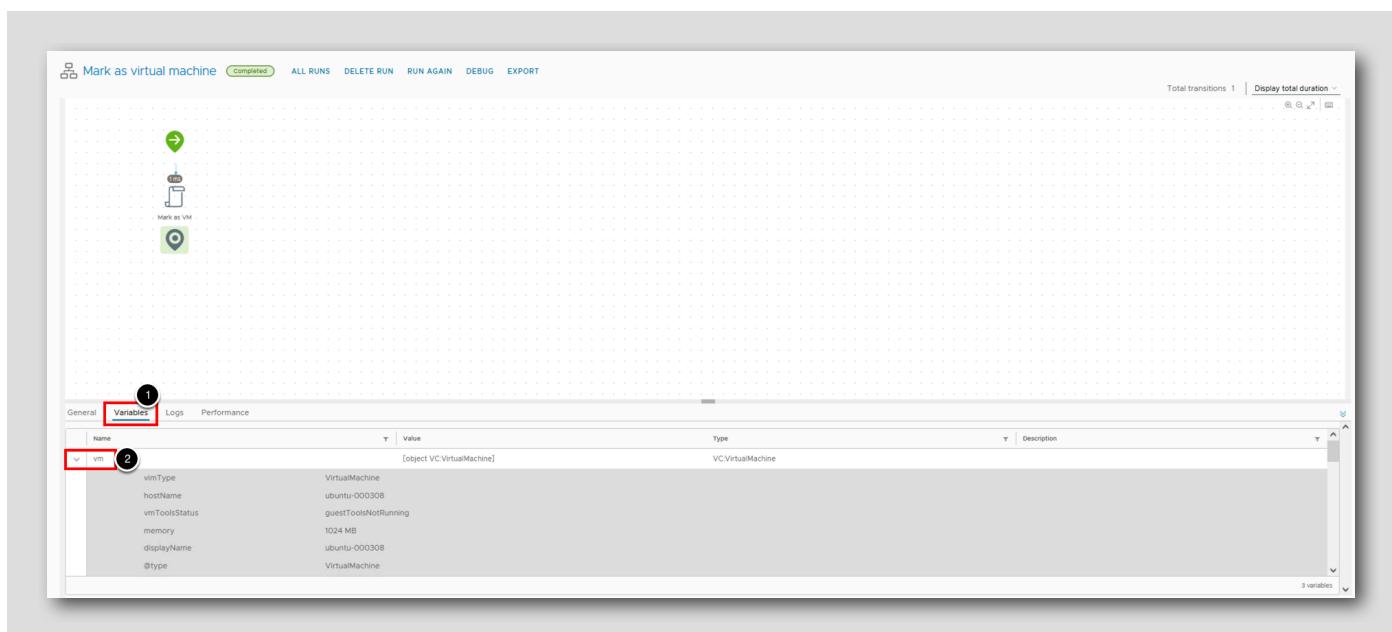
[171]



1. Click RUN

## Inspect the workflow run results

[172]



Once the workflow run has completed:

1. Click the **Variables** tab
2. Expand the **vm** variable to view its properties

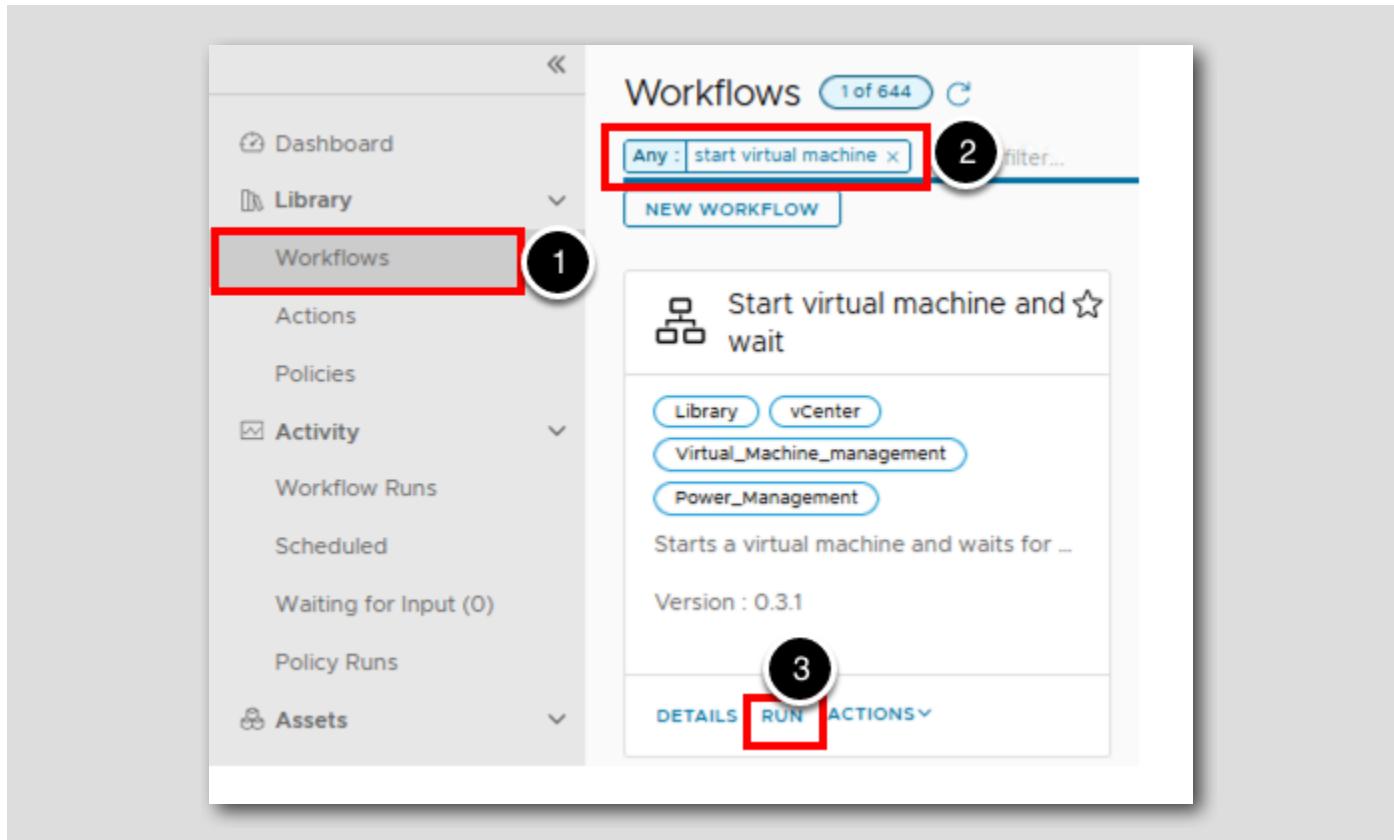
## Inspect the `isTemplate` property

[173]

| Name              | Type         | Description                           |
|-------------------|--------------|---------------------------------------|
| productVendor     |              |                                       |
| memoryOverhead    |              |                                       |
| guestOS           |              | Ubuntu Linux (64-bit)                 |
| dnsesId           |              | vcenter-mgmt.vcf.sddc.lab.id.vm-11009 |
| <b>isTemplate</b> | <b>false</b> |                                       |
| sdksId            |              | vcenter-mgmt.vcf.sddc.lab/vm-11009    |
| committedStorage  |              | 13164 MB                              |

1. Scroll down to find the `vm` input parameter's `isTemplate` property
2. Confirm that `isTemplate` is false, indicating that the Virtual Machine is no longer a template
3. Click **CLOSE**

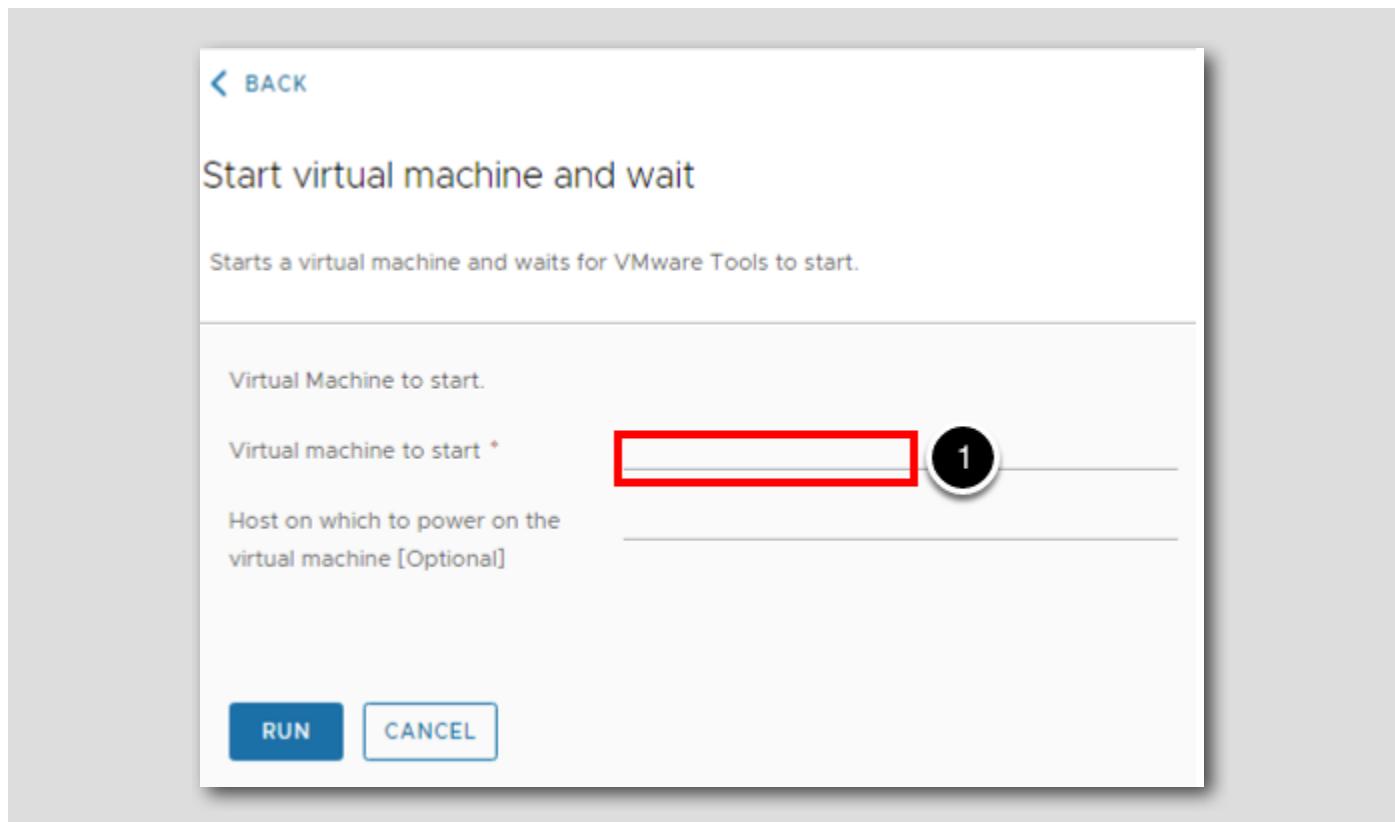
## Initiate the workflow run



Finally, let's power ubuntu-000308 back on.

1. Click the Workflows tab
2. In the filter, remove any existing entries by clicking on the x icon next to it (not shown), then enter **start** Virtual Machine and press return
3. On the Start Virtual Machine and wait workflow card, click RUN

Select the Virtual Machine template field



1. Click the field Virtual machine to start

## Select Virtual Machine

Select VC:VirtualMachine

The screenshot shows the 'Select VC:VirtualMachine' dialog box. On the left, a tree view of the vCenter Server structure is displayed. A red box highlights the path: 'vSphere vCenter Server' > 'https://vcenter-mgmt.vcf.sddc.lab:443/sdk (VirtualCenter-8.0.2.0)' > 'Datacenters' > 'mgmt-datacenter-01' > 'vm'. A circled '1' is placed near the 'vm' node. Another red box highlights the specific virtual machine 'ubuntu-000308', which has a checked checkbox. A circled '2' is placed near the 'SELECT' button at the bottom right. On the right, a detailed table provides information about the selected virtual machine:

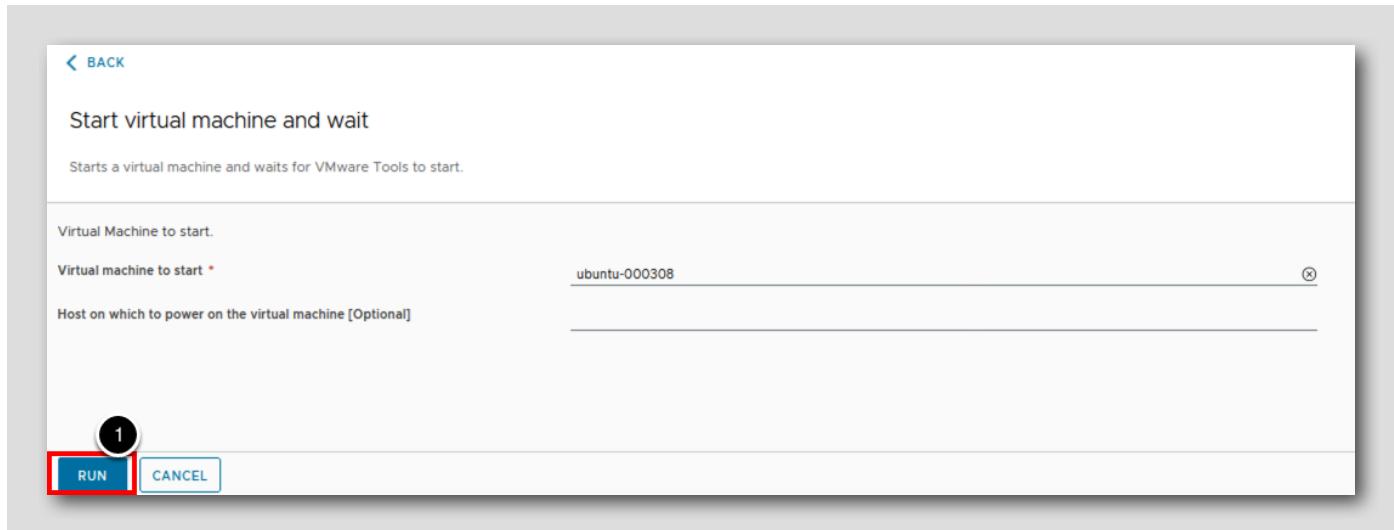
|  |                                      |
|--|--------------------------------------|
| SDK Type                                 | VirtualMachine                       |
| DNS Name                                 | ubuntu-000308                        |
| VMware Tools Status                      | guestToolsRunning                    |
| Consumed Host Memory                     | 0 MB                                 |
| Display name of this virtual machine     | ubuntu-000308                        |
| Non-shared Storage                       | 12708 MB                             |
| configStatus                             | green                                |
| type                                     | VirtualMachine                       |
| 128-bit SMBIOS UUID of a virtual machine | 422b96d7-a74e-1886-d9d-4954d8513b96  |
| Provisioned Storage                      | 52680 MB                             |
| instanceId                               | 502b6de2-811c-3464-22cf-f3b3cef96216 |
| Memory                                   | 1024 MB                              |
| Power State                              | poweredOn                            |
| id                                       | vm-11009                             |
| Orchestrator unique ID                   | vm-11009                             |
| Annotation                               |                                      |
| VM Connection State                      | connected                            |
| Consumed Host CPU                        | 59 MHz                               |
| guestHeartbeatStatus                     | green                                |
| IP address                               | 10.64.12.10                          |

**CANCEL** **SELECT**

1. Navigate to vSphere vCenter Plugin > <https://vcenter-mgmt.vcf.sddc.lab:443/sdk> > Datacenters > mgmt-datacenter-01 > vm  
> Workloads > ubuntu-000308
2. Click SELECT

## Run the workflow

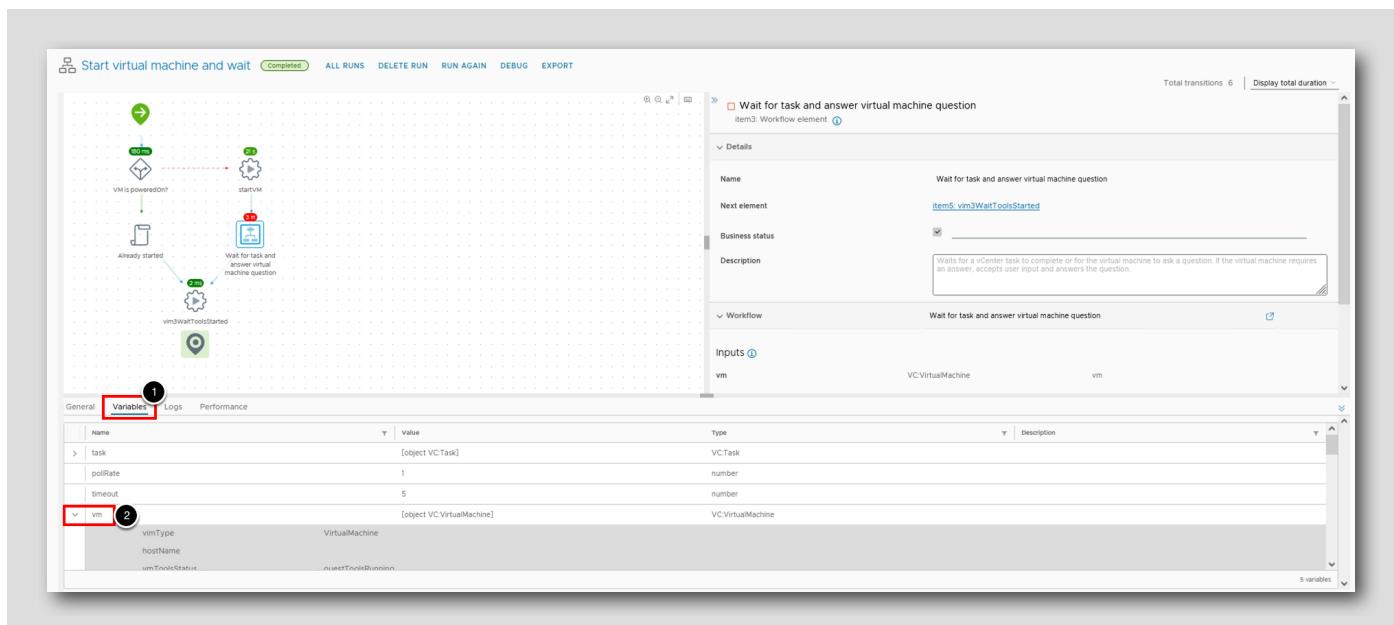
[177]



1. Click RUN

## Inspect the workflow run results

[178]



Once the workflow run has completed:

1. Click the **Variables** tab
2. Expand the **vm** variable to view its properties

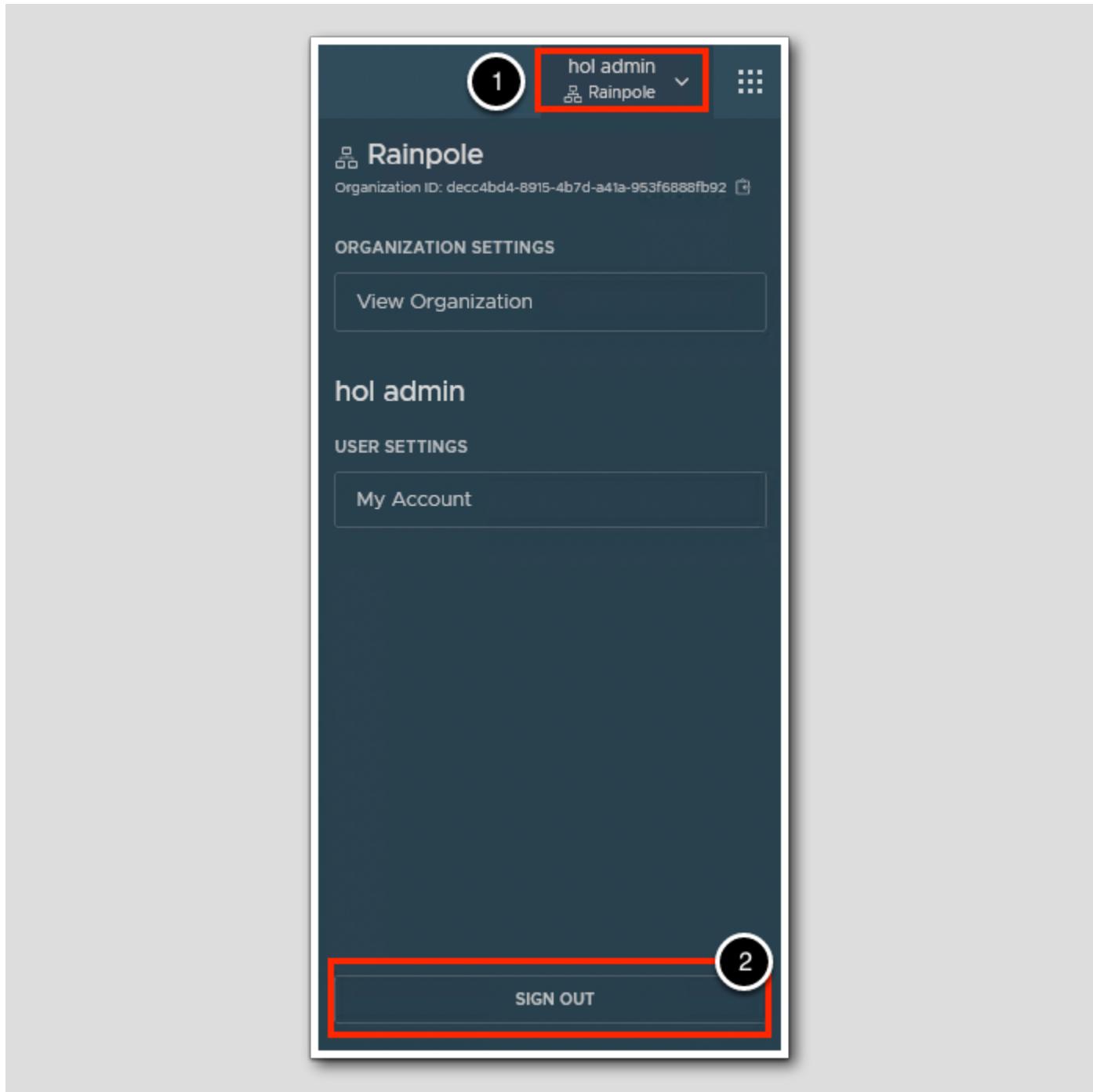
## Inspect the state property

[179]

| Name         | Type             | Description                          |
|--------------|------------------|--------------------------------------|
| biosId       |                  | 422b96d7-a74e-1886-df9d-4954d8513b96 |
| totalStorage |                  | 52680 MB                             |
| instanceId   |                  | 502b6de2-81fc-3464-22cf3b3ce196216   |
| mem          |                  | 1024 MB                              |
| <b>state</b> | <b>poweredOn</b> |                                      |
| <b>id</b>    | <b>vm-11009</b>  |                                      |
| @fullType    |                  | VC.VirtualMachine                    |

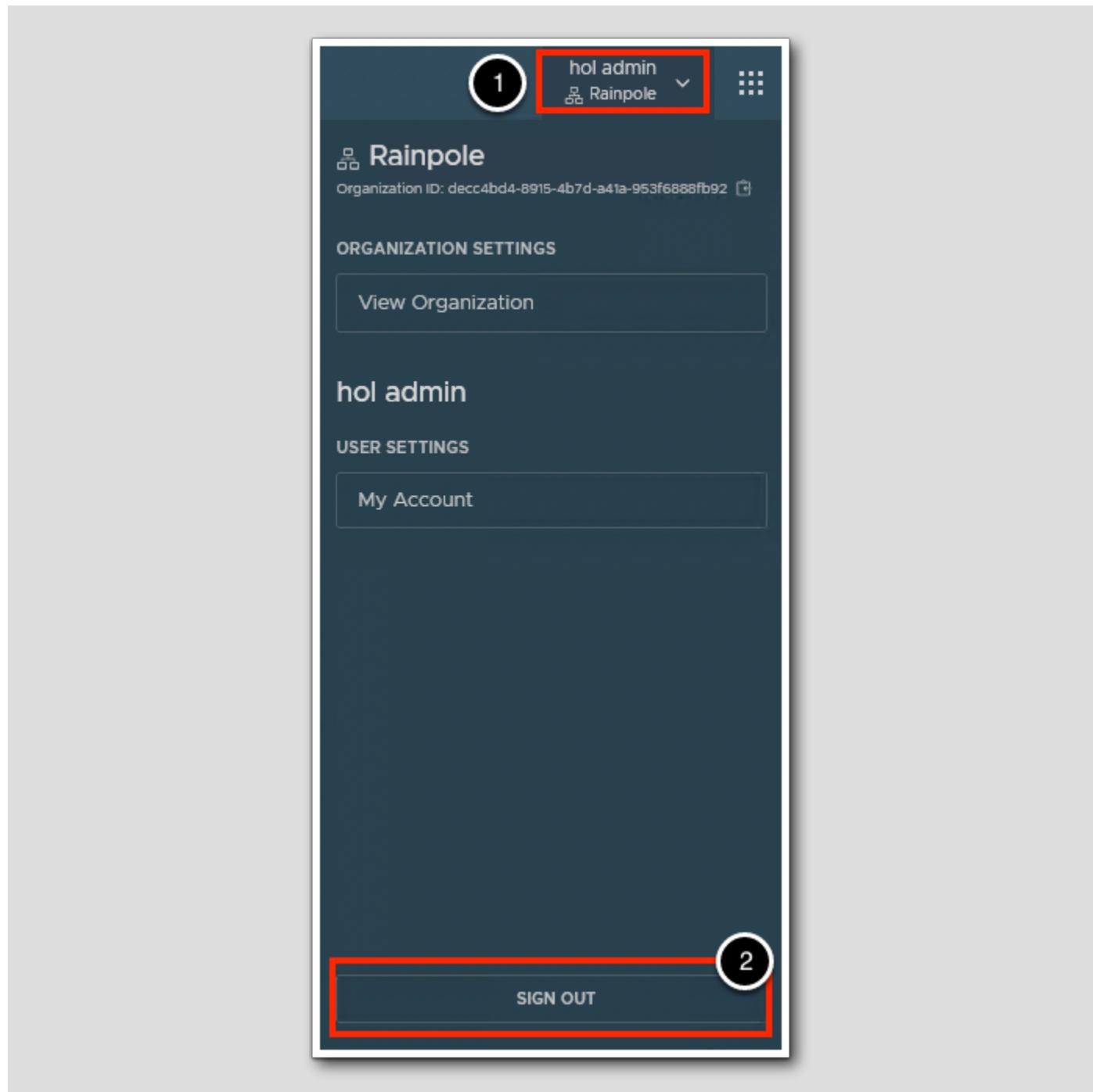
1. Scroll down to find the **vm** input parameter's **state** property
2. Confirm that **state** is **poweredOn**
3. Click **CLOSE**

## Log Out of Aria Automation Orchestrator



To Log out of Aria Automation Orchestrator:

1. Click **hol admin** to open the organization menu.
2. Click **SIGN OUT**.



## Conclusion

[181]

In this module, we walked through the basics of using Aria Automation Orchestrator.

Inputs, outputs, and attributes are the variables available in Aria Automation Orchestrator. They can be passed between workflow

elements within a workflow, or between workflows, and their type allows for consistency across a process.

## You've finished the module

Congratulations on completing the lab module.

If you are looking for additional information on extensibility, have a look here;

<https://www.vmware.com/uk/products/aria-automation.html>

<https://www.vmware.com/uk/products/aria-automation-orchestrator.html>

From here you can:

1. Continue with the next lab module
2. Click [vlp:table-of-contents]Show Table of Contents] to jump to any module or lesson in this lab
3. End your lab and return in the future

## Module 4 - Leverage Existing Scripts in Powershell and Python (45 min) Advanced

### Introduction

[184]

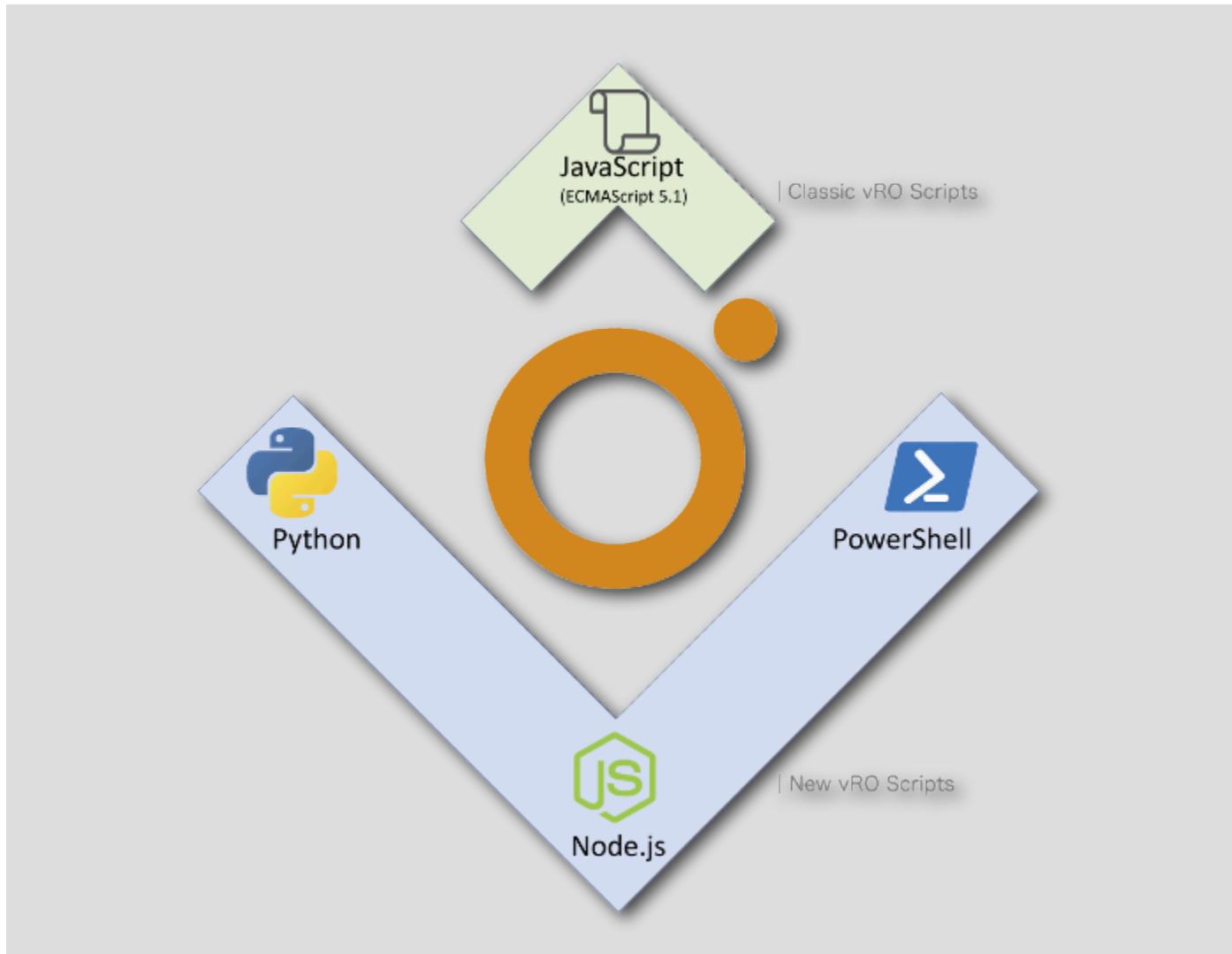
As a Cloud Administrator, it is often required to automate processes that involve disparate systems with different APIs, CLIs, and SDKs. There are numerous scripts publicly available on the Internet that can help with the automation of a process but, often, none of them do exactly what is required for a particular use case. Adding to that, scripting languages used might differ from one task to another. Often the Cloud Administrator has to re-develop one part of the process to fit within the constraints of their particular environment.

With Aria Automation Orchestrator, it is possible to leverage scripts with different languages and stitch them together within a single workflow, thereby easing the integration of all of the different tools, utilities, and processes that a Cloud Administrator needs to use to solve business problems.

#### Lab Captain(s):

- Katherine Skilling, Senior Architect, United Kingdom

## Core Concepts



Aria Automation Orchestrator supports the following runtimes:

- Python 3.10
- Node.js 18\*
- Node.js 20
- PowerCLI 13 / PowerShell 7.4
- PowerCLI 12 / PowerShell 7.1\*

Note: The PowerCLI 12 and Node.js 18 runtimes are deprecated and will be removed in a future release

Aria Automation Orchestrator supports injecting dependencies such that many scripts found on the Internet can be integrated into an Aria Automation Orchestrator workflow.

In this lesson, let's see how to import existing scripts into Aria Automation Orchestrator.

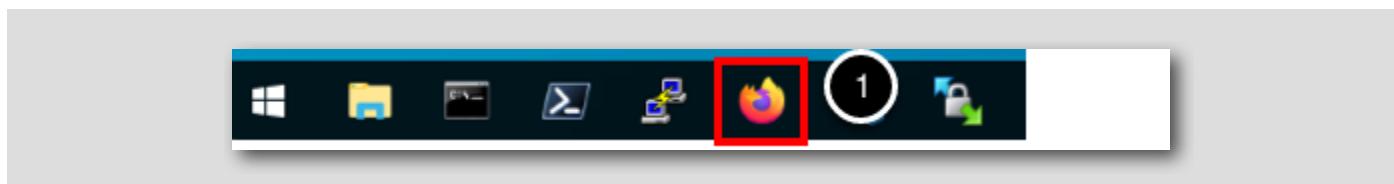
## Log in to Aria Automation Orchestrator

[186]

In the following few pages, we will log in to Aria Automation Orchestrator.

### Open the Firefox Browser from Windows Quick Launch Task Bar

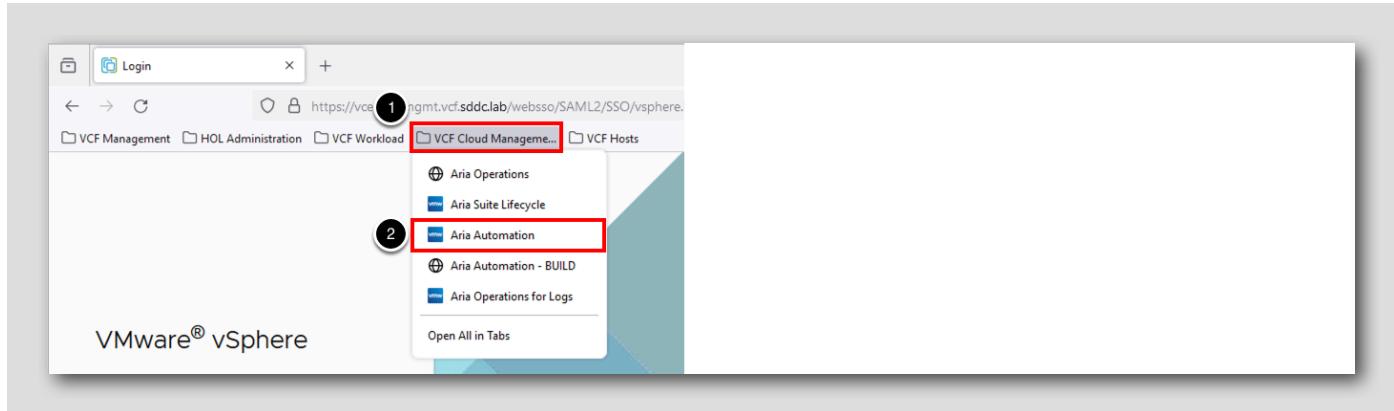
[187]



If the browser is not already open, launch Firefox.

1. Click the Firefox icon on the Windows Quick Launch Task Bar.

## Log in to Aria Automation

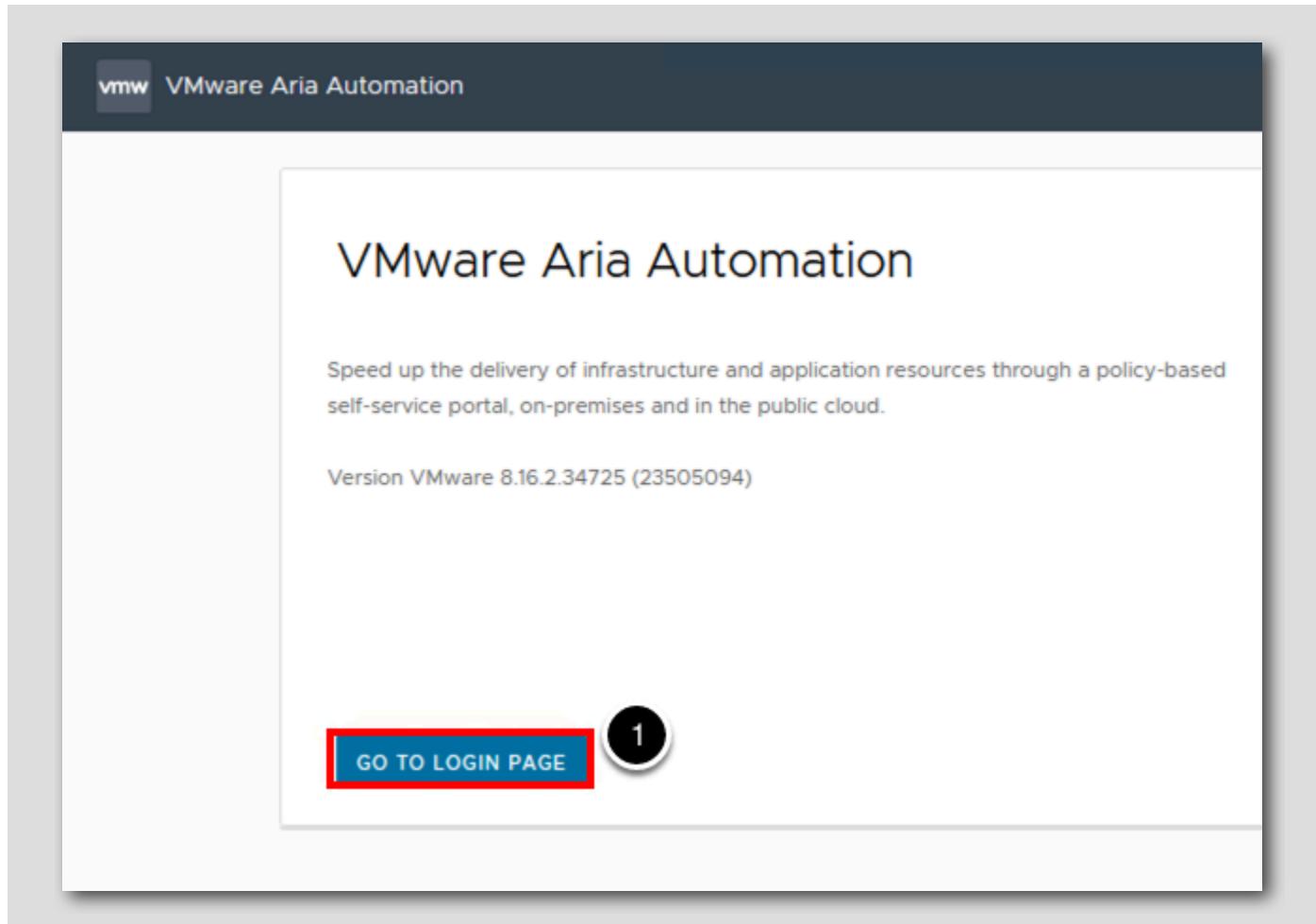


Once Firefox has loaded:

1. Click the VCF Cloud Management bookmark folder
2. Click Aria Automation.

## Redirect to Workspace ONE Access for Sign-On

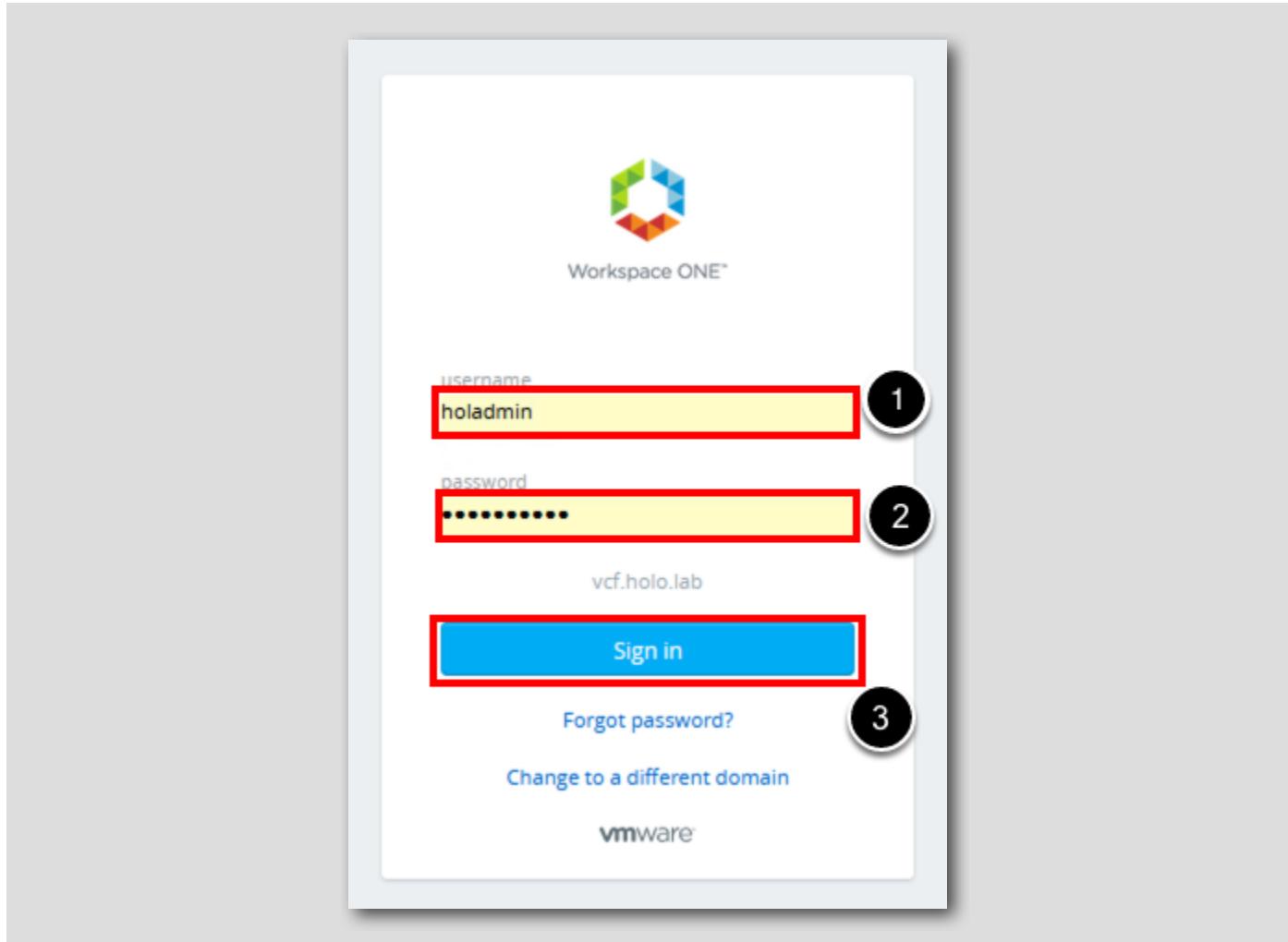
[189]



Aria Automation is integrated with Workspace ONE Access (aka VMware Identity Manager) and we need to redirect to the Workspace ONE Access login page to complete our log in progress.

1. At the VMware Aria Automation page, click GO TO LOGIN PAGE.

## Workspace ONE Access Login

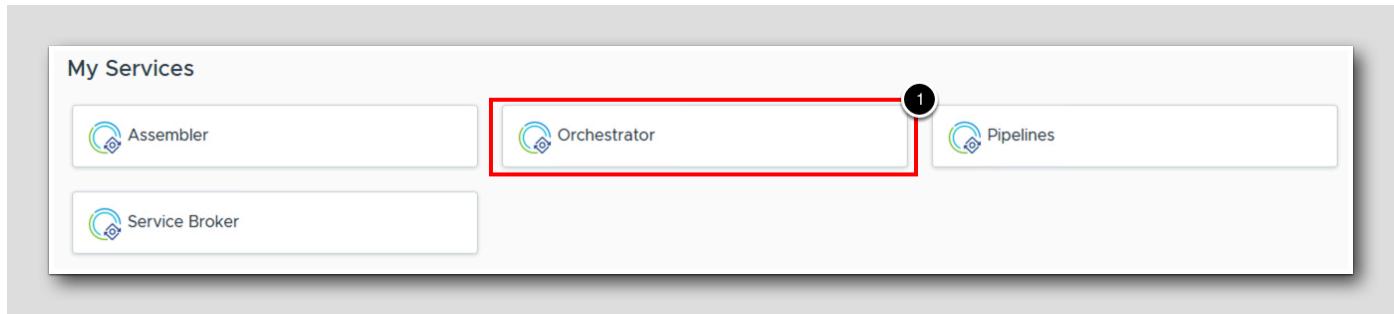


Note: The credentials for **holadmin** should already be cached in the browser window.

At the **Workspace ONE Access** prompt, type in the following user and password information.

1. At the **username** field, type **holadmin**.
2. At the **password** field, type **VMware123!**.
3. Click **Sign in**.

## Launch the Orchestrator Service



From within theCloud Services Console, under My Services:

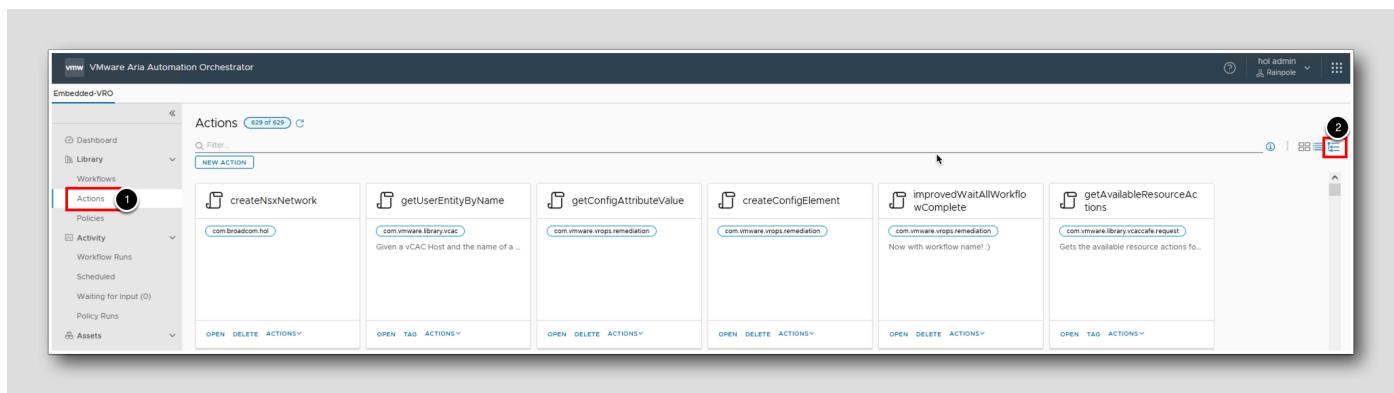
1. Click the Orchestrator service.

## Create a simple PowerCLI script

Aria Automation Orchestrator already has a plugin with numerous out-of-the-box workflows to interact with vSphere environments. However, there are also numerous existing scripts and pieces of code available in PowerShell (leveraging PowerCLI) to perform automation tasks on vSphere environments.

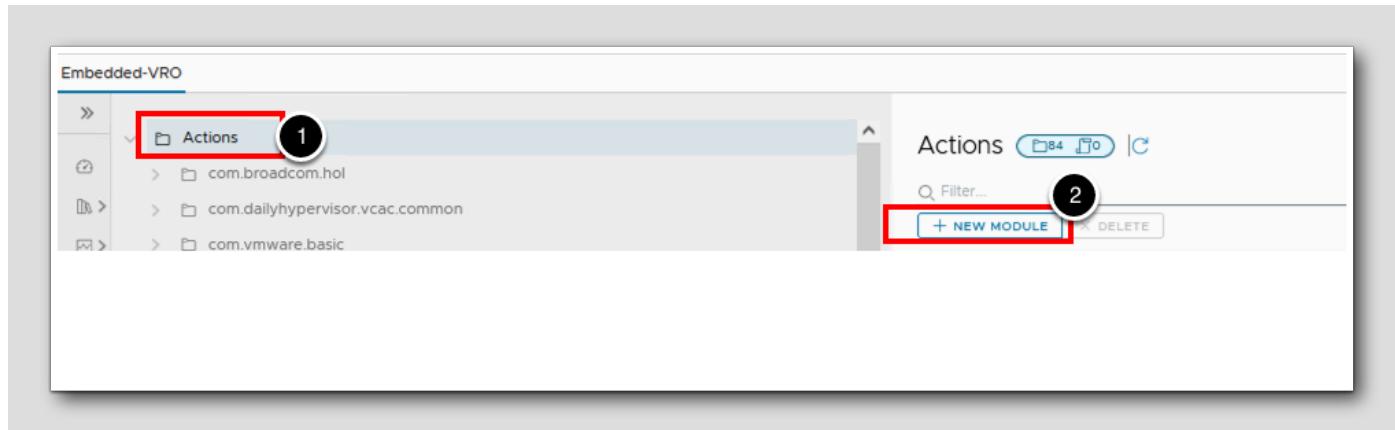
Instead of re-developing scripts in Aria Automation Orchestrator, let's see how it is possible to import PowerCLI scripts and use them directly in Aria Automation Orchestrator.

## Select the Actions tab



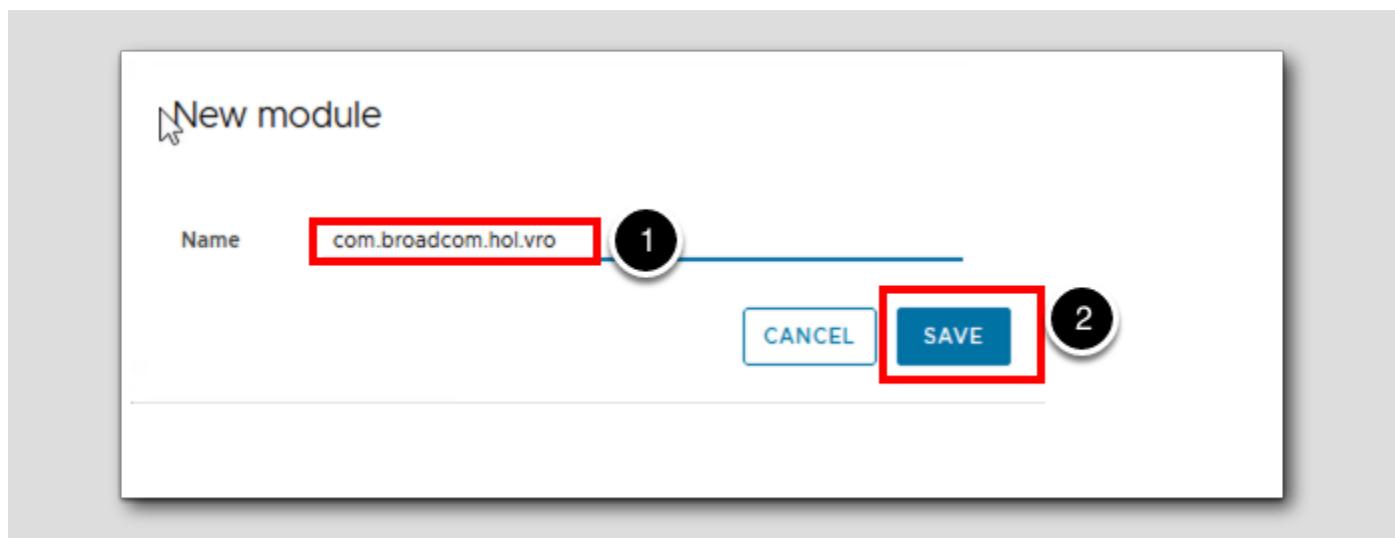
1. Select the Actions tab on from the left menu
2. Select tree view

## Create an Action Module



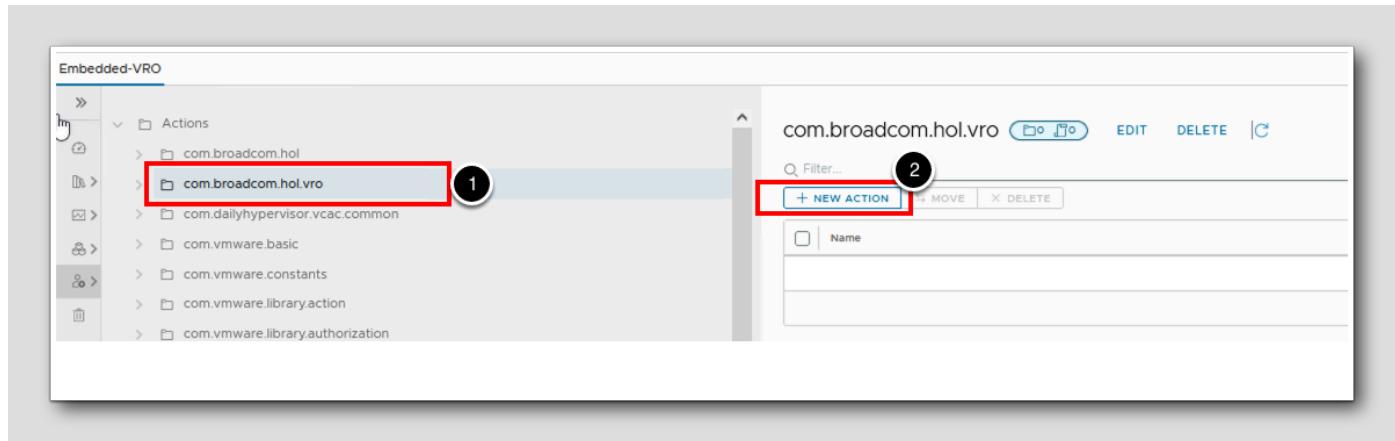
1. Make sure the Actions folder is selected
2. Click + NEW MODULE button

## Name the module



1. Enter the module name: com.broadcom.hol.vro
2. Click the SAVE button

## Create a new action



1. Select the newly created module com.broadcom.hol.vro

2. Click the button + NEW ACTION

Name the action

[197]

Embedded-VRO

getVMWithPS

General Script Version History Audit

Name  1

Module com.broadcom.hol.vro

ID

Description

Version 0.0.0

Tags Enter a new tag

Groups

2

CREATE VERSION CLOSE

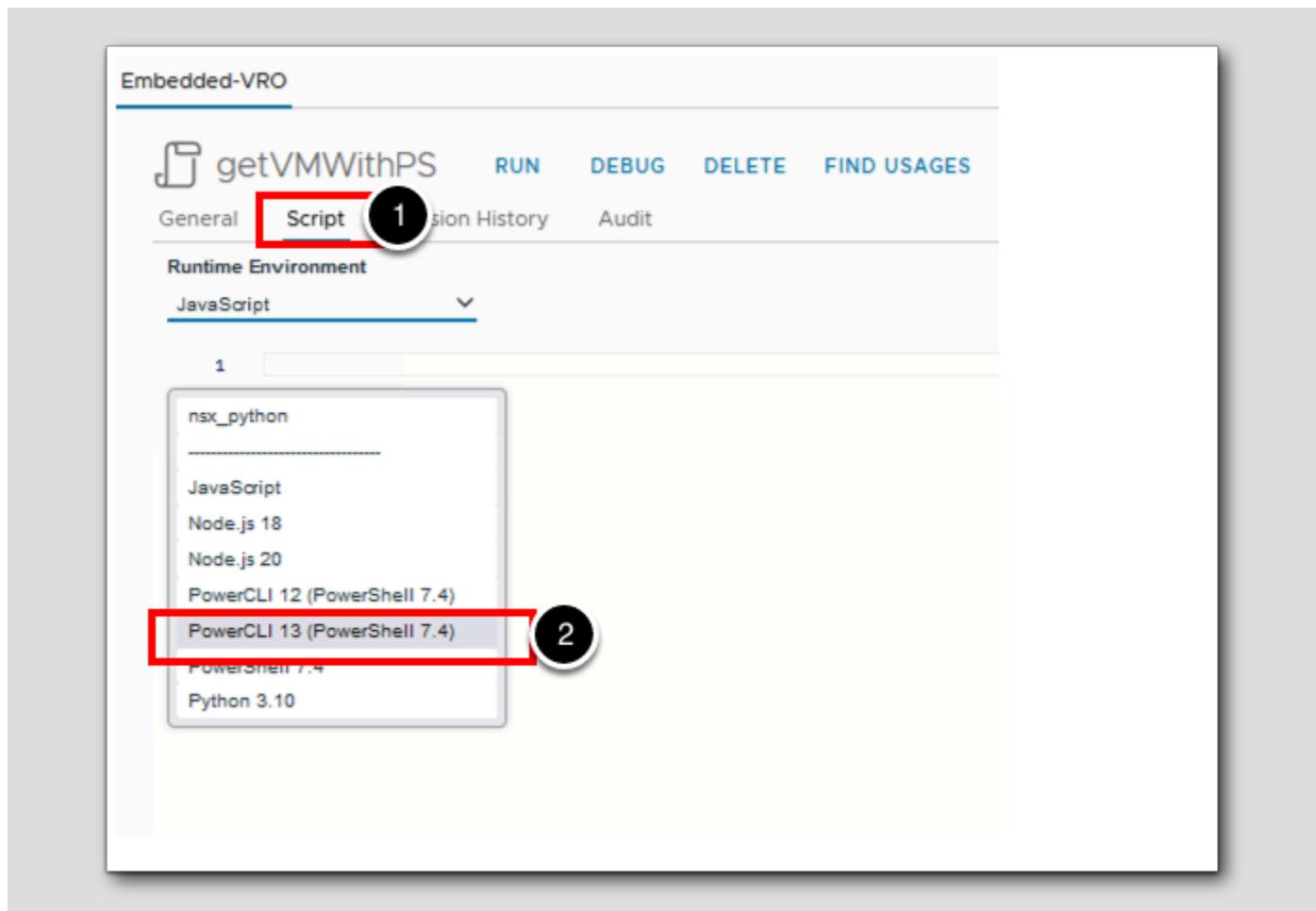
vmware

HANDS-ON LABS MANUAL | 165

1. Enter the action name: getVMWithPS
2. Click the CREATE button

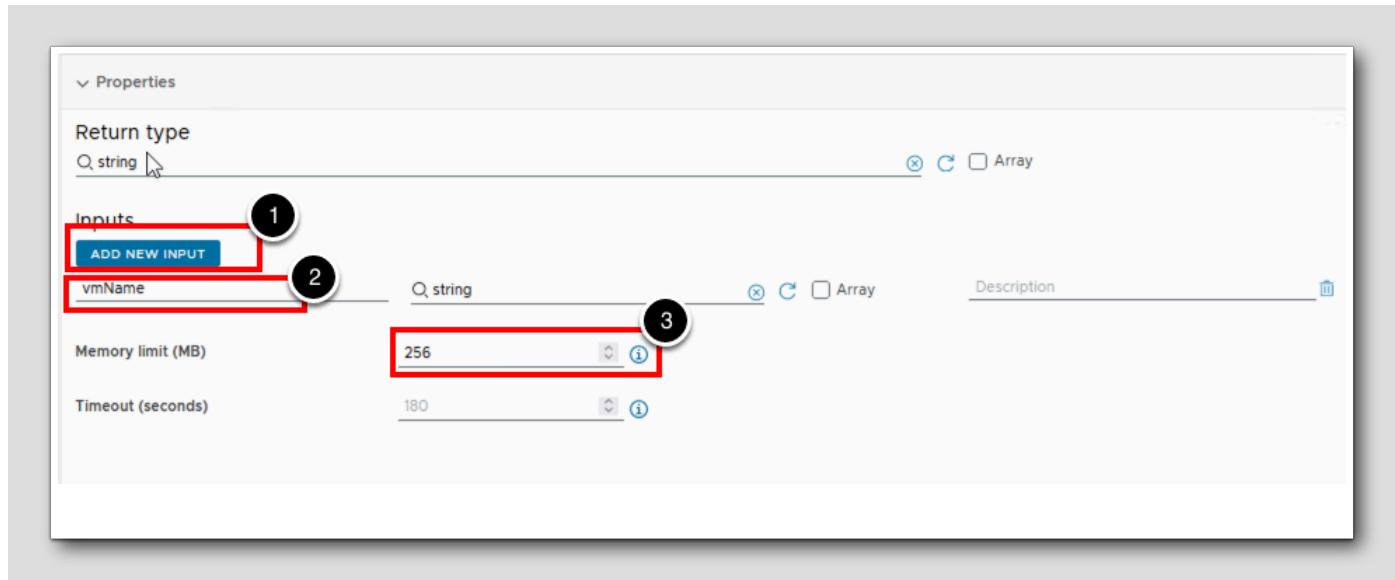
Change runtime

[198]



1. Select the Script tab
2. Select PowerCLI 13 (PowerShell 7.4) from the Runtime dropdown list

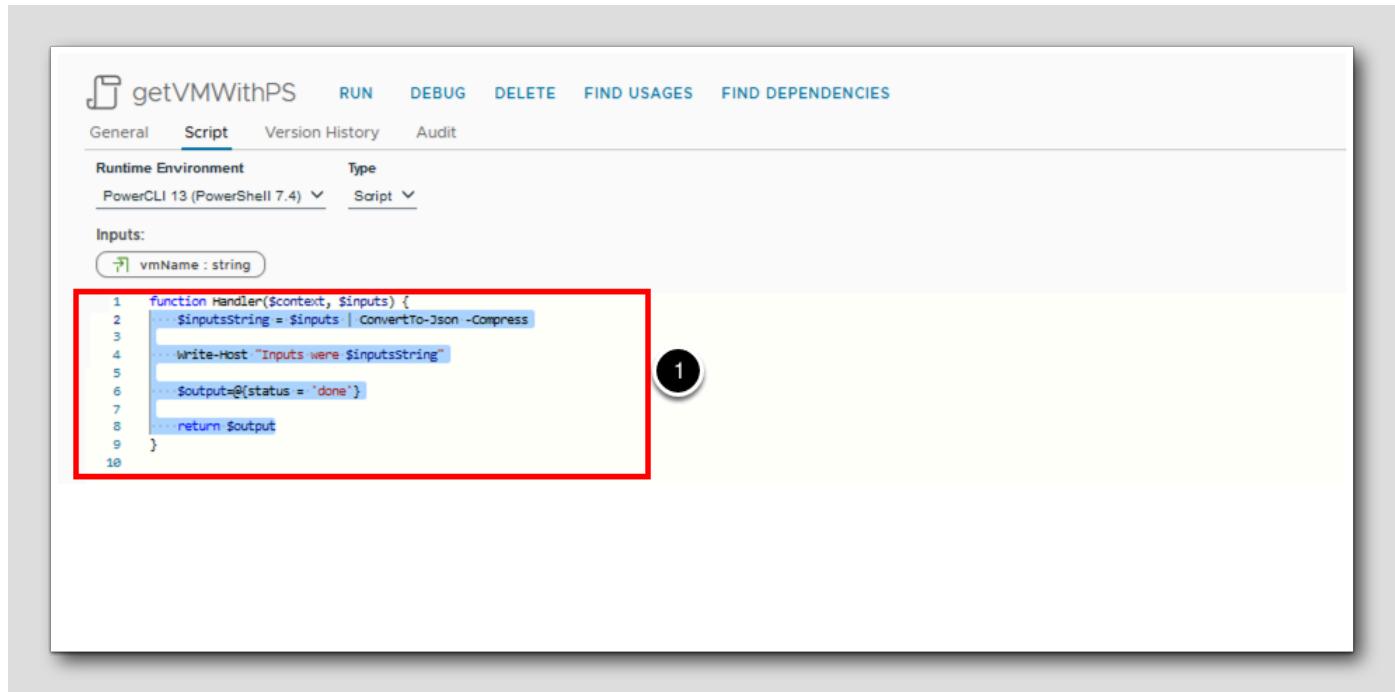
## Configure Action



In the Properties panel (on the right)

1. Click the **ADD NEW INPUT** button to create a new input
2. Name the new input **vmName**
3. Change the Memory limit (MB) to **256** to allow the action to consume additional resources at runtime

## Enter Powershell code



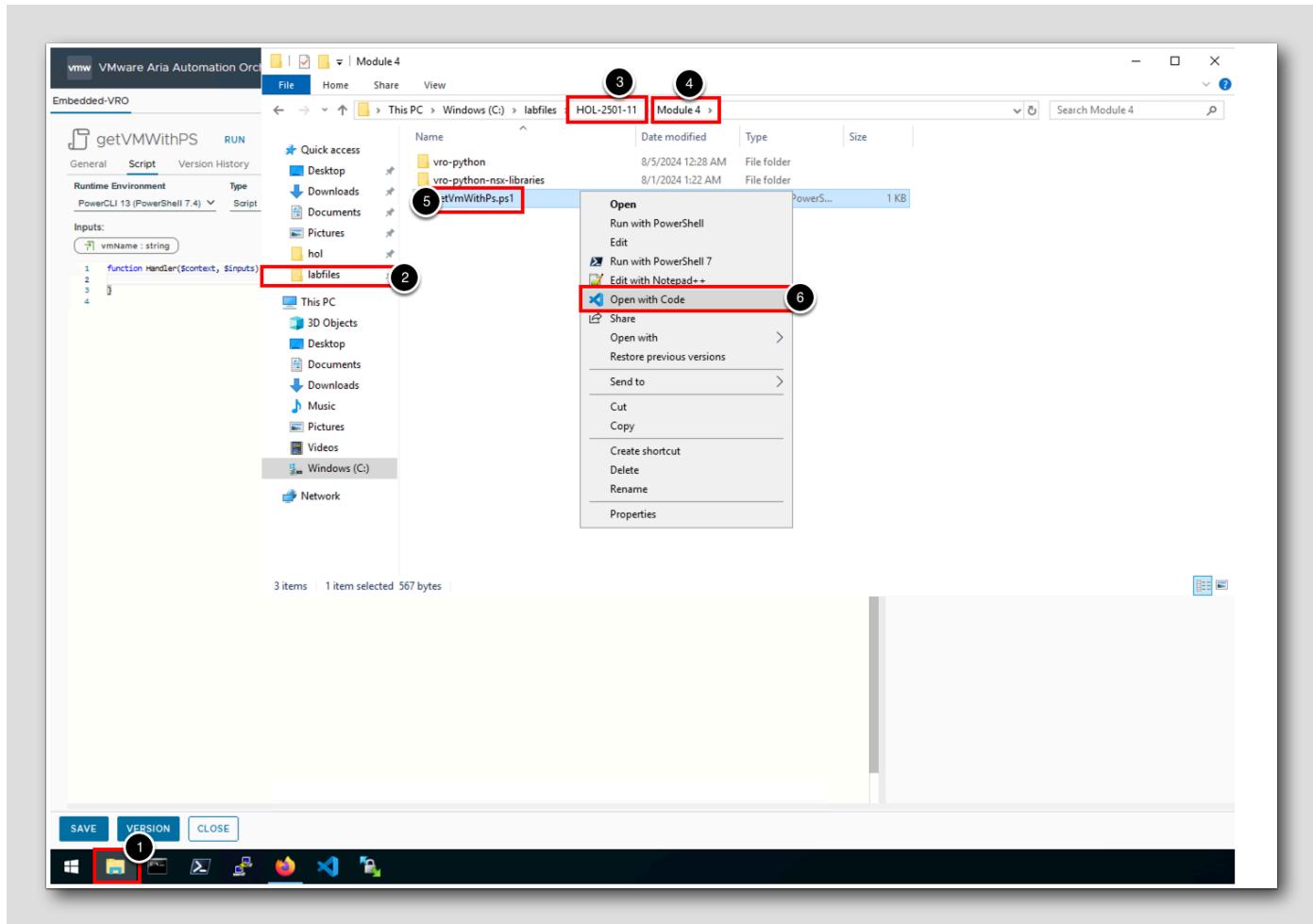
The screenshot shows a PowerShell script editor window titled "getVMWithPS". The "Script" tab is selected. The runtime environment is set to "PowerCLI 13 (PowerShell 7.4)". The script contains the following code:

```
1 function Handler($context, $inputs) {
2     $inputsString = $inputs | ConvertTo-Json -Compress
3     Write-Host "Inputs were $inputsString"
4     $output=@{status = 'done'}
5     return $output
6 }
7
8 }
```

A red box highlights the entire body of the `Handler` function, from line 1 to line 8. A black circle with the number "1" is positioned to the right of the highlighted area.

1. Select all the code inside the function `Handler` (the code between the curly braces) and remove it.

## Open Powershell script



1. Open the Windows Explorer from the taskbar
2. Select Labfiles from the quick access section
3. Navigate to the folder HOL-2501-11
4. Navigate to the folder Module 4
5. Right-Click on the file getVmWithPs.ps1
6. Select Open with Code

## Copy PowerShell code

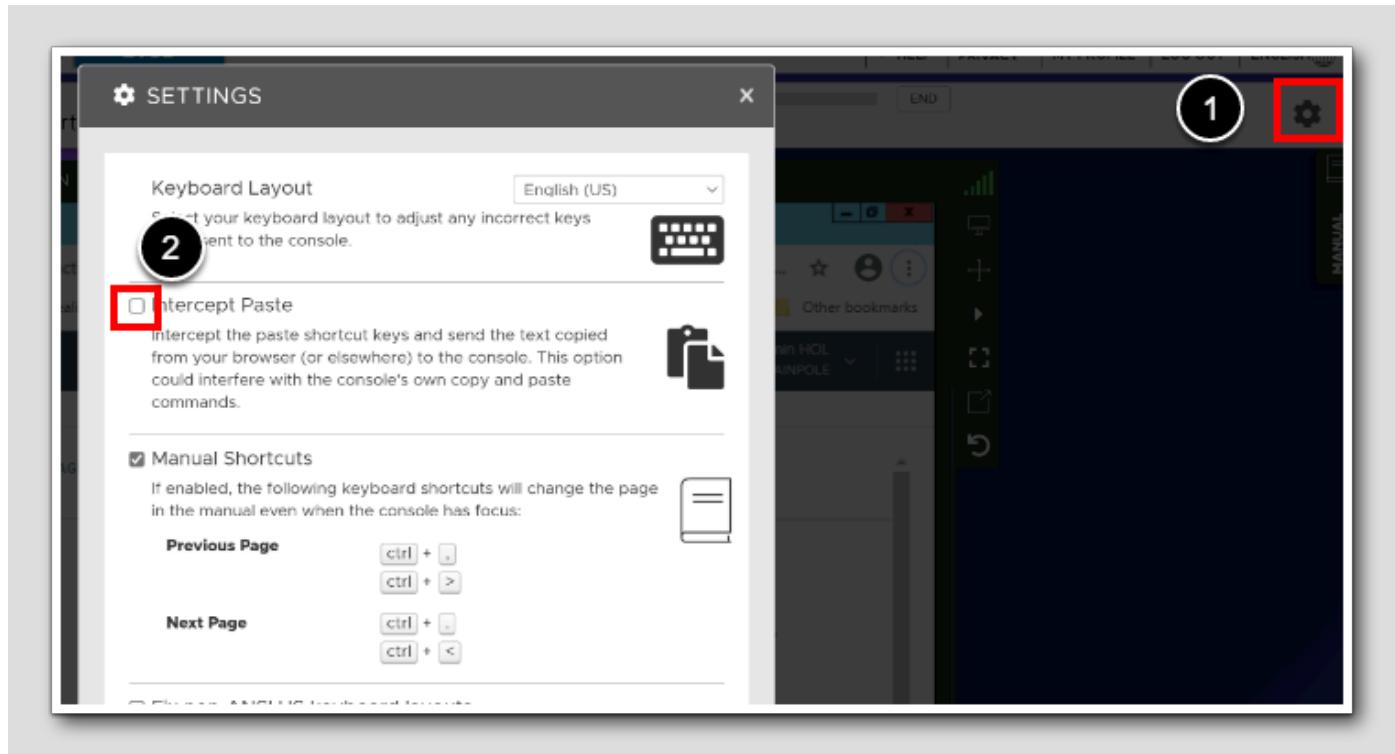


```
> getVmWithPs.ps1 <
C:\> labfiles > HOL-2501-11 > Module 4 > > getVmWithPs.ps1
1 # Connect to vCenter
2 Set-PowerCLIConfiguration -InvalidCertificateAction Ignore -Confirm:$false | Out-Null 2> $null
3 Connect-VIServer "vcenter-mgmt.vcf.sddc.lab" -username "administrator@vsphere.local" -password "VMware123!" -ErrorAction Stop | Out-Null 2> $null
4
5 # Search for the VM by name
6 $vmPartialName = $inputs.vmName
7 Write-Host "Searching VM with name matching $vmPartialName"
8 $matchingVms = Get-VM | Where-Object { $_.Name -like "*$vmPartialName*" }
9 Write-Host "$($matchingVms.Count) VM(s) found with name matching $vmPartialName"
10
11 return $matchingVms[0].Name
```

1. Select all the code from the script (either use the mouse or type ctrl-A)
2. Copy the text to the clipboard (either right-click, Copy or ctrl-C)

Close Visual Studio Code window

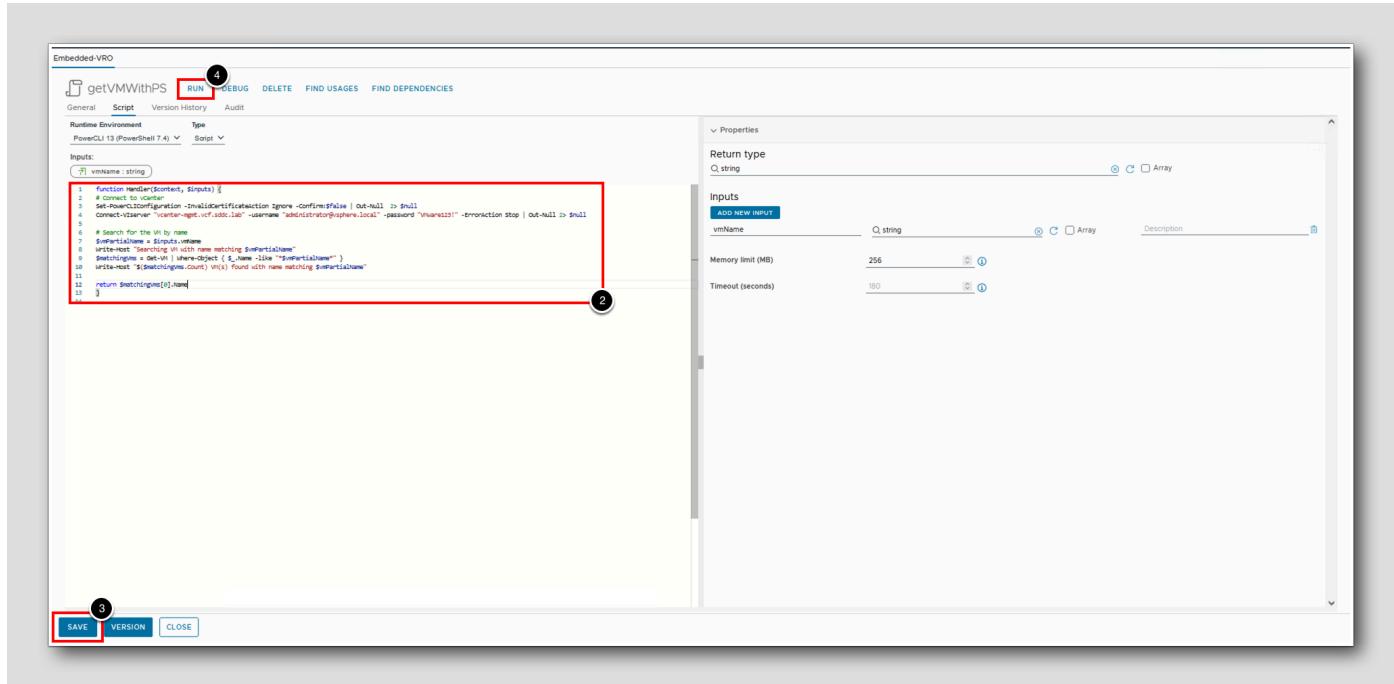
## Verify Paste Setting in the Learning Platform Interface



Since we will need to use a keyboard shortcut to paste the code into the lab console, we must first make sure that the lab interface paste settings are correct.

1. Click the gear icon in the top-right corner of the lab interface
2. Make sure that the Intercept Paste check box is NOT checked
3. [not shown] Click Close to close the settings window

## Paste PowerShell code

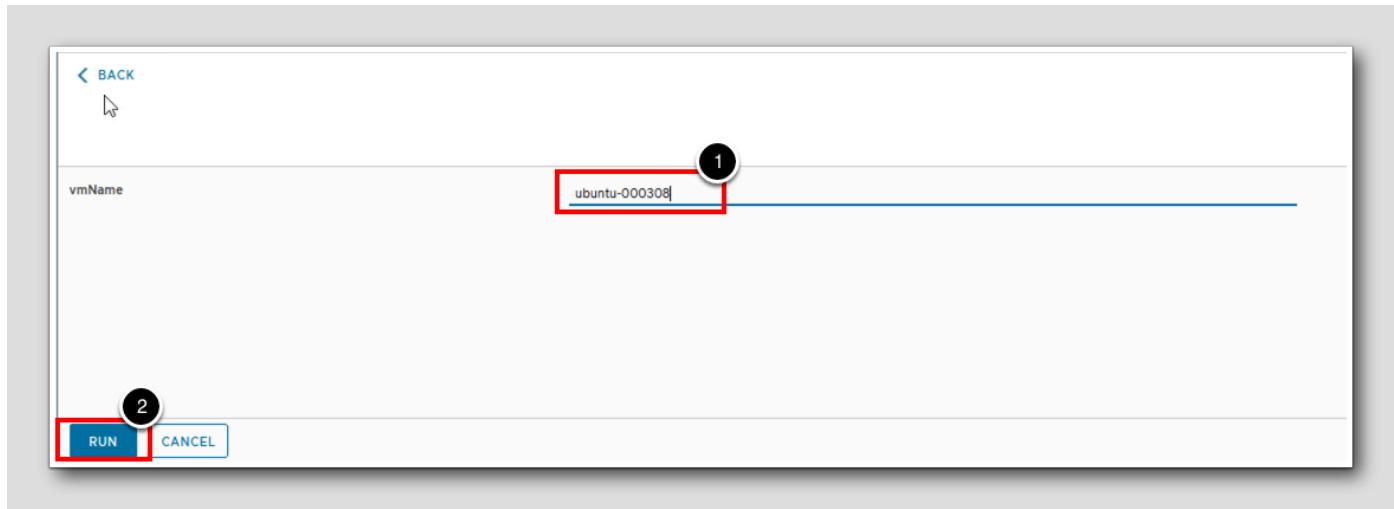


1. [Not shown] Switch back to Orchestrator Client by selecting **Chrome** from the taskbar
2. Return to the Aria Automation browser tab and **Paste** the code within the Handler function (between { and }) with the keyboard shortcut:  
**CTRL + V**
3. Click **SAVE**
4. Click **RUN** to run the action

Note: if you see a pop-up window asking whether you want to enable paste intercept, be sure to cancel that option.

## Run the action

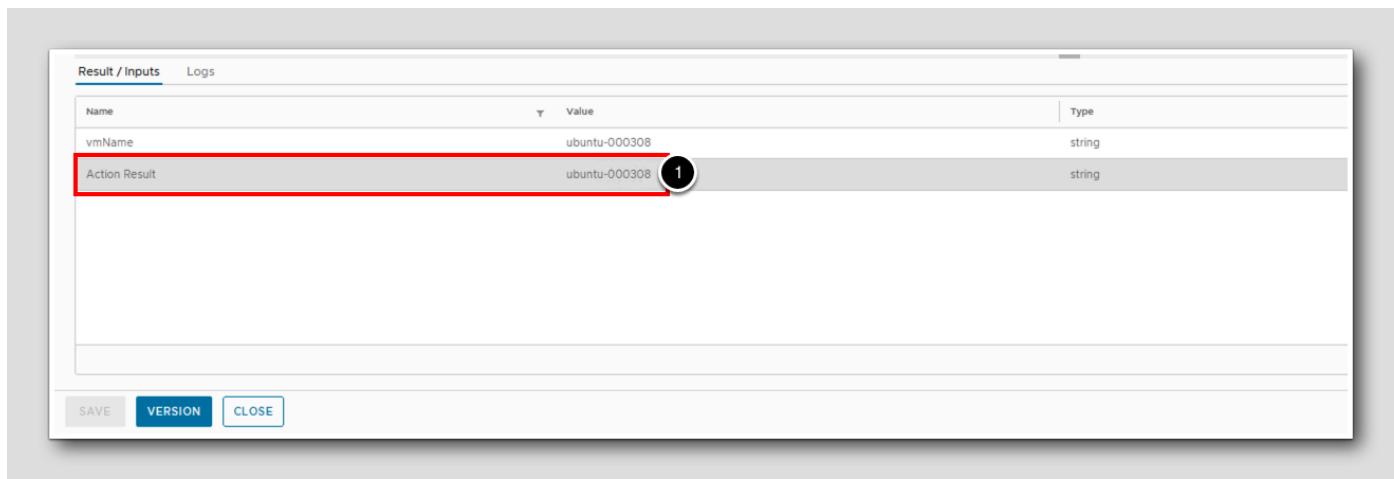
[205]



1. Enter the name **ubuntu-000308**, this is the name of an existing virtual machine within our vCenter inventory
2. Click the **RUN** button

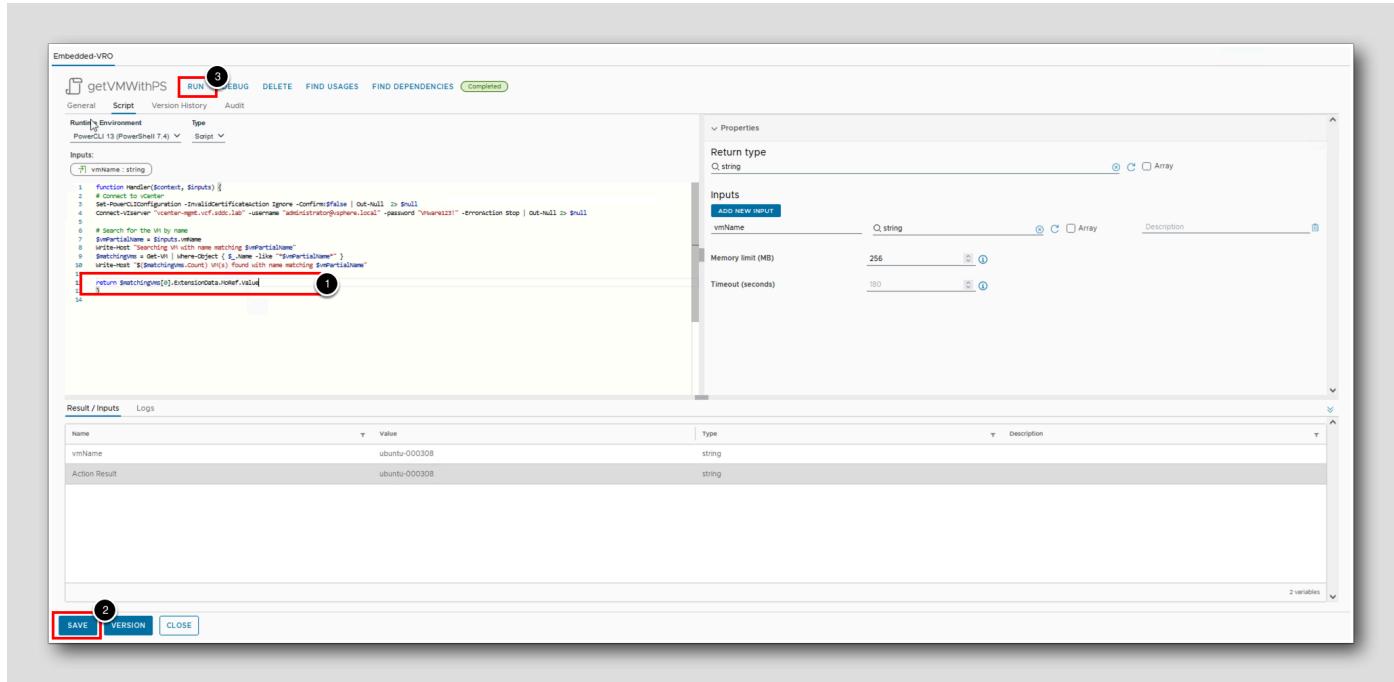
## Analyze the result

[206]



1. After waiting for a few minutes while the Action runs, the **Action Result** should return the name of a VM that matches the input.

## Return the VM ID



1. In the last line of the function, replace the property Name with

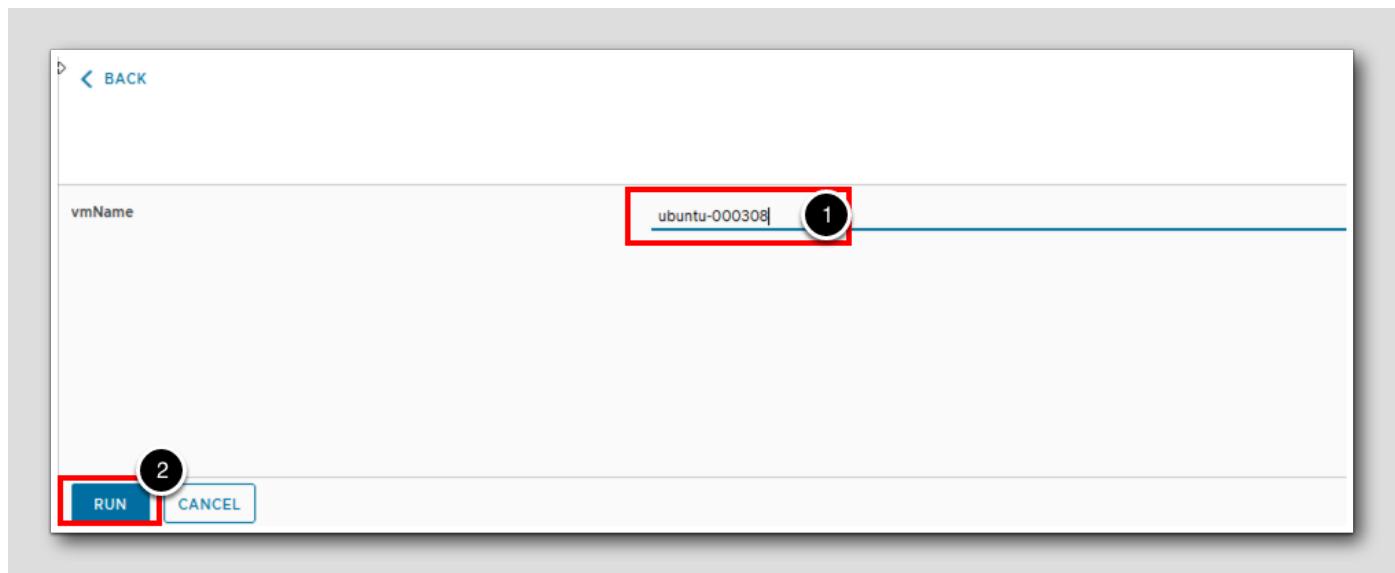
ExtensionData.MoRef.Value

It will now return the vCenter ID instead.

2. Click the **SAVE** button to save the change
3. Click the **RUN** button to run the updated action

## Run the action

[208]



1. Enter the name ubuntu-000308

2. Click the RUN button

## Analyze the result

[209]

| Result / Inputs |        | Logs          |
|-----------------|--------|---------------|
| Name            | Type   | Value         |
| vmName          | string | ubuntu-000308 |
| Action Result   | string | vm-11009 (1)  |
|                 |        |               |
|                 |        |               |
|                 |        |               |

1. After waiting for a few minutes while the Action runs, the **Action Result** should return the ID of the first VM where the name matches "ubuntu-000308".

Leveraging PowerShell scripts in Aria Automation Orchestrator is easy and can drastically improve the re-usability of existing development efforts. On top of that, a PowerShell script can interact with Aria Automation Orchestrator Action inputs and outputs so it can be part of a whole end-to-end automated process.

Aria Automation Orchestrator gives the power to create automation leveraging existing scripts!

## Leverage Python to integrate with NSX

[210]

NSX (formally NSX-T) doesn't have a plugin for Aria Automation Orchestrator but it does have a Python SDK as well as [sample scripts](#). We can run an existing Python script as a Aria Automation Orchestrator Action, that was not specifically designed to run with Orchestrator by using one of the following approaches:

1. Packaging and importing the script and its dependencies into our Aria Automation Orchestrator Action as a zip file
2. Creating a private repository to hold the dependencies, and configuring the repository within Aria Automation Orchestrator as the runtime environment for our Action

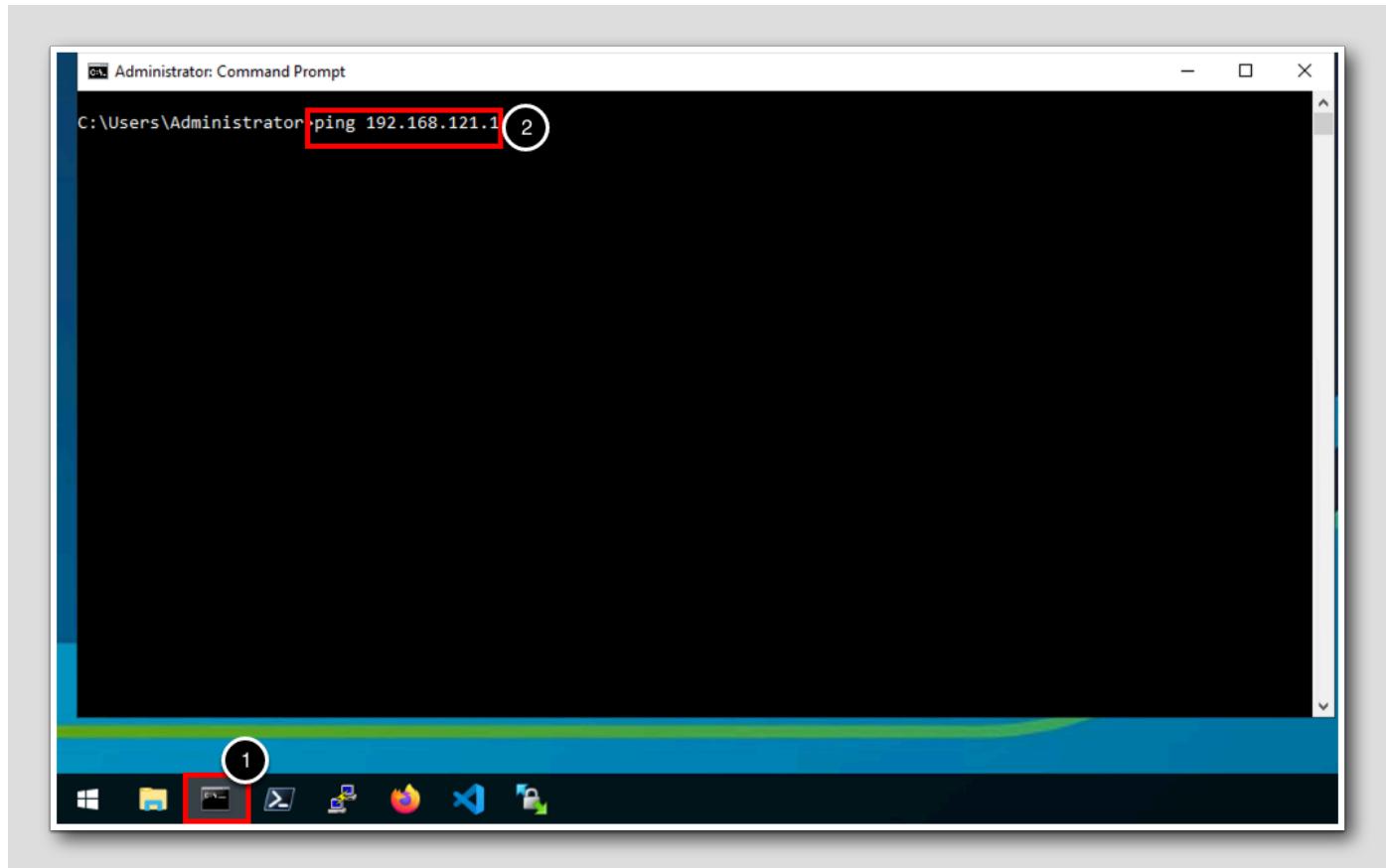
In this lesson we will explore the second option, which was introduced in Aria Automation Orchestrator 8.11.2. This option allows us to specify the exact version of dependency modules Aria Automation Orchestrator should use when running our Action, and where it should install these dependencies from. For our use case of integrating with the NSX Python SDK, this is essential due to a mismatch between some of the Python modules provided by the Polyglot functionality in Aria Automation Orchestrator and the dependencies of the Python SDK.

To achieve the desired outcome of using a Python script to create a new segment within NSX using Aria Automation Orchestrator we will perform four steps:

1. The first step is simply to check that the network address ranges we want to create are not already present. We will use a simple ping command to confirm this.
2. We will then run the script we want to use with Aria Automation Orchestrator in the main console machine to make sure it works and does what's expected, Visual Studio Code will be used to facilitate this.
3. Next, we will amend the script from the previous step to develop a script that is consumable by Aria Automation Orchestrator.
4. Finally we will configure the script and dependencies in Aria Automation Orchestrator.

Note in our environment we have preconfigured the Environment and Repository we will use with our Aria Automation Orchestrator Action to save time.

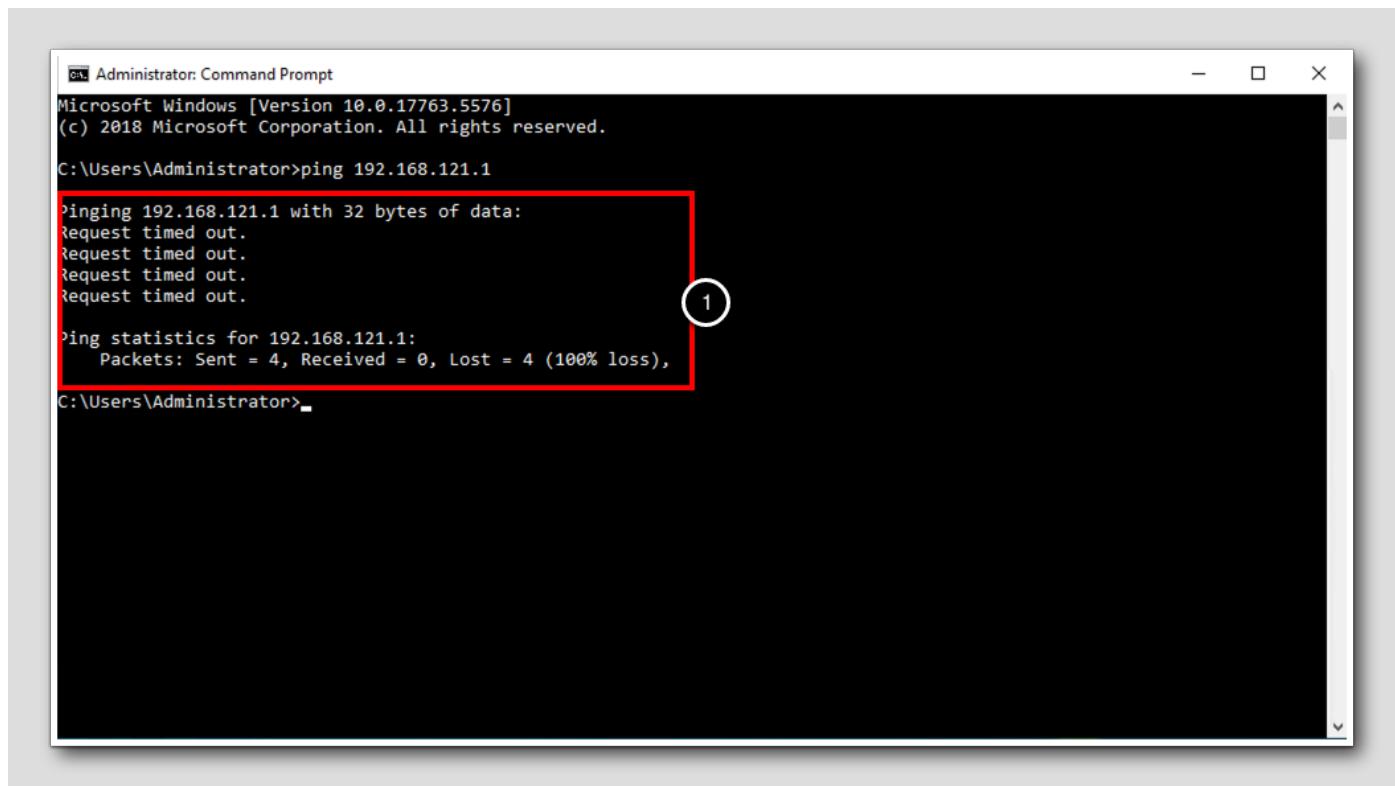
## Confirm the First Network Segment is Not Accessible



Before we start developing and testing our script we are going to quickly check that the gateway addresses we will use with our new Segments are not already in use within the environment. We could log in to the NSX Manager GUI and check for Segments, however this would not confirm that the addresses are not used for traditional networks within the environment. As this is a lab environment we are simply going to use a ping command to perform our check to save time.

1. Open a Windows Command Prompt
2. Enter the command `ping 192.168.121.1` and press Enter

## Review the Results of the Command



The screenshot shows an 'Administrator: Command Prompt' window on Windows 10. The command 'ping 192.168.121.1' was run, resulting in four timed-out requests and a 100% loss report. A red box highlights the output, and a white circle with the number '1' points to the '100% loss' line.

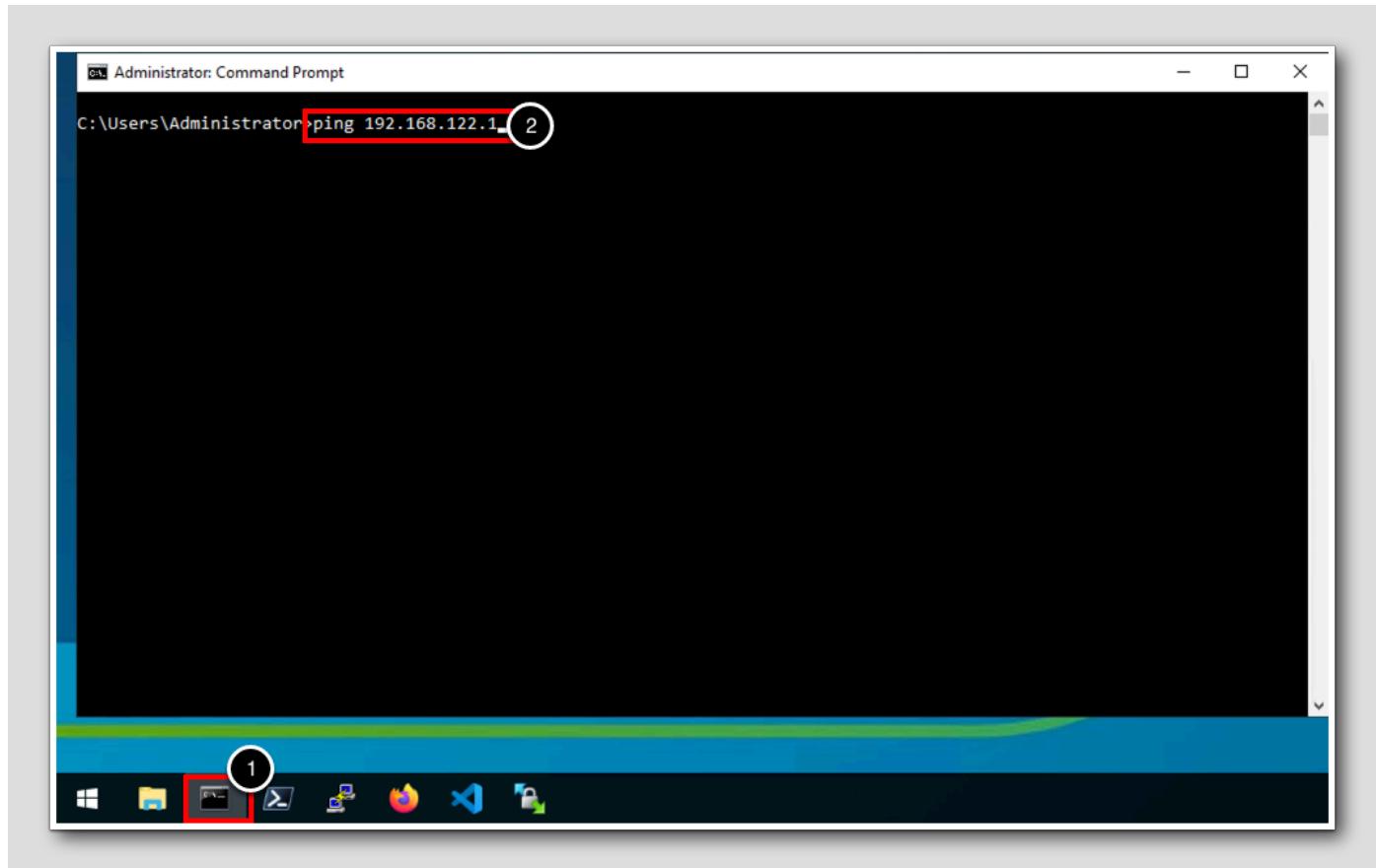
```
Administrator: Command Prompt
Microsoft Windows [Version 10.0.17763.5576]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\Administrator>ping 192.168.121.1
Pinging 192.168.121.1 with 32 bytes of data:
Request timed out.
Request timed out.
Request timed out.
Request timed out.

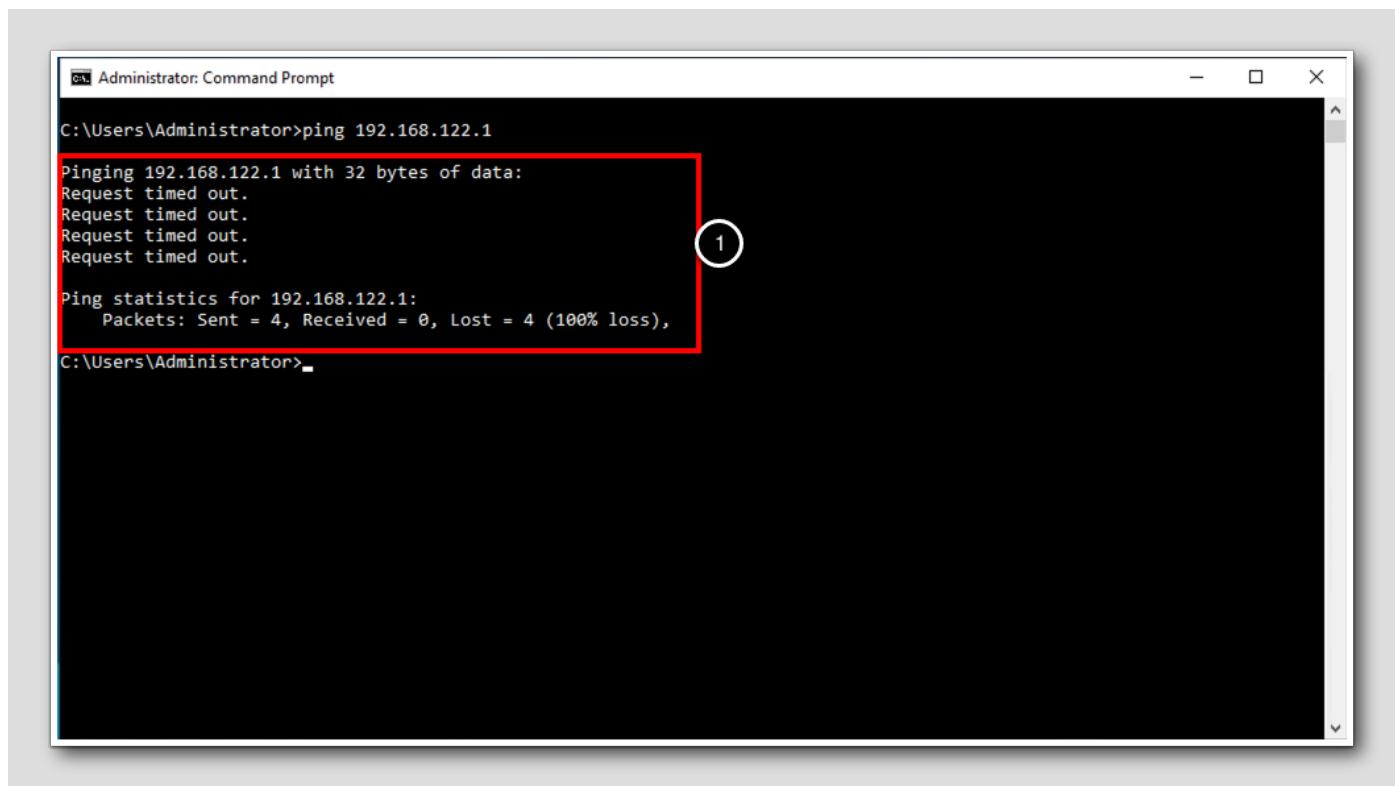
Ping statistics for 192.168.121.1:
    Packets: Sent = 4, Received = 0, Lost = 4 (100% loss),
C:\Users\Administrator>
```

1. The result of the command should show that 4 packets were sent and 4 were lost

Confirm the Second Network Segment is Not Accessible



1. Open a Windows Command Prompt
2. Enter the command ping 192.168.122.1



The screenshot shows an 'Administrator: Command Prompt' window. The command entered is 'ping 192.168.122.1'. The output shows four requests timed out, followed by ping statistics indicating 4 packets sent, 0 received, and 4 lost (100% loss). A red box highlights the statistics line, and a white circle with the number '1' is drawn around the 'lost' value in the statistics.

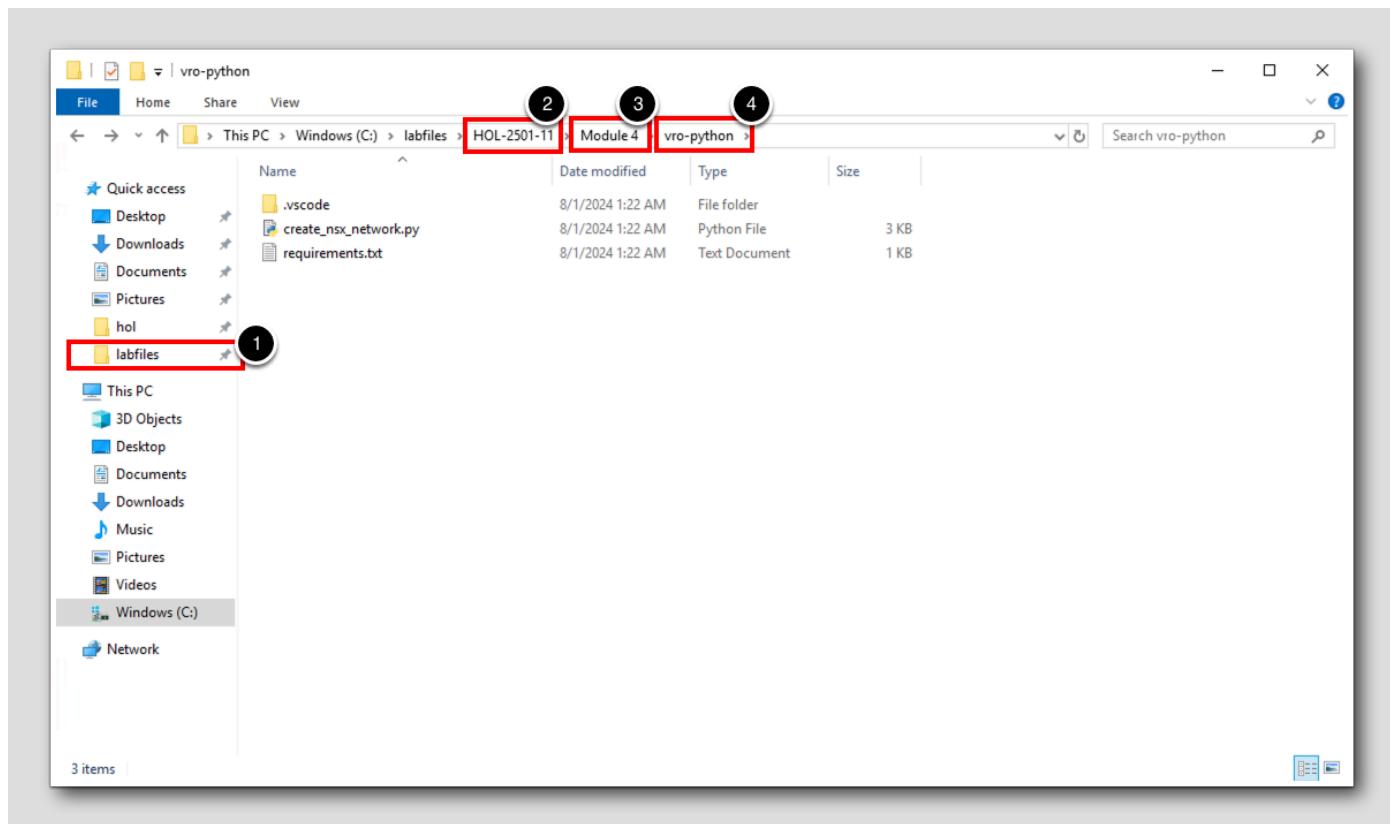
```
C:\Users\Administrator>ping 192.168.122.1
Pinging 192.168.122.1 with 32 bytes of data:
Request timed out.
Request timed out.
Request timed out.
Request timed out.

Ping statistics for 192.168.122.1:
    Packets: Sent = 4, Received = 0, Lost = 4 (100% loss),
C:\Users\Administrator>
```

1. The result of the command should show that 4 packets were sent and 4 were lost

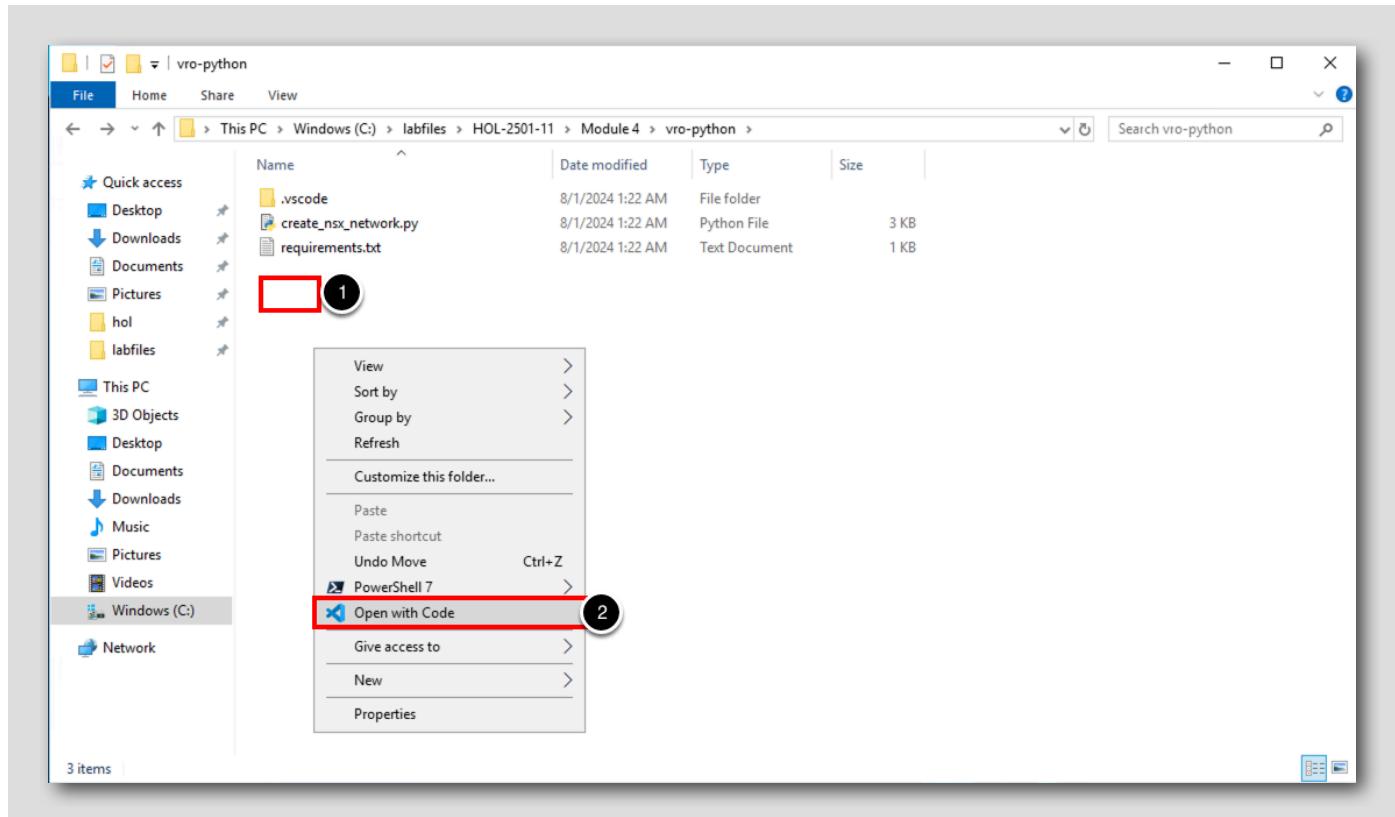
Now that we have confirmed that the two network addresses are not accessible we can progress with the development of our script. Let's move on to the second stage in our process, which is to test our script on our Windows desktop before we move it into Aria Automation Orchestrator. We will use Visual Studio Code to perform the next steps.

## Open the Working Directory



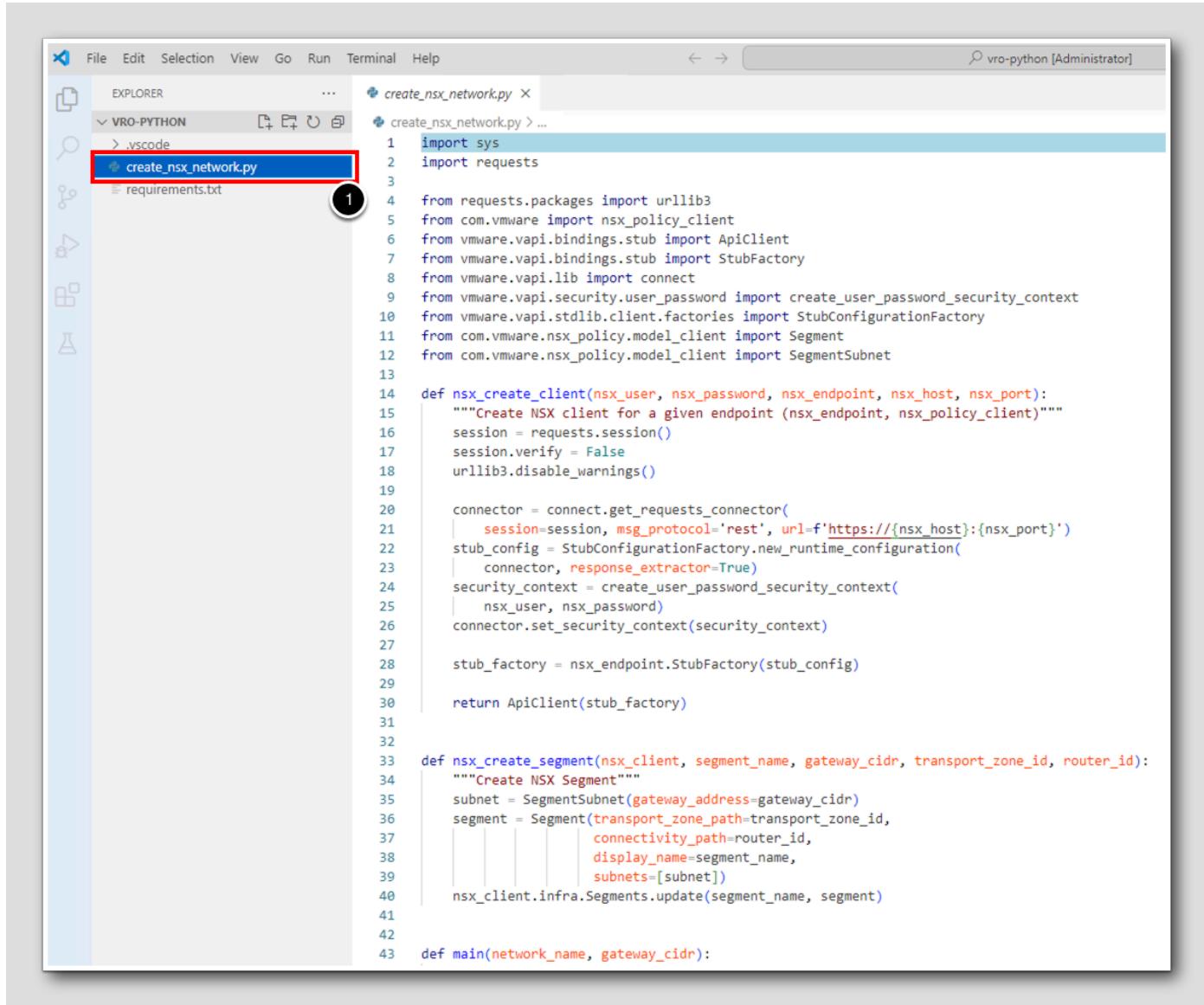
1. Open File Explorer from the taskbar (not shown) and click on **labfiles** in the quick access section
2. Navigate to **HOL-2501-11**
3. Navigate to **Module 4**
4. Navigate to **vro-python**

## Open the script with Visual Studio Code



1. Right-click inside the folder (but NOT on any file)
2. Click on Open with Code from the menu

## Open the Python file



The screenshot shows the VS Code interface with the title bar "vro-python [Administrator]". In the left sidebar, there's a tree view labeled "EXPLORER" with a red box around it. Under "VRO-PYTHON", the "create\_nsx\_network.py" file is highlighted with a blue selection bar and has a black circle with the number "1" next to it. Below it are ".vscode" and "requirements.txt". The main editor area shows the Python code for creating NSX segments.

```

1 import sys
2 import requests
3
4 from requests.packages import urllib3
5 from com.vmware import nsx_policy_client
6 from vmware.vapi.bindings.stub import ApiClient
7 from vmware.vapi.bindings.stub import StubFactory
8 from vmware.vapi.lib import connect
9 from vmware.vapi.security.user_password import create_user_password_security_context
10 from vmware.vapi.stdlib.client.factories import StubConfigurationFactory
11 from com.vmware.nsx_policy.model_client import Segment
12 from com.vmware.nsx_policy.model_client import SegmentSubnet
13
14 def nsx_create_client(nsx_user, nsx_password, nsx_endpoint, nsx_host, nsx_port):
15     """Create NSX client for a given endpoint (nsx_endpoint, nsx_policy_client)"""
16     session = requests.session()
17     session.verify = False
18     urllib3.disable_warnings()
19
20     connector = connect.get_requests_connector(
21         session=session, msg_protocol='rest', url=f'https:///{nsx_host}:{nsx_port}')
22     stub_config = StubConfigurationFactory.new_runtime_configuration(
23         connector, response_extractor=True)
24     security_context = create_user_password_security_context(
25         nsx_user, nsx_password)
26     connector.set_security_context(security_context)
27
28     stub_factory = nsx_endpoint.StubFactory(stub_config)
29
30     return ApiClient(stub_factory)
31
32
33 def nsx_create_segment(nsx_client, segment_name, gateway_cidr, transport_zone_id, router_id):
34     """Create NSX Segment"""
35     subnet = SegmentSubnet(gateway_address=gateway_cidr)
36     segment = Segment(transport_zone_path=transport_zone_id,
37                       connectivity_path=router_id,
38                       display_name=segment_name,
39                       subnets=[subnet])
40     nsx_client.infra.Segments.update(segment_name, segment)
41
42
43 def main(network_name, gateway_cidr):

```

1. Select the `create_nsx_python.py` python file from the file explorer pane

The NSX libraries for Python requires a number of dependencies. Due to our environment not having internet access, these have been preinstalled in advance using the command:

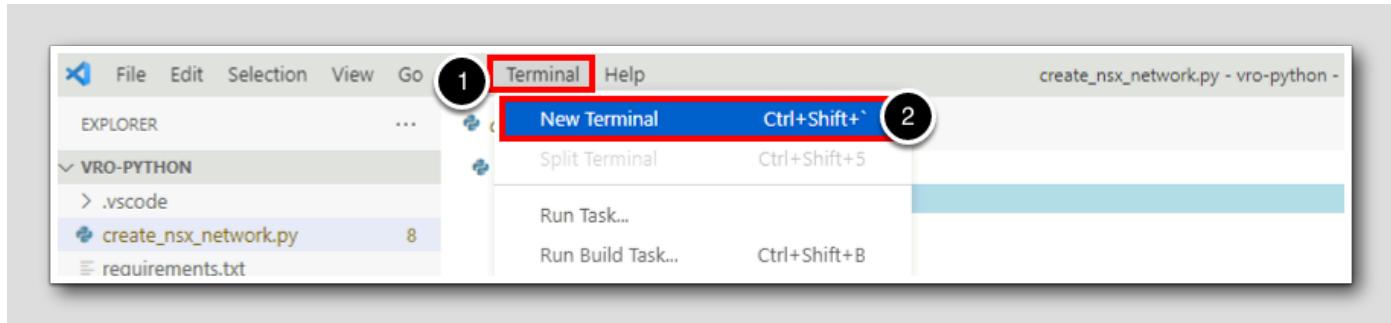
```
pip install --upgrade git+https://github.com/vmware/vsphere-automation-sdk-python.git
```

As the dependencies are already installed we do not need to re-run the command within our environment. Attempting to run the command will result in a failure as there is no internet access from within our environment.

The NSX libraries for Python are not available directly from the Python public repository (pypi). The NSX libraries for Python can be downloaded from from [GitHub](#).

## Open a Terminal

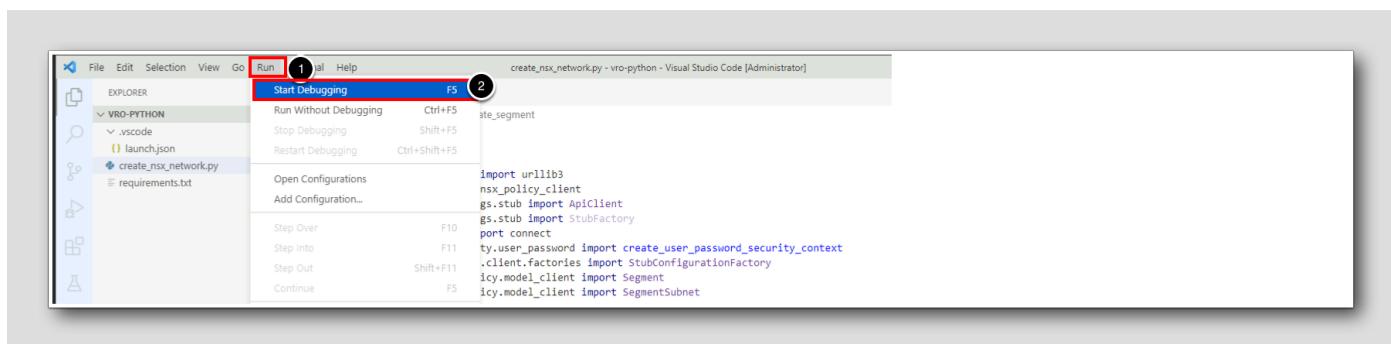
[218]



1. Select the **Terminal** menu
2. Click the option **New Terminal**

## Run the Script

[219]



1. Select the **Run** menu
2. Click the **Start Debugging** from the drop down menu. We use the debugging option so that Visual Studio Code loads our predefined configuration specified in the launch.json file in the vro-python directory

When starting debugging the script, if a pop up box appears in the bottom right corner of the screen warning that a virtual environment is not currently selected for your Python interpreter you can ignore this message or click on the "Don't show again" button to dismiss it.

## Check the Script Execution

[220]

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PowerShell 7.4.1
A new PowerShell stable release is available: v7.4.4
Upgrade now, or check out the release page at:
https://aka.ms/PowerShell-Release?tag=v7.4.4
PS C:\labfiles\HOL-2501-11\Module 4\vro-python> & C:\Users\Administrator\AppData\Local\Programs\Python\Python311\python.exe "C:\Users\Administrator\.vscode\extensions\ms-python.python.debugpy-2.9.2.8.0-win32-x64\bundle\adapters\..\..\debugpy\launcher" "65319" -- "C:\labfiles\HOL-2501-11\Module 4\vro-python\create_nsx_network.py" '192.168.121.1/24'
Connecting to NSX Manager...
Creating Network segment vscode-net - 192.168.121.1/24
PS C:\labfiles\HOL-2501-11\Module 4\vro-python>

```

- From the vs code configuration, 2 arguments have been added to the Python file execution. These are required parameters for the function `main()` from the script. The first parameter is the **segment name** (`vscode-net`), the second one is the **gateway IP address** with its associated netmask using the CIDR notation (`192.168.121.1/24`)
- The script ends by logging the creation of the new segment based on the two input values

This script has been preconfigured with the key values required to allow the connection to NSX and creation of a new segment, for example the credentials used for the connection. These are defined within the `main` function of the script on lines 45 to 49 of the script. Feel free to browse the script and review these values.

## Check the new network segment

[221]

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\labfiles\HOL-2501-11\Module 4\vro-python> ping 192.168.121.1 1
Pinging 192.168.121.1 with 32 bytes of data:
Reply from 192.168.121.1: bytes=32 time=3ms TTL=63
Reply from 192.168.121.1: bytes=32 time=3ms TTL=63
Reply from 192.168.121.1: bytes=32 time=1ms TTL=63
Reply from 192.168.121.1: bytes=32 time=1ms TTL=63
Ping statistics for 192.168.121.1:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 1ms, Maximum = 3ms, Average = 2ms
PS C:\labfiles\HOL-2501-11\Module 4\vro-python>

```

Now let's test the connectivity with the new NSX network segment.

- In the Terminal, enter the command `ping 192.168.121.1`
- The ping command returns the result that 4 Packets have been Sent and Received and 0 Lost. This confirms the new segment was successfully created

Now that we know this script works well on our Windows server, lets now add it into Aria Automation Orchestrator.

## Open Firefox Browser from Windows Quick Launch Task Bar

[222]

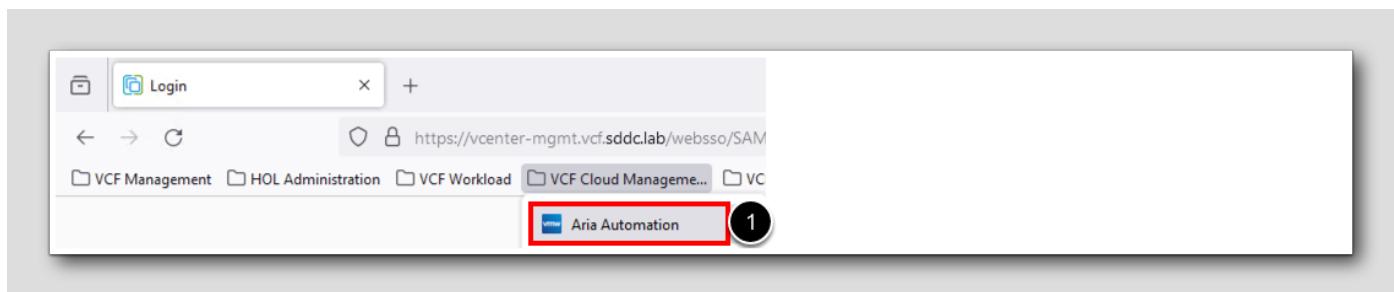


If Firefox is already opened and logged in to Aria Automation Orchestrator, move 5 steps forward to the Open the Environment section of the manual. Otherwise we are going to open the Firefox browser and login to Aria Automation Orchestrator.

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

## Launch Aria Automation

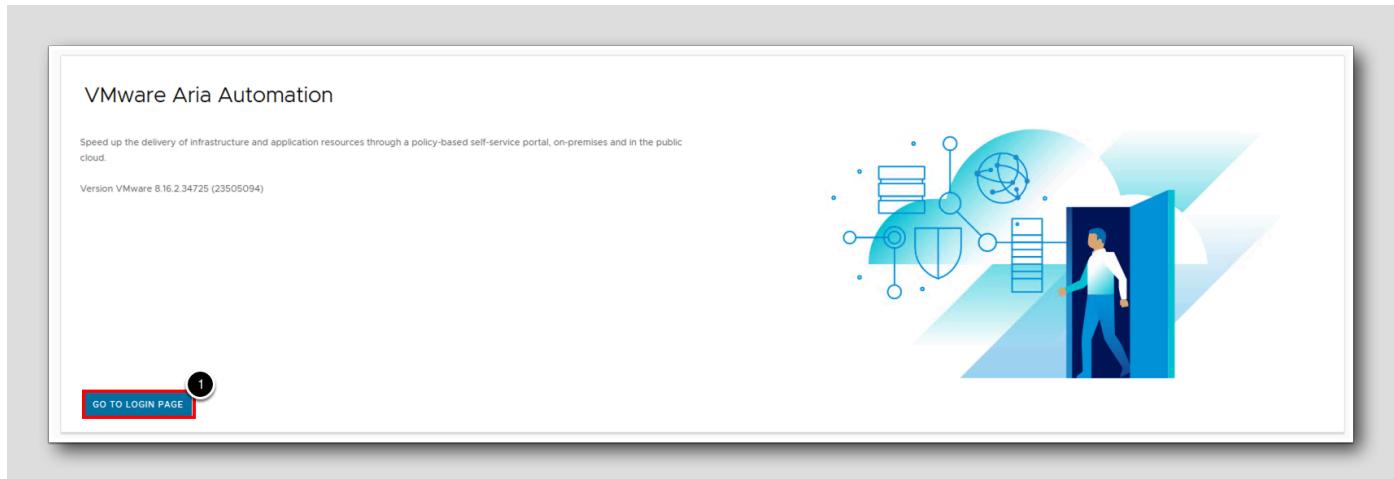
[223]



From within the Firefox web browser:

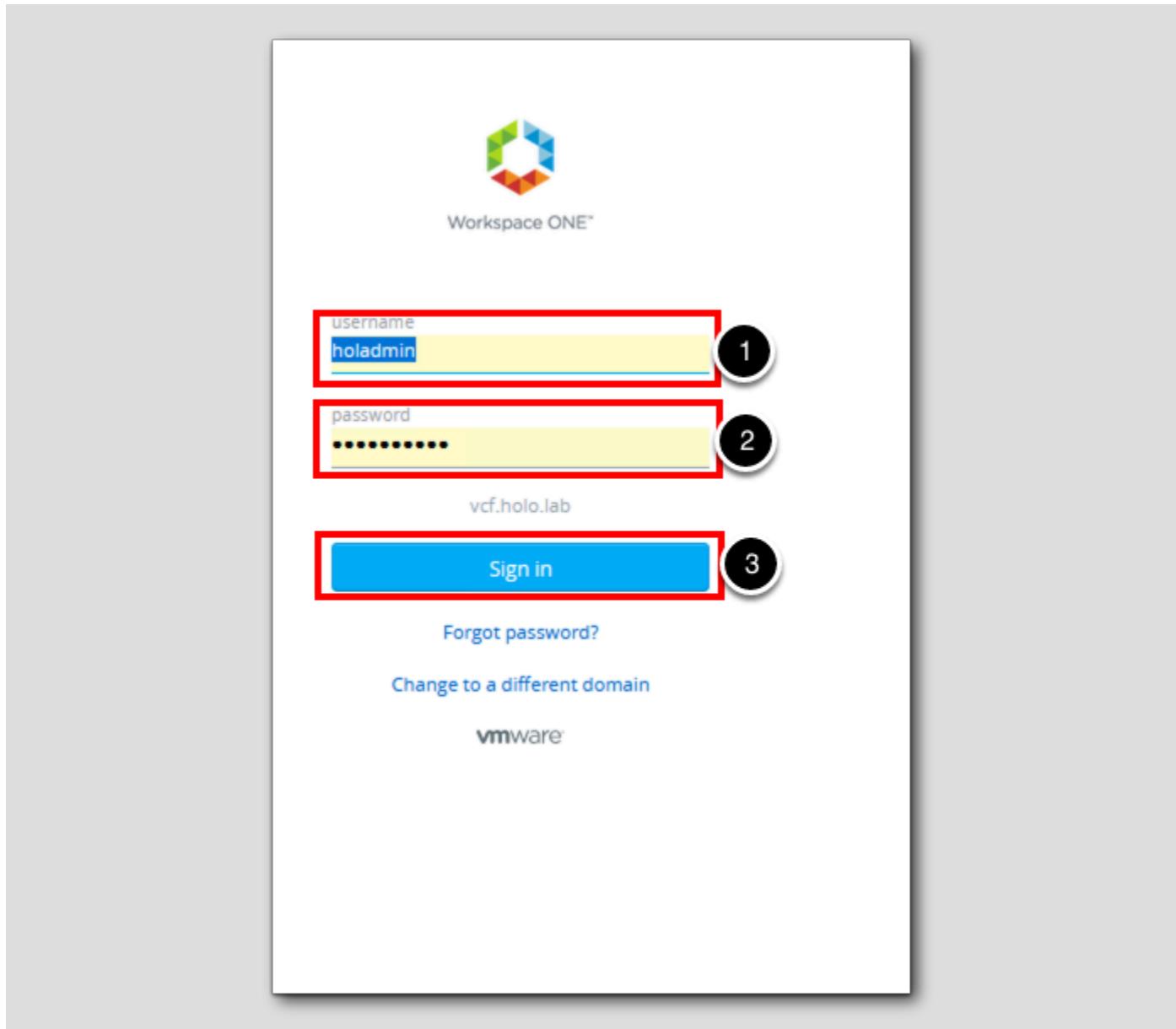
1. Click on VCF Cloud Management >Aria Automation from the bookmarks bar

## Open the Login Page



1. On the Aria Automation splash screen click GO TO LOGIN PAGE

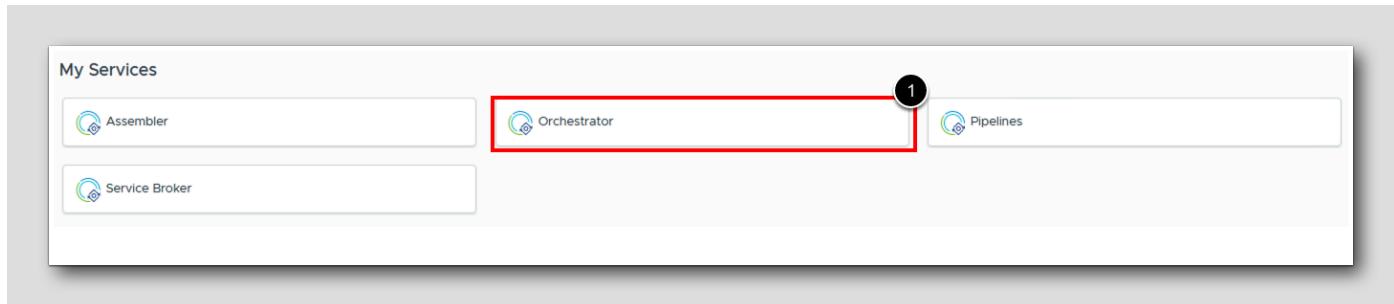
## Log in to Aria Automation



At the Workspace ONE login screen:

1. Enter username **holadmin**
2. Enter password **VMware123!**
3. Click **Sign In**

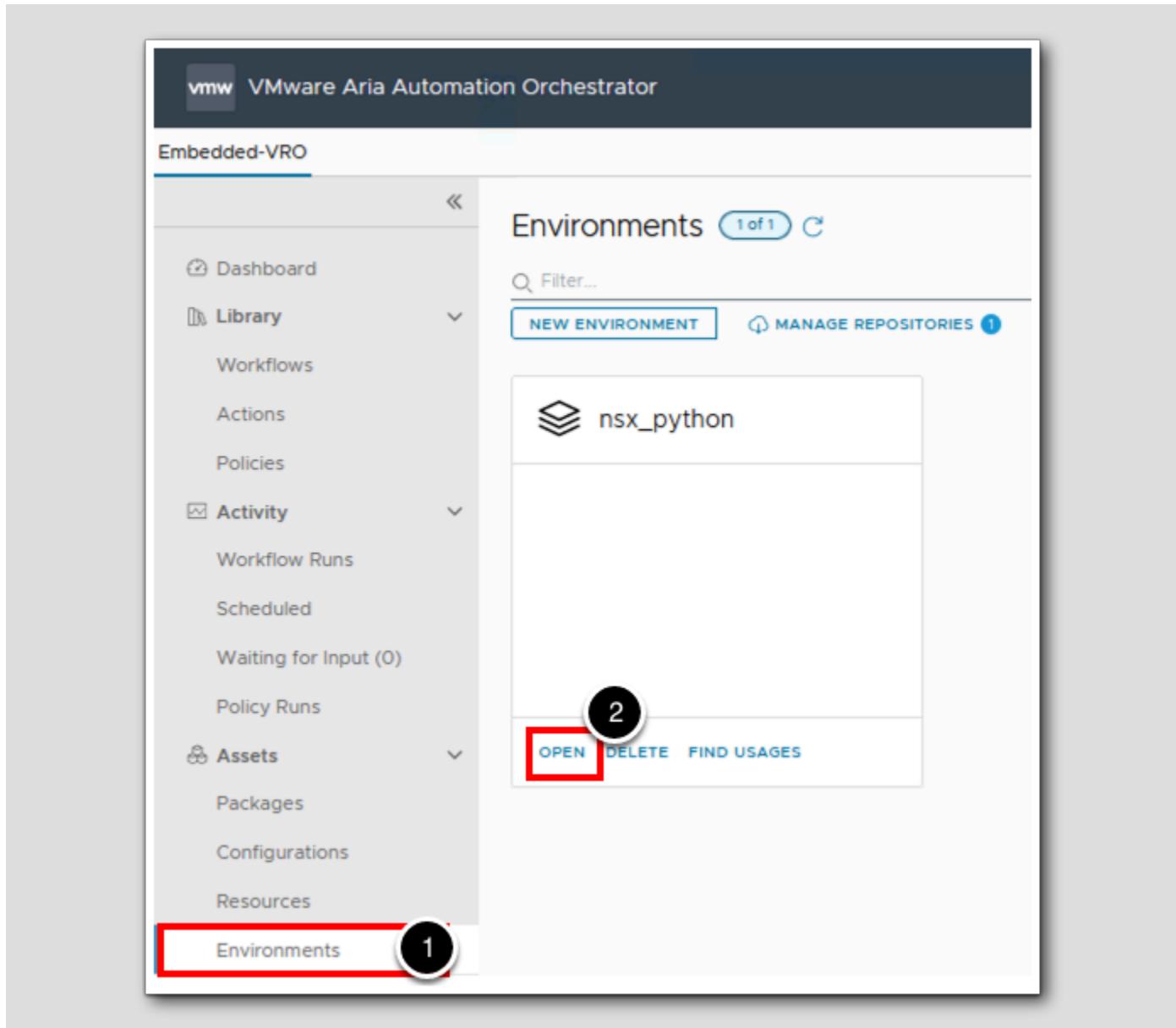
## Launch the Orchestrator Service



From within the Cloud Services Console, under **My Services**:

1. Click the Orchestrator service

## Open the Environment



Before we create and run our Action we first need to update the version of the Environment that has been preconfigured within Orchestrator. This step is required as in our lab Orchestrator does not always detect changes made within the Environment if the version number has not changed. This means it will not detect the presence of the Python dependencies we have pre-staged within the Orchestrator Environment and our action will fail to run. Let's review the contents of our Environment and update the version number:

1. Click on **Environments** under the Assets heading in the left menu pane
2. Click **OPEN** on the **nsx\_python** tile to open the Environment details page

Within Aria Automation Orchestrator an Environment can be used to define the Runtime Environment settings used by a script, the Dependencies required by a script (including the Repository from which they are downloaded) and any Environment Variables required when the script runs, or the Dependencies are downloaded.

When specifying Dependencies within an Environment we now have an option to specify what Repository Aria Automation Orchestrator should use for downloads. By default the Repository will be set to Default Public Repository. When the Runtime Environment is set to Python, the Default Public Repository will be the Python Package Index (PyPI). It is important to understand that Aria Automation Orchestrator must be able to access the site pypi.org via an internet connection in order to download Dependencies from the Default Public Repository. If it is not possible to connect to the internet, or we wish to use private packages we can configure Aria Automation Orchestrator to use a private repository. The private repository can be hosted on a private server accessible to Aria Automation Orchestrator, or via a SaaS solution such as GitLab, which provides a free Repository feature.

In our Aria Automation Orchestrator instance we are using a private repository hosted using the [devpi](#) open source project. We selected this solution to remove the dependency for internet access from the Aria Automation Orchestrator instance.

## Review the Environment

[228]

| Name                                      | Version | Repository                    |
|---|---------|-------------------------------|
| <a href="#">nsx-policy-python-sdk</a>     | 4.2.0   | HOL_Package_Repo(python.3.10) |
| <a href="#">nsx-python-sdk</a>            | 4.2.0   | HOL_Package_Repo(python.3.10) |
| <a href="#">vmware-vapi-common-client</a> | 2.52.0  | HOL_Package_Repo(python.3.10) |
| <a href="#">vmware-vapi-runtime</a>       | 2.52.0  | HOL_Package_Repo(python.3.10) |
| <a href="#">setupools</a>                 | 65.5.0  | HOL_Package_Repo(python.3.10) |
| <a href="#">urllib3</a>                   | 2.22    | HOL_Package_Repo(python.3.10) |
| <a href="#">requests</a>                  | 2.32.3  | HOL_Package_Repo(python.3.10) |

Total: 7

| Name                         | Value                    |
|------------------------------|--------------------------|
| <a href="#">trusted-host</a> | devpi-00212.vcf.sddc.lab |

Total: 1

Before we update the version of the Environment, lets take a moment to look at some of the key configuration values of the environment.

1. Click on the **Definition** tab
2. Here the Python version, memory limit and timeout values are defined. We can see the Python version has been set to 3.10 and the memory limit for our script configured at **512MB**
3. The Python dependencies we require for our Action are also defined with an explicit version number and which repository they are associated with. We can see that our dependencies are set to be downloaded from our private repository named **HOL\_Package\_Repo**
4. Under Environment Variables we can see that the **trusted-host** flag has been defined with the value set to the fqdn of our devpi server. This is required as we are using http for our repository server within the lab

## Increment the version of the Environment

[229]

nsx\_python Up To Date FIND USAGES FIND DEPENDENCIES DELETE

**General** Definition Download Logs Version History Audit

**1** Name nsx\_python

ID 1f6758e2-a3cd-4d4c-958d-e8d44ea84bd3

Description

Version **1.0.0** 2  
version can only be incremented.

Tags Enter a new tag i

Groups i

**3** **SAVE** **4** **CLOSE**

The screenshot shows a software interface for managing a Python package named 'nsx\_python'. The 'General' tab is active, indicated by a red box around it and a circled '1' above it. The 'Definition' tab is also present but not active. The 'Version' field contains '1.0.0', which is highlighted with a red box and circled '2' above it, with a note below stating 'version can only be incremented.'. The 'Tags' and 'Groups' sections are shown below, each with an 'Enter a new tag' input field and an information icon. At the bottom, there are 'SAVE' and 'CLOSE' buttons, both of which are highlighted with red boxes and circled '3' and '4' respectively.

Let's now increment the version of our Environment ready for use.

1. Click on the General tab
2. In the Version field update the version number to 1.0.0
3. Click SAVE. Wait for the green banner confirming the environment was successfully saved to appear at the top of the screen
4. Click CLOSE

## Create a new Action

[230]

The screenshot shows the VMware Aria Automation Orchestrator interface. On the left, there is a navigation sidebar with the following items:

- Dashboard
- Library
- Workflows (highlighted with a red box and circled '1')
- Actions (highlighted with a red box and circled '1')
- Policies
- Activity
  - Workflow Runs
  - Scheduled
  - Waiting for Input (0)
  - Policy Runs
- Assets

The main panel is titled "Actions" and shows a list of actions. At the top right of this panel is a button labeled "NEW ACTION" which is highlighted with a red box and circled '2'. Below this button, there is a card for the action "createNsxNetwork" with the identifier "com.broadcom.hal". At the bottom of the main panel, there are buttons for "OPEN", "DELETE", and "ACTIONS".

Now we have reviewed the configuration of the Environment we will use in Aria Automation Orchestrator lets create a new Action for running our script.

1. In the left pane select Library -> Actions

2. Click the NEW ACTION button

Name the new Action

[231]

Embedded-VRO

## createNSXNetwork

General    Script    Version History    Audit

Name 1 createNSXNetwork

Module 2 com.broad

ID 3 com.broadcom.hol

Description

Version 0.0.0

Tags  (i)

Groups  (i)

4

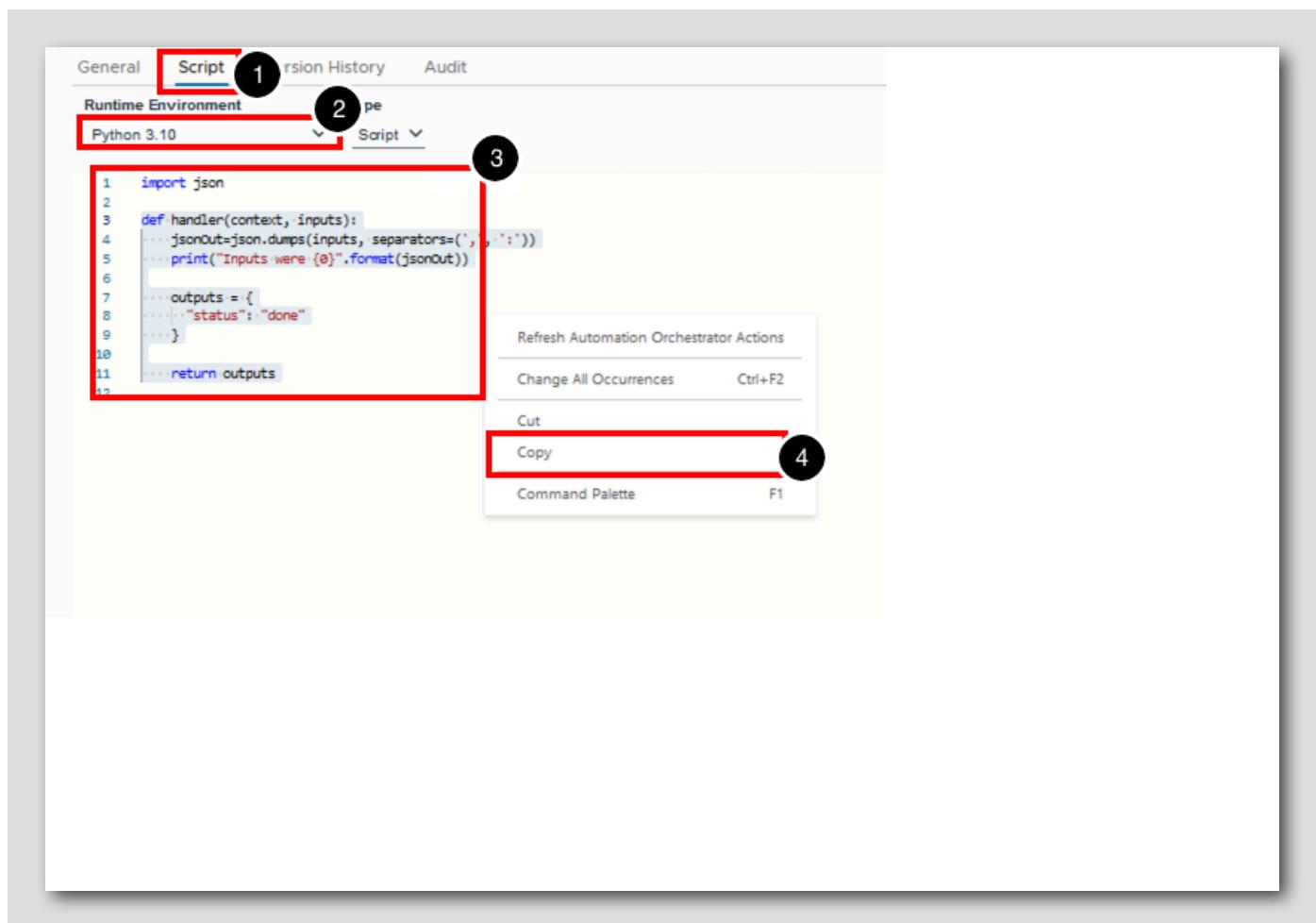
 CREATE VERSION CLOSE

HANDS-ON LABS MANUAL | 198

1. Enter the action name: createNsxNetwork
2. For the module, search for com.broad
3. Click the module com.broadcom.hol
4. Click the CREATE button
- 5.[Not shown] Wait for the green banner "Action Successfully Created"

## Copy Python sample script

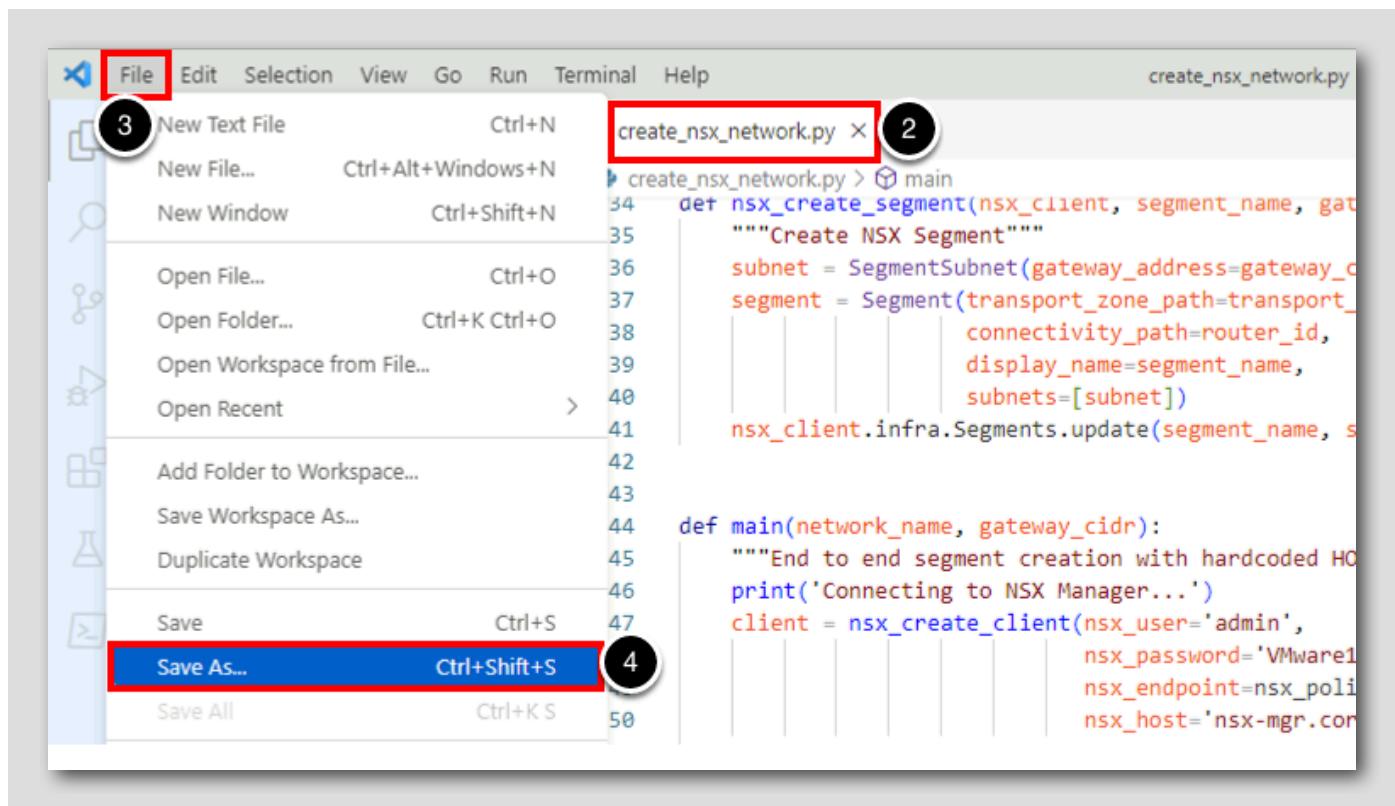
[232]



1. Select the Script tab
  2. Select Python 3.10 as the Runtime from the dropdown list
  3. Click in the script editor
  4. Right-click anywhere in the script editor and from the menu that appears select Copy
- Select all of the text except the first line `import json`

## Create a new script for Orchestrator

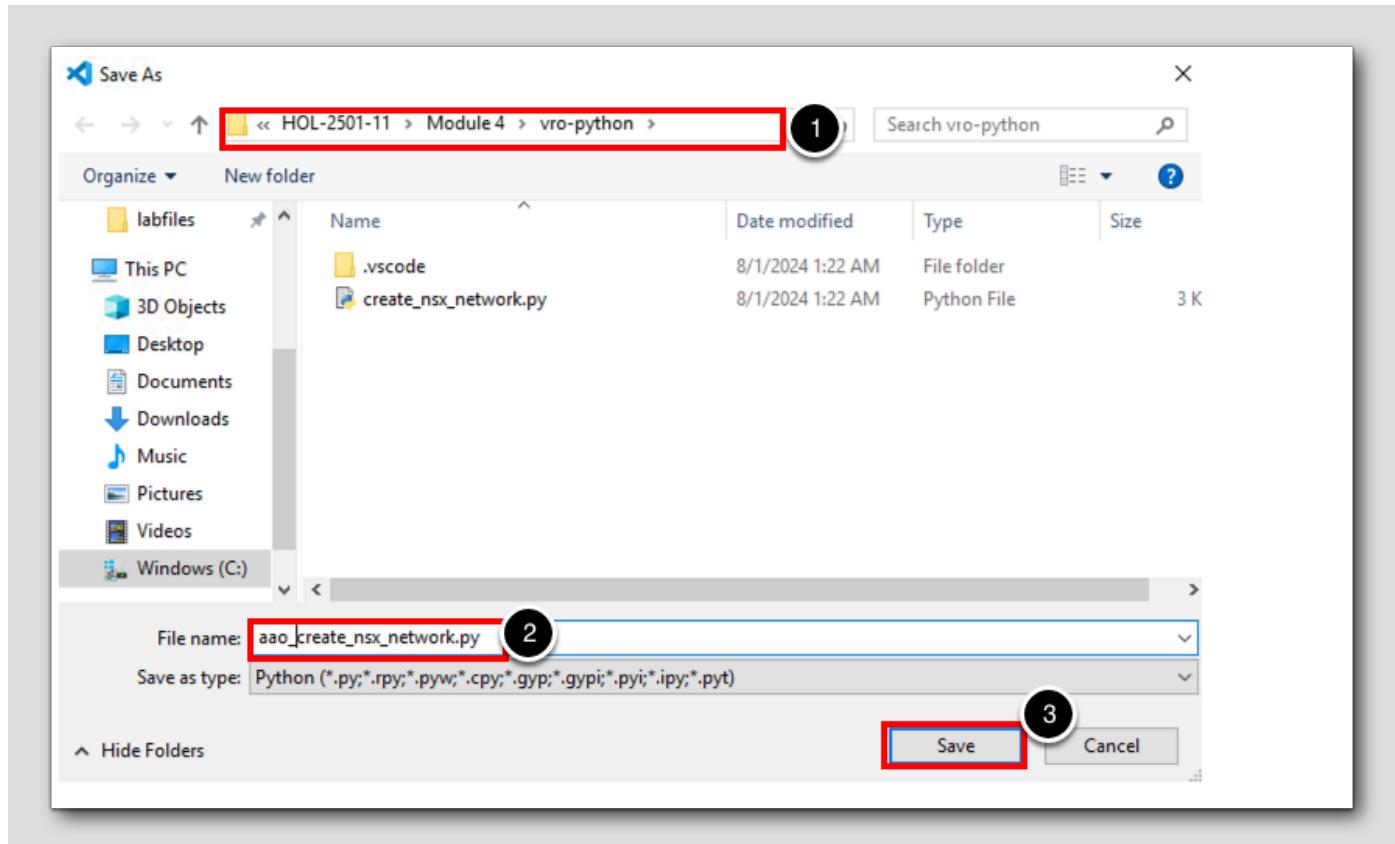
[233]



Aria Automation Orchestrator will require a function to handle the inputs to our script and functions for the creation of the NSX network. We will now amend our Python script so that it will work correctly when called by Aria Automation Orchestrator and save it as a new file.

1. [Not shown] Select Visual Studio Code from the task bar
2. Ensure the `create_nsx_network.py` file is selected
3. Click on the File menu
4. From the File menu select Save As

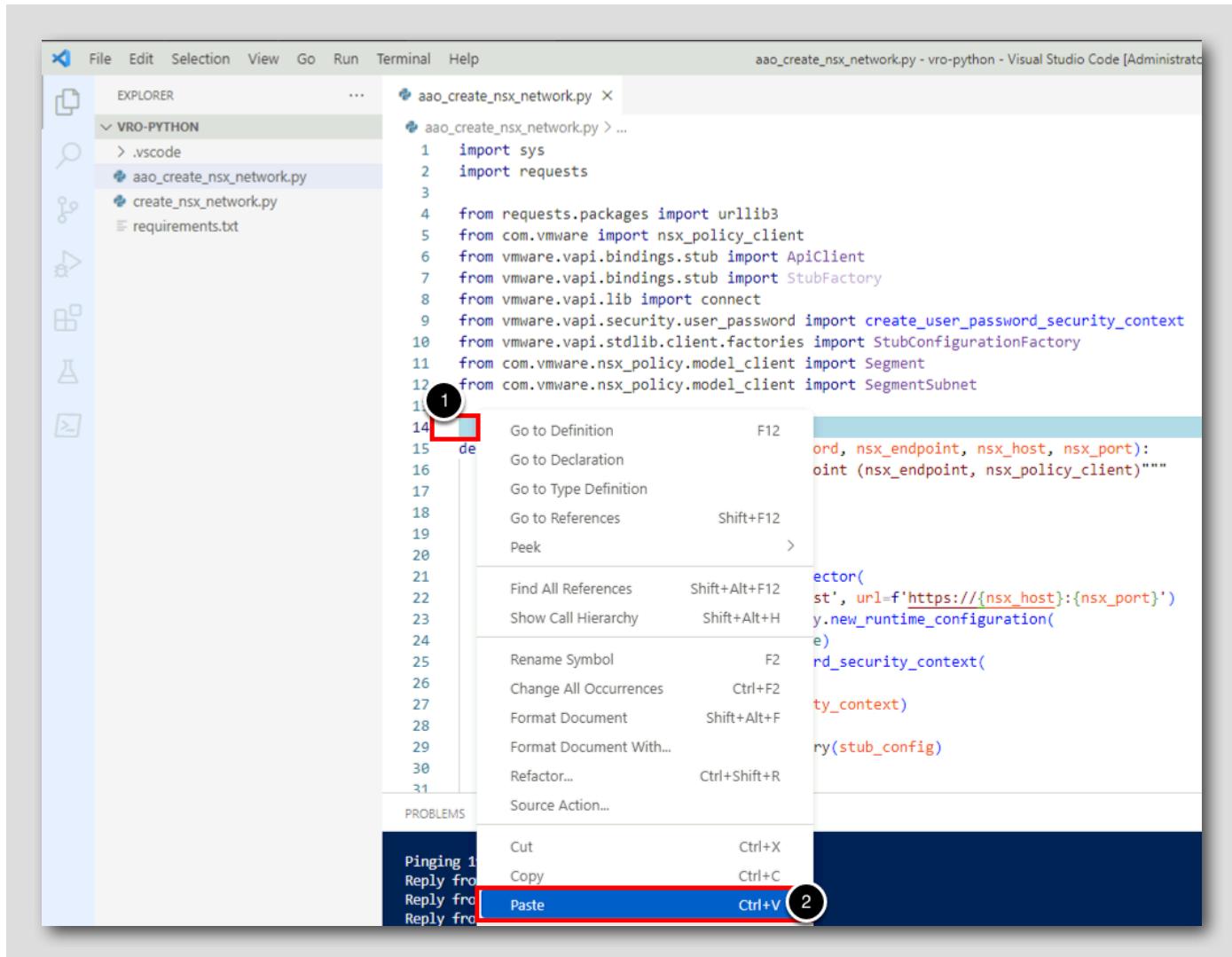
Name the new file



1. Ensure the directory is set to labfiles\HOL-2501-11\Module4\vro-python
2. Name the file **ao\_create\_nsx\_network.py**
3. Click on **Save** to create the file

## Paste script content

[235]



1. Click in the content window for the `aao_create_nsx_network.py` script on line 13 just above the code `def nsx_create_client(nsx_user, nsx_password, nsx_endpoint, nsx_host, nsx_port):`
2. Right click and select **Paste** from the menu that appears to paste the content that has been copied from Orchestrator.

Make sure that the indentation is the same as the source formatting and looks like the code block below:

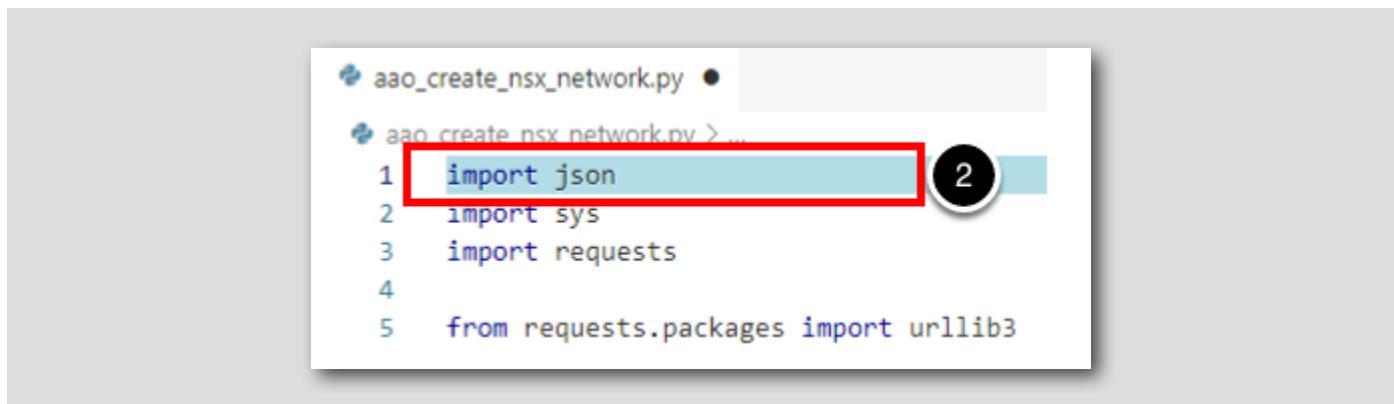
```
<p>
def handler(context, inputs):
    jsonOut=json.dumps(inputs, separators=(',', ':'))
    print("Inputs were {0}".format(jsonOut))

    outputs = {
        "status": "done"
    }

    return outputs
</p>
```

Add import directive to the script

[236]



The code in Orchestrator contained an import statement that is not present in our Python script, we choose not to copy it with the code for our function as we want to keep all import statements together at the start of our script. Lets now update our script to include the additional import statement

1. [not shown] Click into the start of line 1 of the script and press **Enter** to insert a new line
2. In the new line add the code `import json`

## Add a call to the main function



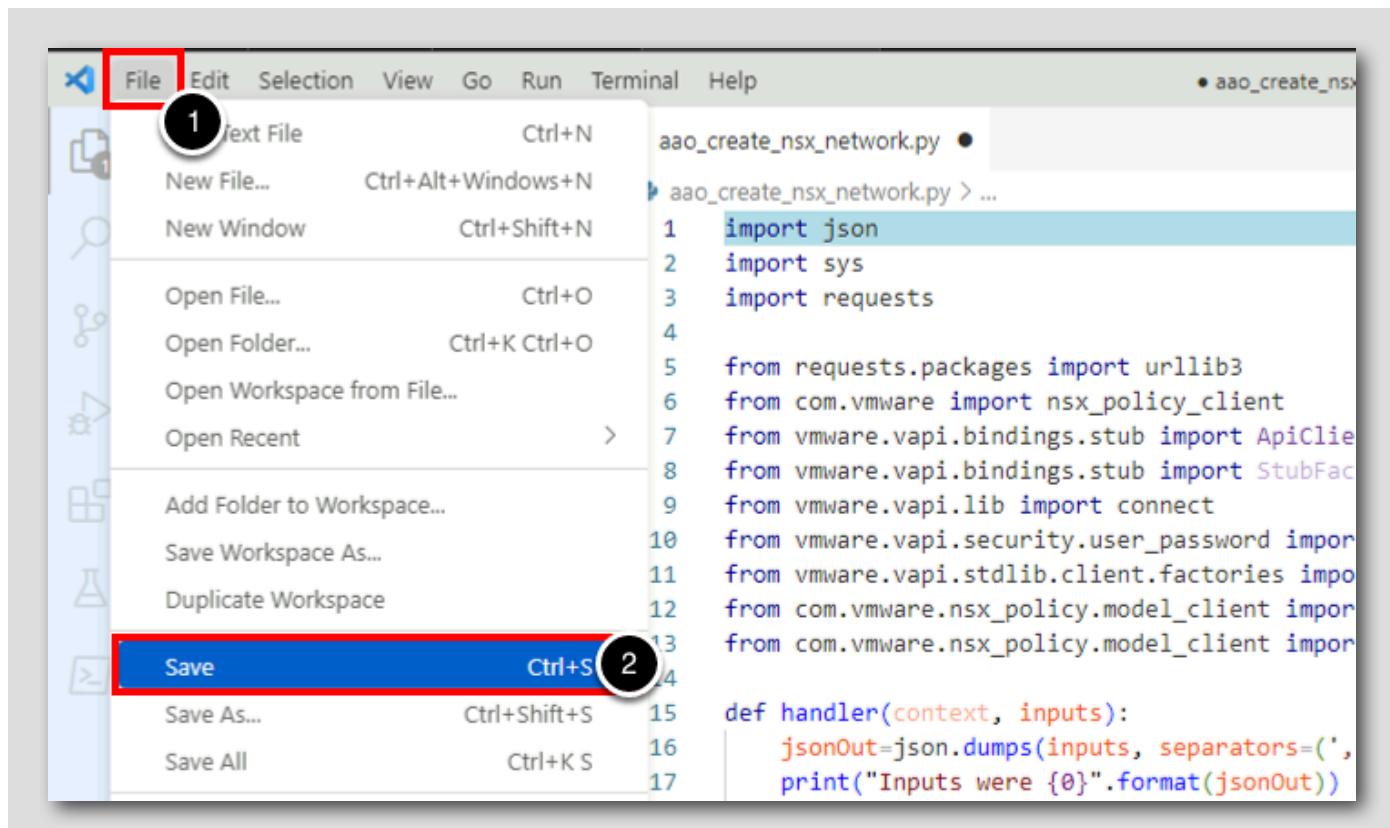
```
18     outputs = {
19         "status": "done"
20     }
21
22     main(inputs['name'],inputs['gateway']) 2
23     return outputs
24
25 def nsx_create_client(nsx_user, nsx_password, nsx_endpoint, nsx_host, nsx_port):
26     """Create NSX client for a given endpoint (nsx_endpoint, nsx_policy_client)"""
27     session = requests.Session()
28     session.verify = False
29     urllib3.disable_warnings()
```

One final addition needed to our script is to add a call inside the handler function to call the main function, which is where the steps to create our new segment are defined.

1. [not shown] Within the script click into the start of line 22 which contains the code `return outputs` and press **Enter** to insert a new line
2. On the newly inserted line, press the **TAB** key if needed to indent the cursor inline with the `return` statement on line 23 and then enter the code `main(inputs['name'], inputs['gateway'])` to call our main function inside of the handler function code we pasted from Orchestrator

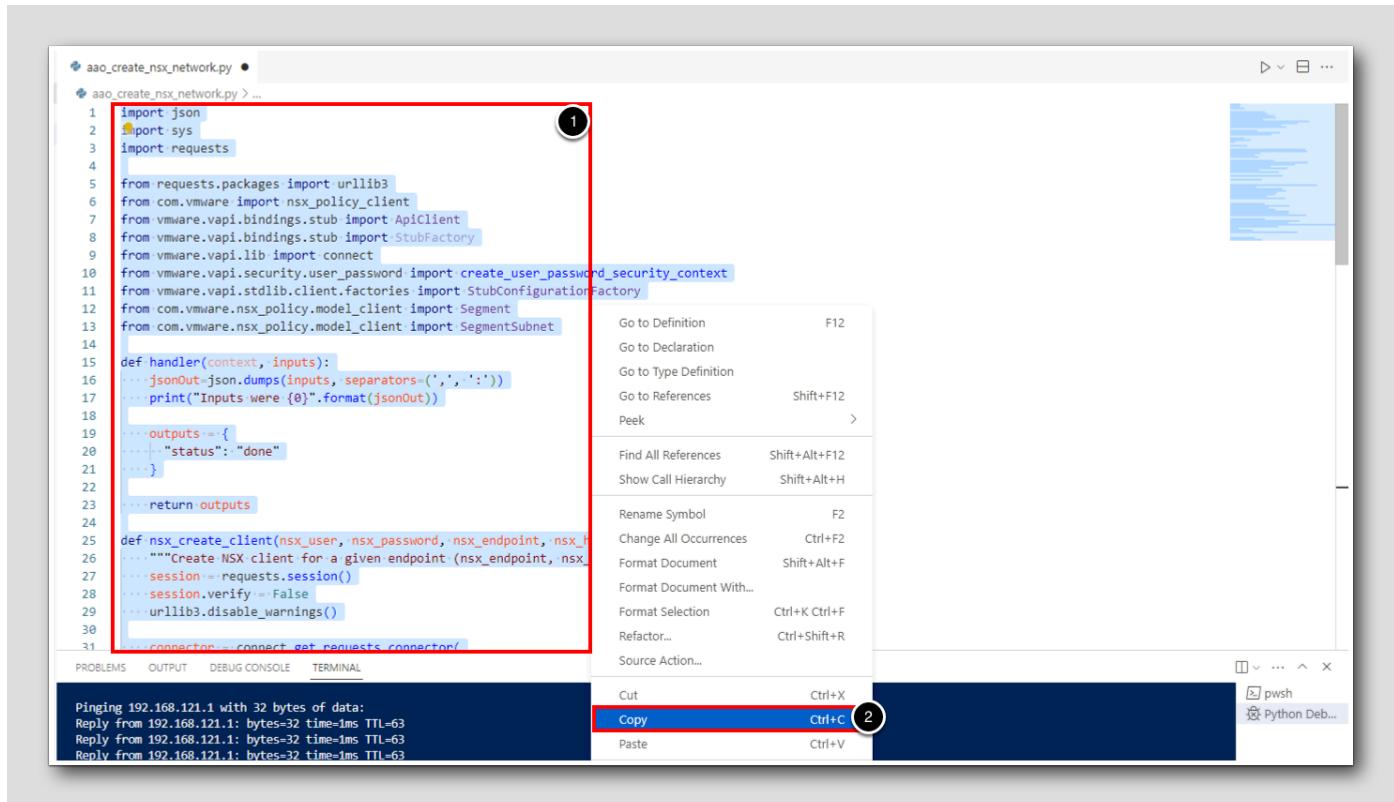
We have now merged the code required by Aria Automation Orchestrator with the code from our Python script. We are ready to save our script and add it to Aria Automation Orchestrator for testing.

## Save the file



1. Select the File menu
2. Click Save to save the file

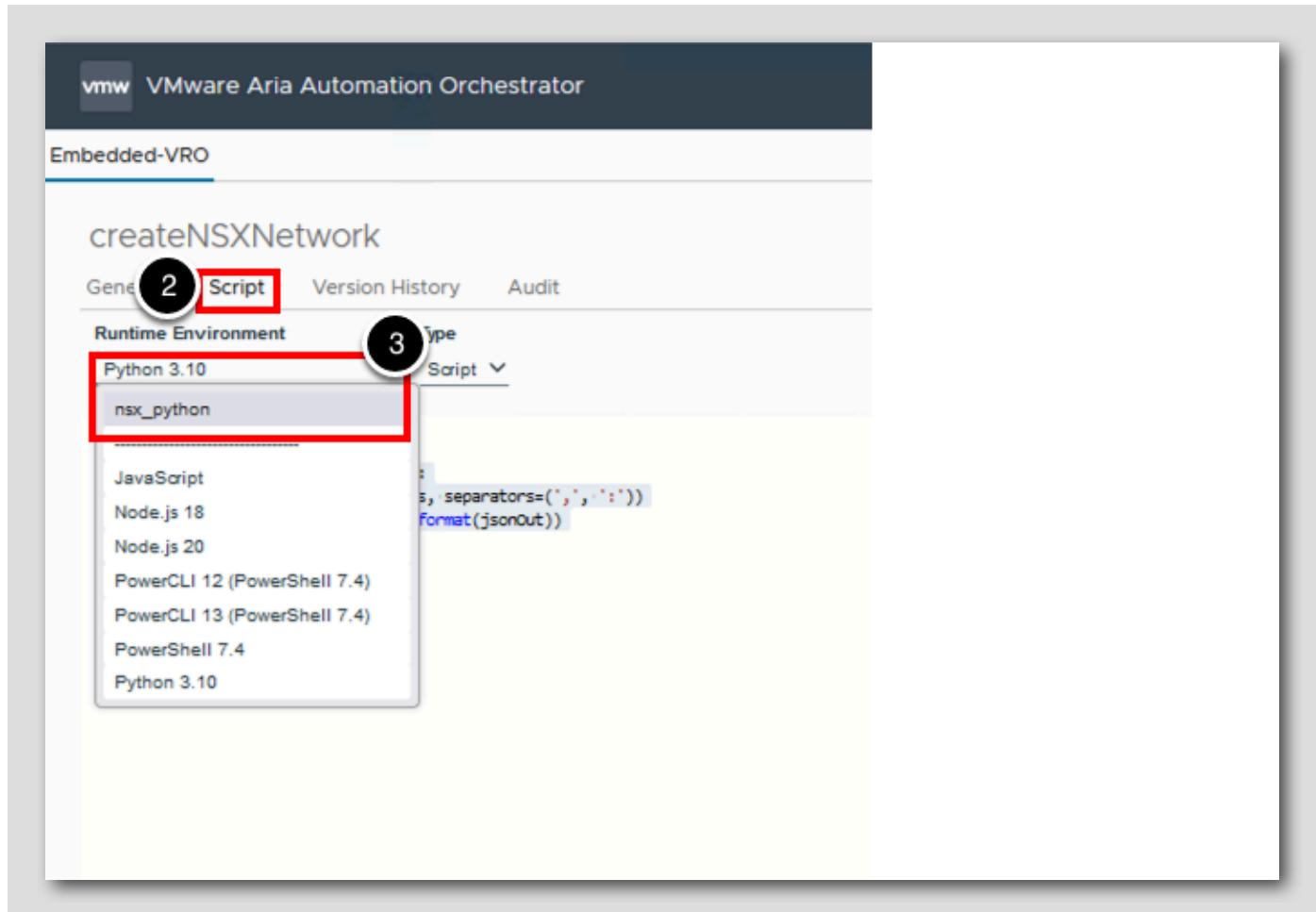
## Copy the script



With our Action configured with the correct Environment lets copy the contents of our updated Python script ready to add it into our new Action.

1. Click inside the `aao_create_nsx_network.py` file and press **CTRL+A** to select all of the script contents
2. Right click and from the menu that appears select **Copy** to copy the script contents

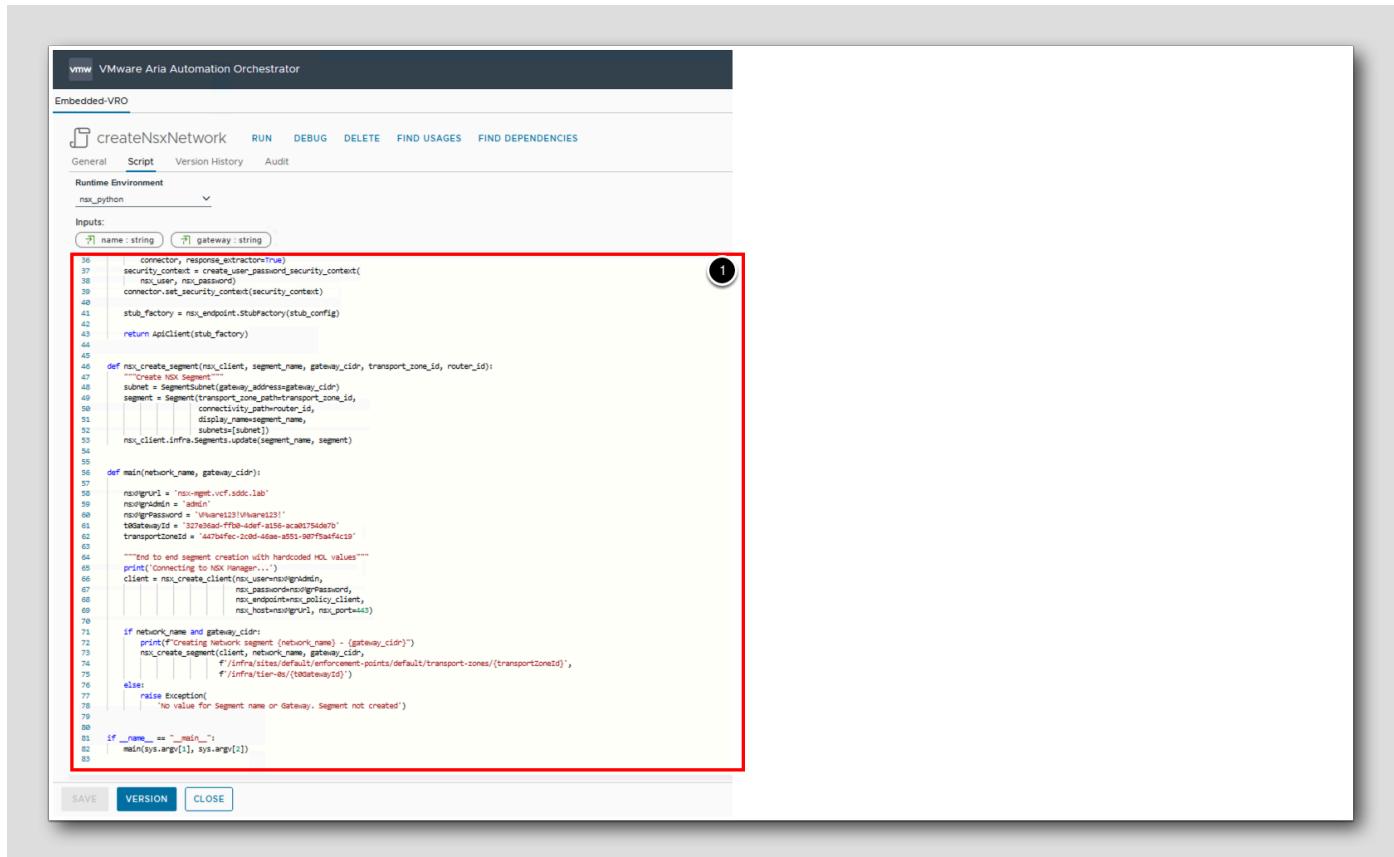
## Configure Action runtime



Before we add our code to the Action we need to configure it to use our Environment with the pre-staged dependencies. If we were to leave the Action as using the default Python runtime the Python dependencies would not be available and our Action would fail.

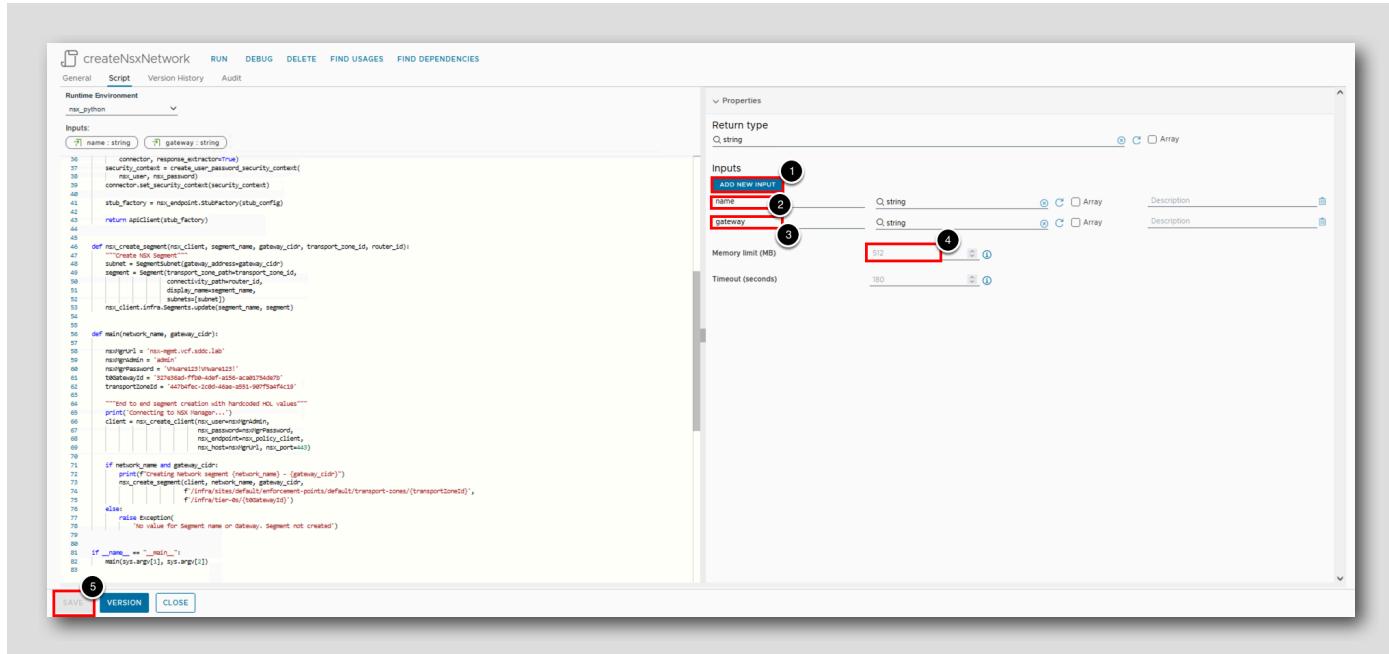
1. [not shown] Switch back to Orchestrator UI by selecting the Firefox tab from the Windows taskbar
2. Select the **Script** tab if it is not already selected
3. Click on the dropdown arrow for the Runtime Environment and select **nsx\_python**

## Add the script to the Action



1. Click inside the scripting pane and delete the example function code including the import json statement in the first line
2. Press CTRL+V on the keyboard to paste the script contents into the Action. Note there is no paste option on the right click menu in Orchestrator.

## Configure Action parameters



The final task before we are ready to save and test our Action is to add input parameters to capture the name of the network segment and the gateway address that should be passed to the script.

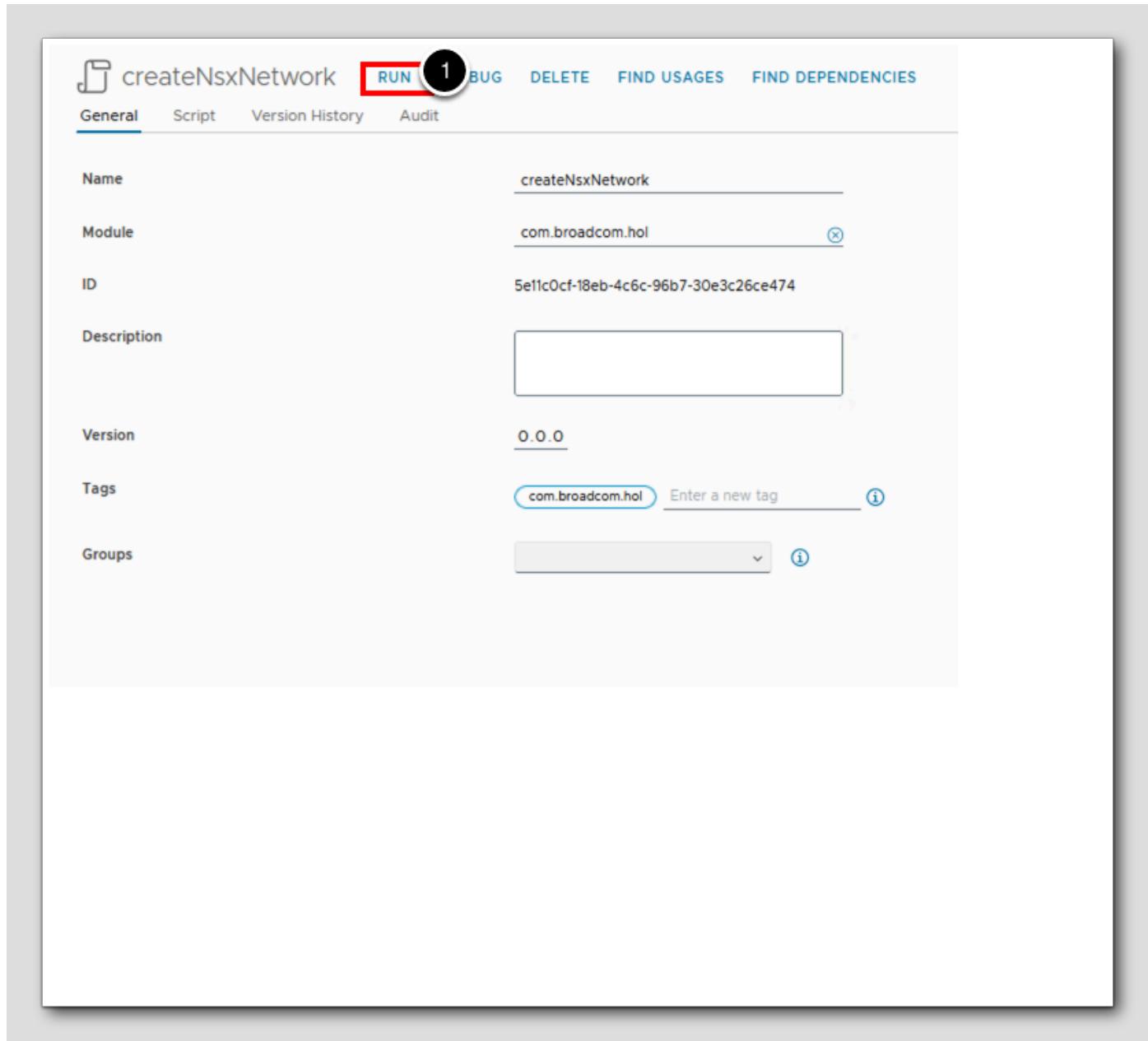
1. Click **twice** the ADD NEW INPUT button in the right pane to create 2 inputs parameters
2. Name the first input: **name**
3. Name the second input: **gateway**
4. Confirm the Memory limit is set to 512 MB as per our nsx\_python environment configuration
5. Click the **SAVE** button

The name of the inputs must match the name that the Python script expects.

The action might take up to 1 min to save. Wait for the green banner saying "Action successfully saved". You may notice a short delay between the blue banner disappearing off the screen and the green banner appearing. This is normal behavior within our environment that has limited resources.

When successfully saved, the RUN button shouldn't be greyed out anymore.

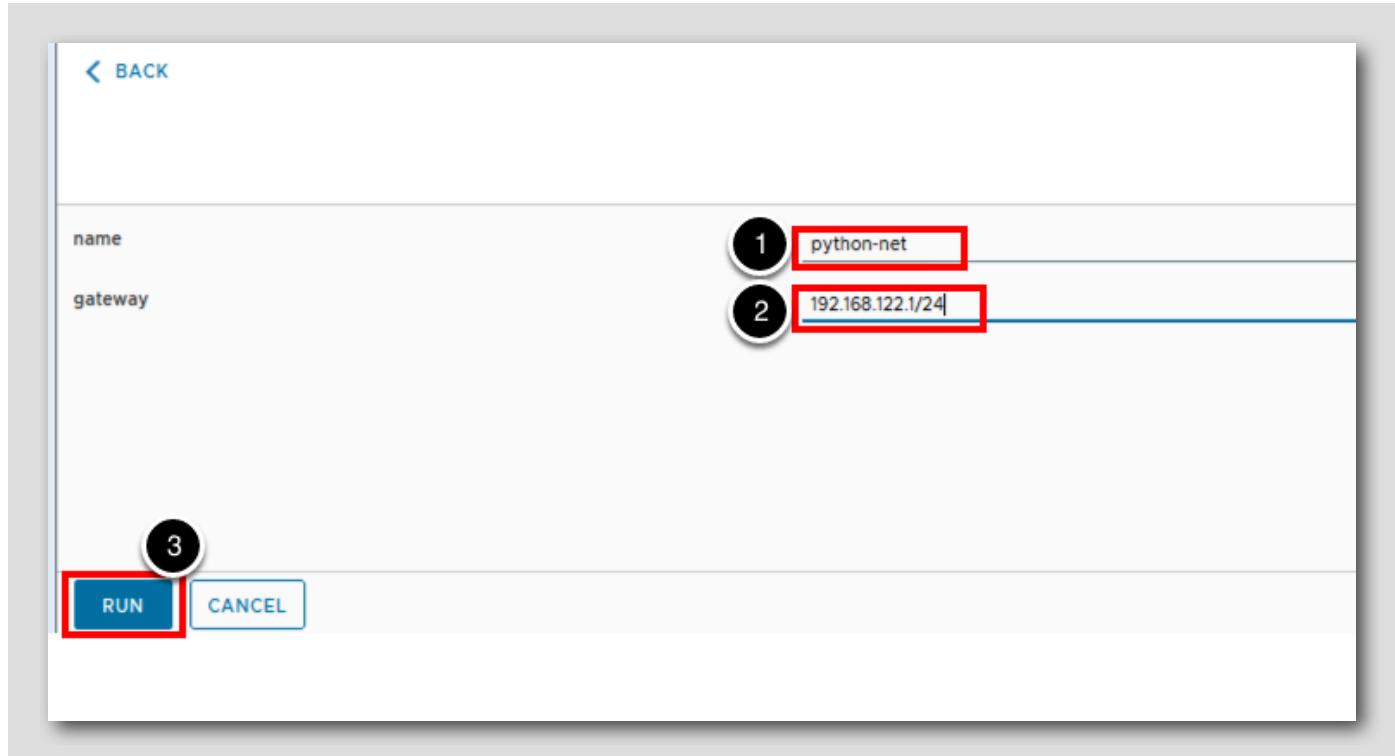
## Run the Action



Once the green confirmation banner appears our Action has been successfully saved and we are ready to test it.

1. Click RUN to launch the Action

Enter the input values



1. Enter the network name as **python-net**
2. Enter the gateway (CIDR notation) as **192.168.122.1/24**
3. Click the **RUN** button

Wait for the action to be completed

The screenshot shows the Aria Automation Orchestrator interface for an 'Embedded-VRO' action named 'createNsxNetwork'. The 'Script' tab is selected, and the status bar at the top right shows a green 'Completed' icon with a '1' count. The script code is displayed in the main pane, and the 'Logs' tab is selected in the bottom navigation bar, showing log entries related to the script's execution.

```

Embedded-VRO
createNsxNetwork RUN DEBUG DELETE FIND USAGES FIND DEPENDENCIES Completed 1

General Script Version History Audit
Runtime Environment nsx_python
Inputs:
  name: string gateway: string
  1 import json
  2 import sys
  3 import requests
  4
  5 from requests.packages import urllib3
  6 from com.vmware import nsx_policy_client
  7 from com.vmware.vapi_directory_client import Apiclient
  8 from com.vmware.vapiStubFactory import StubFactory
  9 from com.vmware.vapi.lib import connect
 10 from com.vmware.vapi.security.user_password import create_user_password_security_context
 11 from com.vmware.vapi_identity_factor import StubConfigurationFactory
 12 from com.vmware.nsx_policy.model_client import Segment
 13 from com.vmware.nsx_policy.model_client import SegmentSubnet
 14
 15
 16 def handle(context, inputs):
 17     jsonOut=json.dumps(inputs, separators=(',', ':'))
 18     print("Inputs were (%s)"%jsonOut)
 19
 20     outputs = {
 21         "status": "done"
 22     }
 23
 24     main(inputs["name"], inputs["gateway"])
 25
 26
Result / Inputs Logs 2
Loading...
 2024-08-05 00:38:28,900 -07:00 INFO Inputs were {"name":"python-net","gateway":"192.168.122.1/24"}
 2024-08-05 00:38:20,930 -07:00 INFO Connecting to NSX Manager...
 2024-08-05 00:38:20,931 -07:00 INFO Creating Network segment python-net - 192.168.122.1/24
  
```

Properties panel on the right shows:

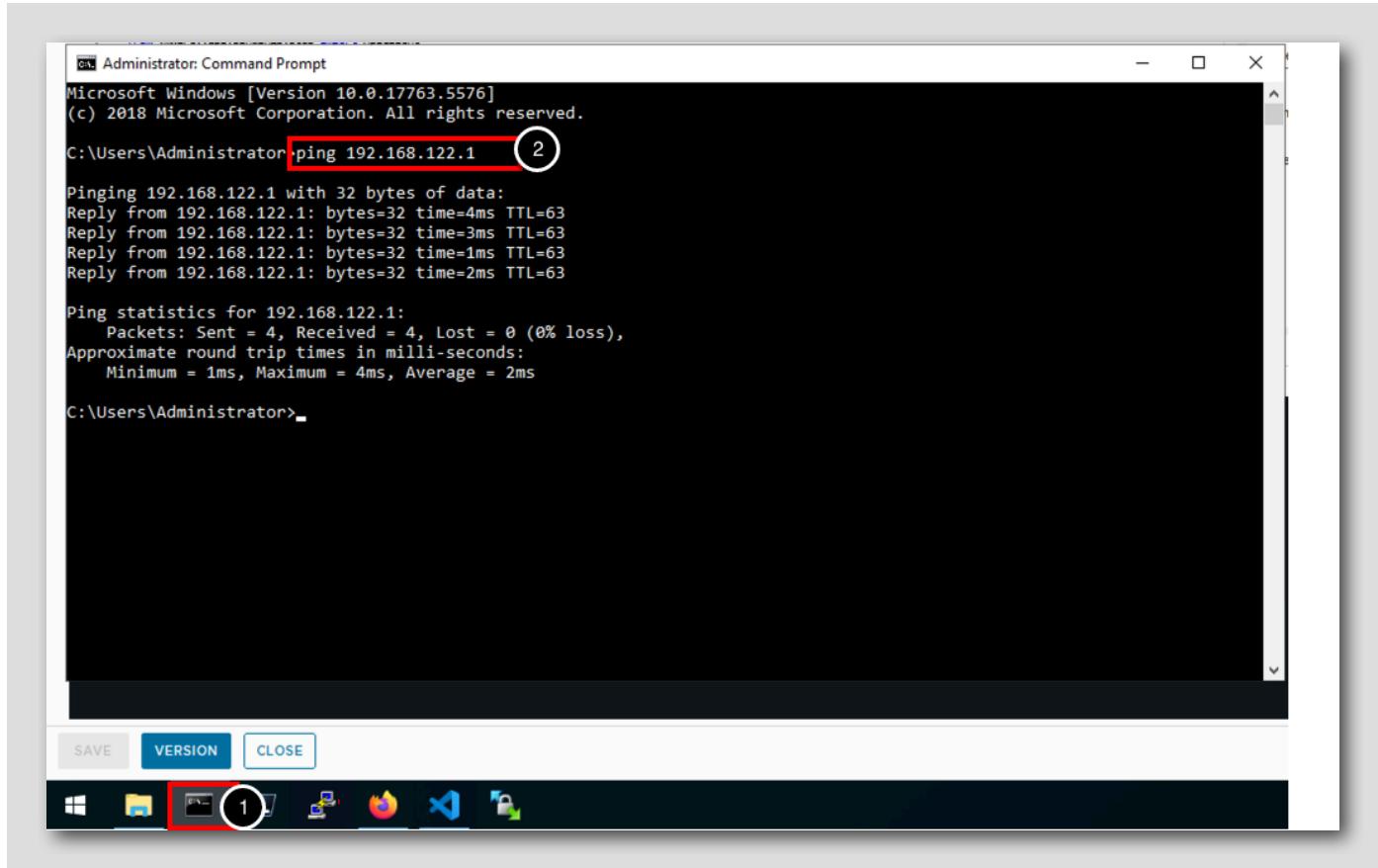
- Return type: Q\_string
- Inputs:
  - name**: Q\_string (Array)
  - gateway**: Q\_string (Array)
- Memory limit (MB): 512
- Timeout (seconds): 180

Buttons at the bottom: SAVE, VERSION, CLOSE.

1. Wait for the action to be completed and the green Completed icon to appear at the top of the Action
2. Click the Logs tab and observe the output of the python script directly in Aria Automation Orchestrator. We can see that the Action inputs match the values we entered when running the Action and two log entries for the connection to the NSX Manager and the creation of the Network segment.

The action may take a few minutes to complete. You can open the Logs tab and observe the updates once it reaches the point of our script where the first logging command is present.

## Ping the gateway



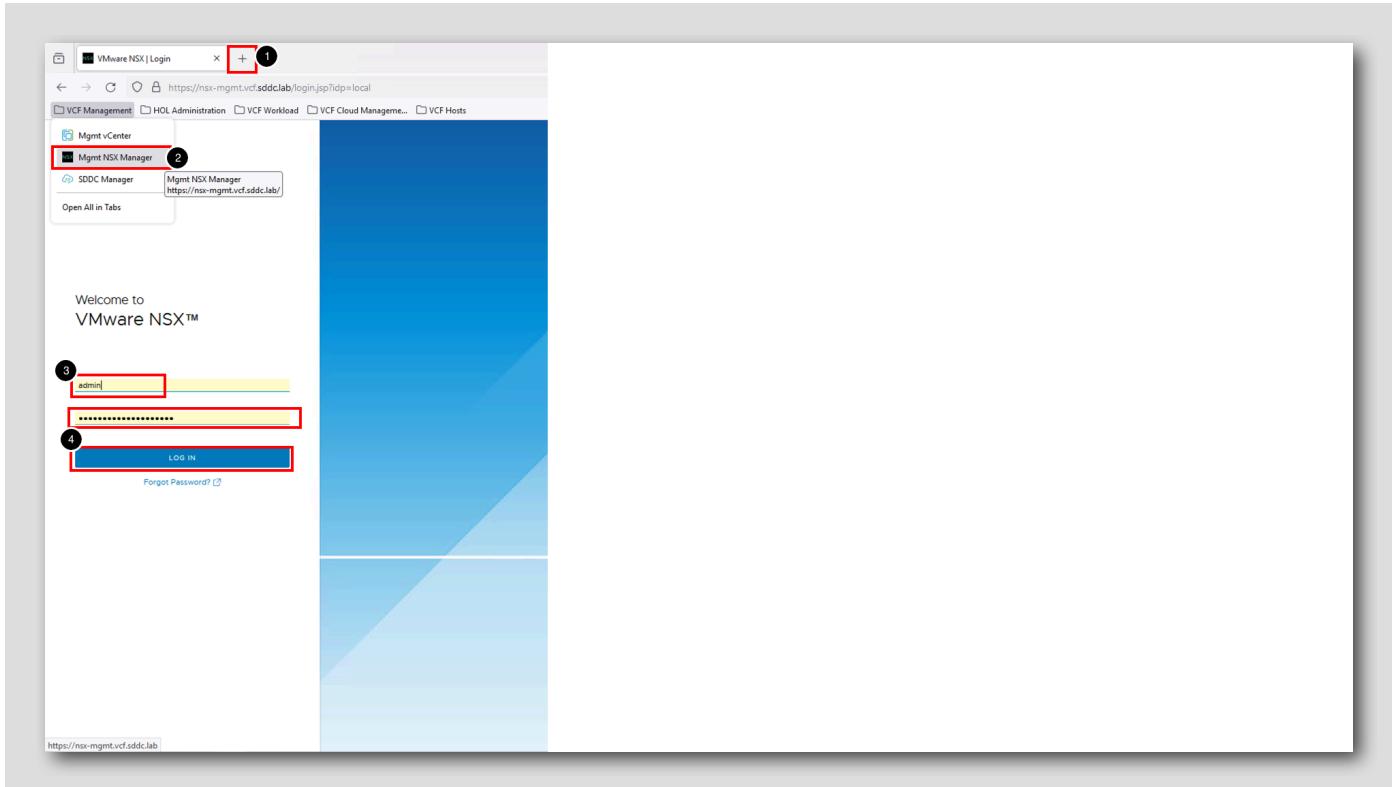
Now let's test the connectivity with the new NSX network segment using the ping command, exactly as we did when we ran our script in VSCode earlier in the exercise.

1. Open a Windows Command Prompt
2. Enter the command

```
ping 192.168.122.1
```

The gateway IP address of the new NSX segment should now respond to ping, confirming that the segment was successfully created.

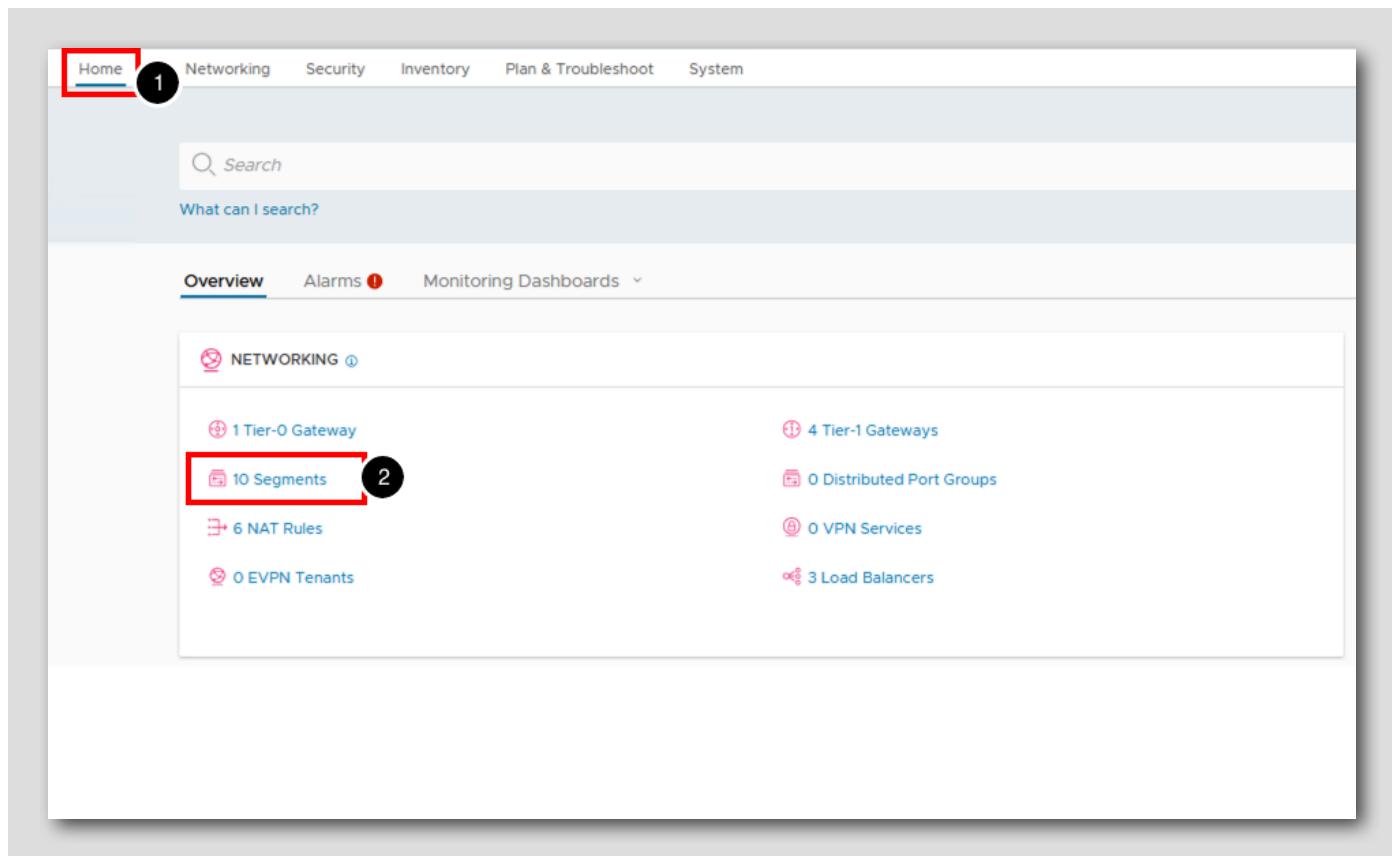
## Verify in NSX



We can also verify the creation and configuration of the new segment directly in the NSX Manager GUI.

1. Open a new browser tab
2. Click the VCF Management>Mgmt NSX Manager bookmark
3. If not already pre-filled, enter the credentials:  
username = **admin** password = **VMware123!VMware123!**
4. Click the LOG IN button

## Observe the Segment



1. In the Home tab
2. Click the **Segments** link

Note the number of Segments listed in your lab may differ from the screenshot.

Observe the newly created segment

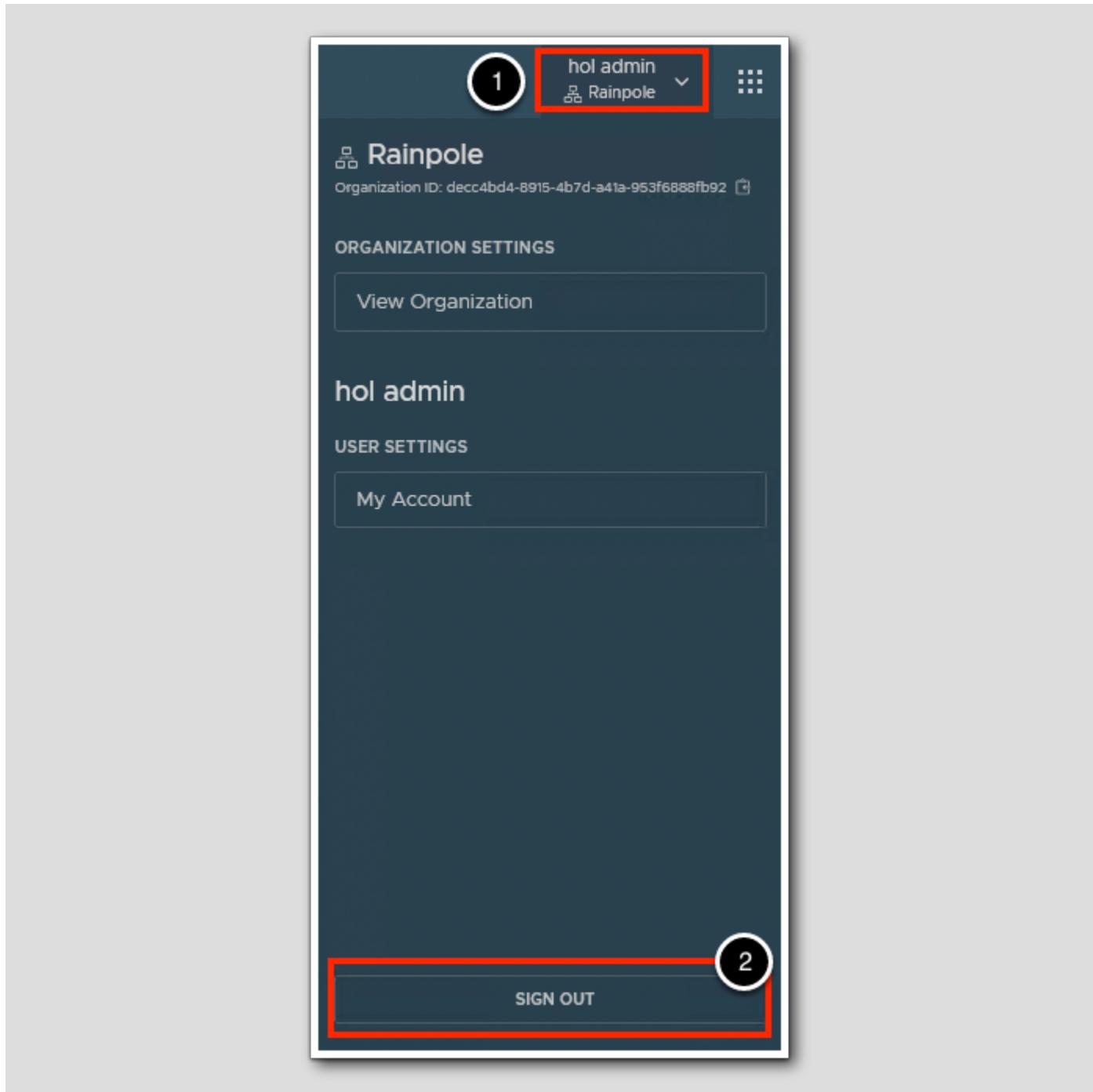
| Name   | Connected Gateway   | Transport Zone                     | Subnets          |
|--|---|------------------------------------|------------------|
| hol-prod   | VLC-Tier-0  | mgmt-domain-tz-overlay01   Overlay | 10.64.12.1/24    |
| python-net   | VLC-Tier-0  | mgmt-domain-tz-overlay01   Overlay | 192.168.122.1/24 |
| region-seg01   | VLC-Tier-1  | mgmt-domain-tz-overlay01   Overlay | 10.50.0.1/24     |
| seg-domain-c4001:7f78b6fb-b8cd-4be4-86a9-226e2a6056f2-rainpole-rtr | t1-domain-c4001:7f78b6fb-b8cd-4be4-86a9-226e2a6056f2-rainpole-rtr | mgmt-domain-tz-overlay01   Overlay | 10.244.0.17/28   |
| VCF-edge_EC-01_segment_uplink1_11                                  | None  | VCF-edge_EC-01_uplink-tz   VLAN    | Not Set          |
| VCF-edge_EC-01_segment_uplink2_12                                  | None  | VCF-edge_EC-01_uplink-tz   VLAN    | Not Set          |

1. On the Segments page we can see the newly created segment

As demonstrated in this lesson, Aria Automation Orchestrator can run pretty much any existing Python scripts. There are only 2 requirements needed:

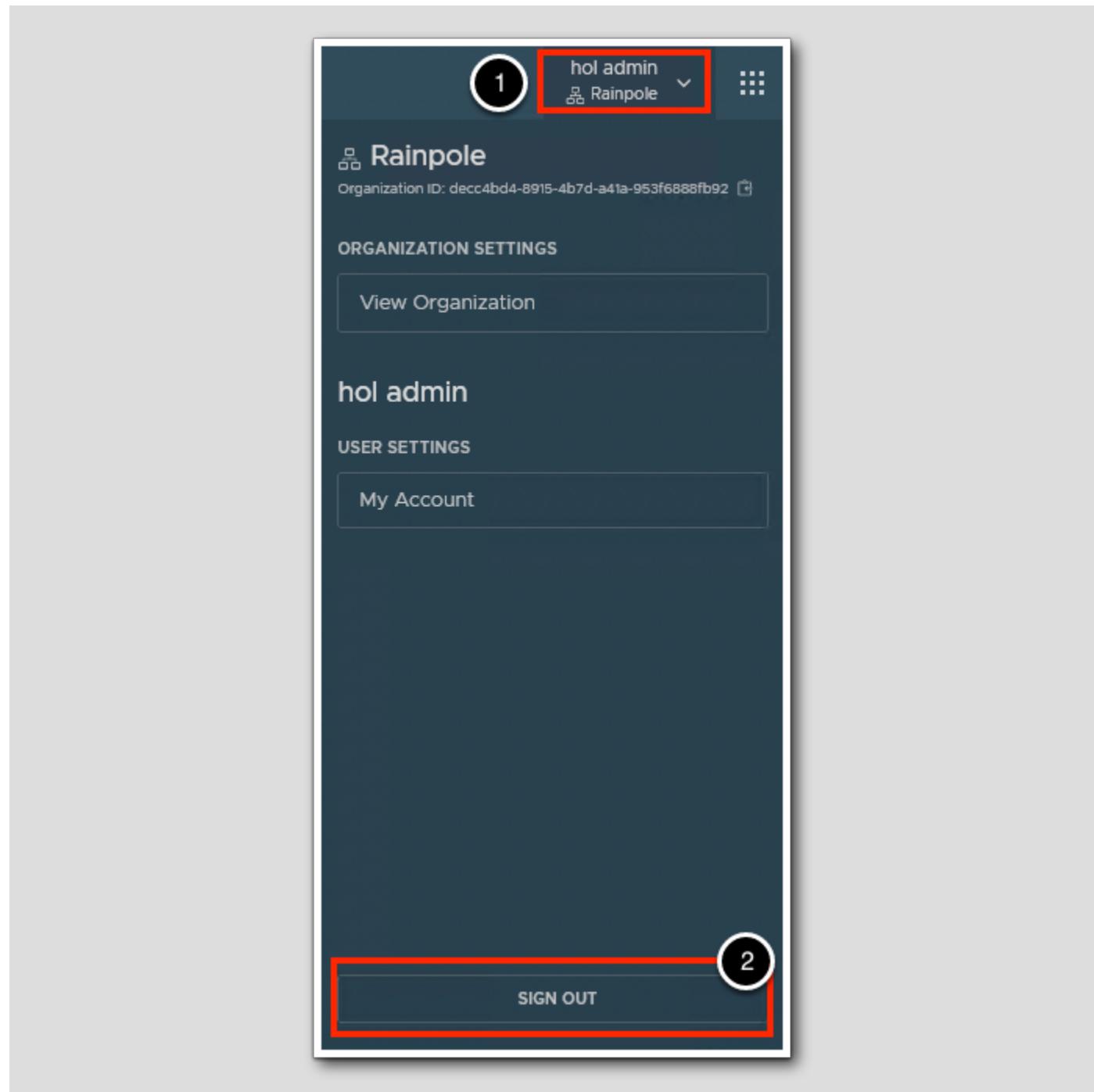
- Bundle all the dependencies in a zip file or provide Aria Automation Orchestrator with access to a repository from which they can be downloaded.
- Create a Python script with a function that will pass the parameters from Aria Automation Orchestrator to any other functions within the script.

## Log Out of Aria Automation Orchestrator



To Log out of Aria Automation Orchestrator:

1. Click **hol admin** to open the organization menu.
2. Click **SIGN OUT**.



## Conclusion

[251]

With Aria Automation Orchestrator it is very easy to leverage existing scripts in other languages (PowerShell, Python & Node.js). A very simple wrapper can be implemented around almost any existing scripts and imported into Aria Automation Orchestrator. Stitching

existing scripts together, even across different languages, has become easier than ever.

## You've finished the Module

[252]

Congratulations on completing the lab module.

If you are looking for additional information on running Python, Node.js and PowerShell with Aria Automation Orchestrator:

- [Core Concepts](#)
- [Step-by-step documentation](#)
- [Sample scripts](#)

From here you can:

1. Continue with the next lab module
2. Click [vlp:table-of-contents] Show Table of Contents] to jump to any module or lesson in this lab
3. End your lab and return in the future

## Module 5 - Integrate Aria Orchestrator and Aria Operations (45 minutes) Intermediate

### Introduction

[254]

Integrating Aria Orchestrator with Aria Operations to automate alert remediation through custom workflows is a powerful way to enhance operational efficiency and reliability in your vSphere environment. This integration allows you to streamline incident response by automatically executing predefined actions in response to alerts generated by Aria Operations. By setting up this integration, you can significantly reduce manual intervention, minimize downtime, and ensure that issues are addressed promptly and consistently.

This module will guide you through the process of setting up this integration, demonstrating how to configure and leverage custom workflows to handle alerts effectively, ultimately contributing to a more resilient and agile IT infrastructure.

While this module falls within the Aria Orchestrator Lab, much of the work will be in Aria Operations. This module covers the steps needed to integrate Orchestrator with Operations.

#### Lab Captain(s):

- Scott Bowe, Solutions Architect, United States

### Overview

[255]

In this module we will utilize an Aria Orchestrator workflow to dynamically resize a virtual machine based on its CPU workload. We will accomplish this by creating an Aria Operations alert for high and low CPU usage. We will configure Aria Operations to trigger a workflow to add more resources to a virtual machine when high CPU demand is detected, and remove CPU when low CPU demand is detected. At a high level we will accomplish this by executing the following steps:

- Modify an Aria Automation Template to support dynamic resizing and deploy a Virtual Machine
- Review Aria Orchestrator Workflows
- Configure Aria Operations
- Generate high CPU usage to trigger resizing

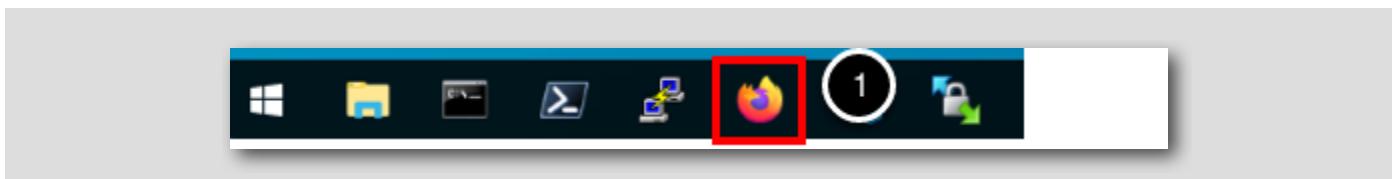
### Log in to Aria Automation

[256]

In the following few pages, we will log in to Aria Automation.

### Open the Firefox Browser from Windows Quick Launch Task Bar

[257]

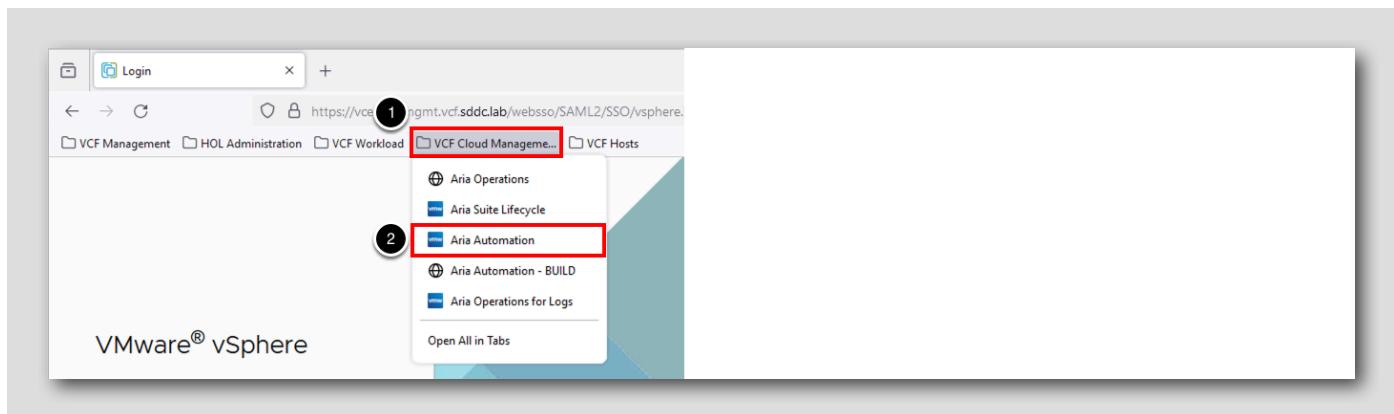


If the browser is not already open, launch Firefox.

1. Click the Firefox icon on the Windows Quick Launch Task Bar.

## Log in to Aria Automation

[258]

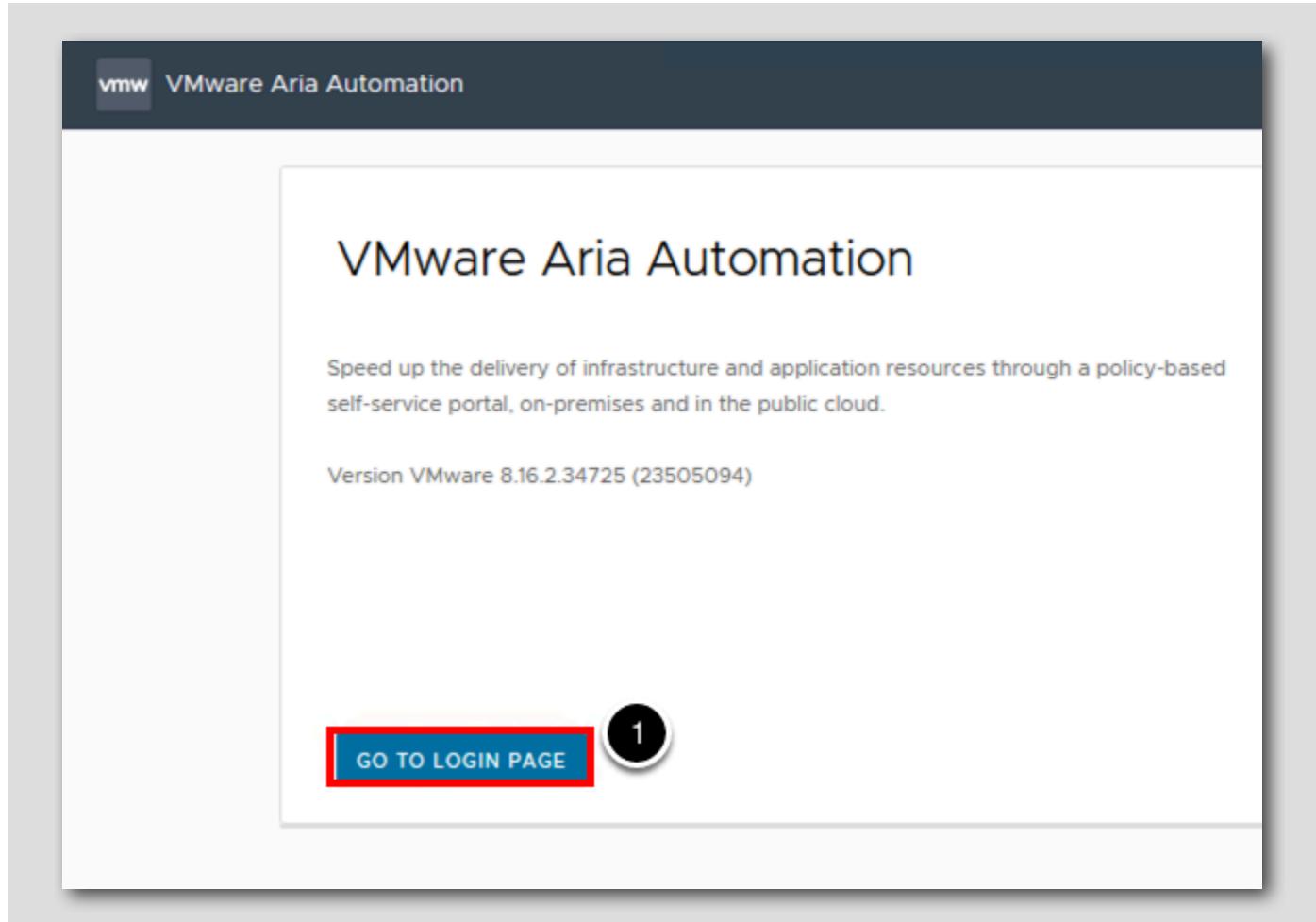


Once Firefox has loaded:

1. Click the VCF Cloud Management bookmark folder
2. Click Aria Automation.

Redirect to Workspace ONE Access for Sign-On

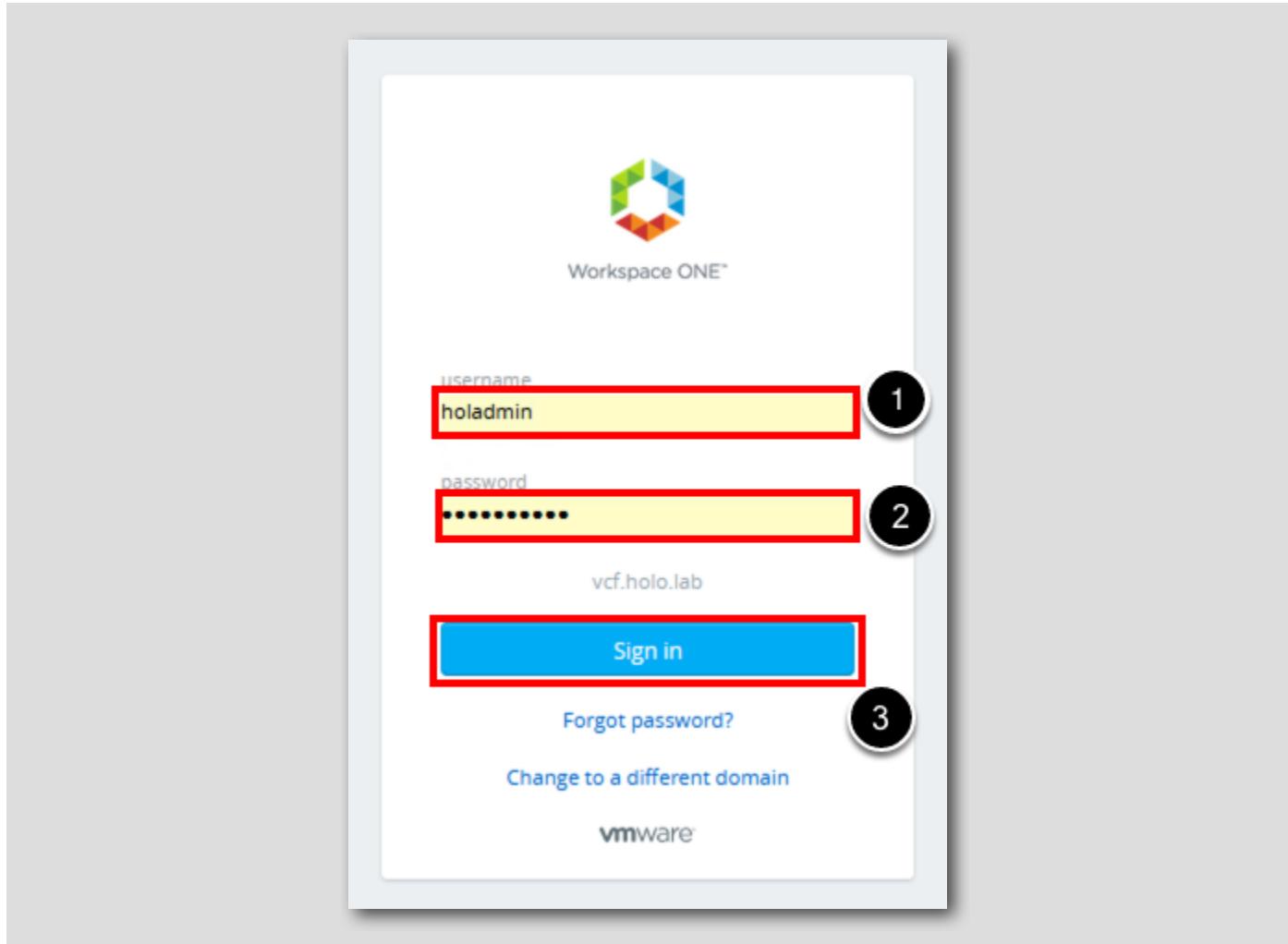
[259]



Aria Automation is integrated with Workspace ONE Access (aka VMware Identity Manager) and we need to redirect to the Workspace ONE Access login page to complete our log in progress.

1. At the VMware Aria Automation page, click GO TO LOGIN PAGE.

## Workspace ONE Access Login



Note: The credentials for **holadmin** should already be cached in the browser window.

At the **Workspace ONE Access** prompt, type in the following user and password information.

1. At the **username** field, type **holadmin**.
2. At the **password** field, type **VMware123!**.
3. Click **Sign in**.

## Deploy a virtual machine

As we discussed during the introduction, the integration between Aria Operations and Aria Orchestrator will update an existing Aria

Automation Deployment. To accomplish this we will need to modify a cloud template, and deploy a virtual machine for use later in the exercise. The modifications we are making will allow Aria Operations to scope it's automated response to only systems that we want targeted and to allow the resource changes to be managed by Aria Automation.

## Navigate to Assembler

The screenshot shows the VMware Aria Operations interface. At the top, there is a "Quickstart" section for "VMware Aria Automation". It includes a brief description, a time-to-complete estimate of "Approx. 10 minutes", and a "LAUNCH QUICKSTART" button. To the right of this is an illustration of two people working on a large server or storage unit with network and database icons. Below this is a "My Services" section with four tiles: "Assembler" (highlighted with a red box and a circled "1"), "Orchestrator", "Pipelines", and "Service Broker". A vertical "SUPPORT" button is on the far right.

1. Launch Assembler by clicking on the Assembler tile.

## Duplicate Cloud Template

[263]

The screenshot shows the VMware Aria Automation interface. The top navigation bar includes tabs for Assembler, Infrastructure, Extensibility, Tenant Management, and Alerts. The 'Assembler' tab is selected, indicated by a red box labeled '1'. The main content area is titled 'Templates' and shows a list of five items. The second item in the list, 'Ubuntu 22', has its checkbox checked, indicated by a red box labeled '2'. At the top of the list, there is a toolbar with buttons for 'NEW FROM...', 'SYNC', 'CLONE' (which is highlighted with a red box labeled '3'), 'DEPLOY', 'DOWNLOAD', and 'DELETE'. The 'CLONE' button is the third button from the left.

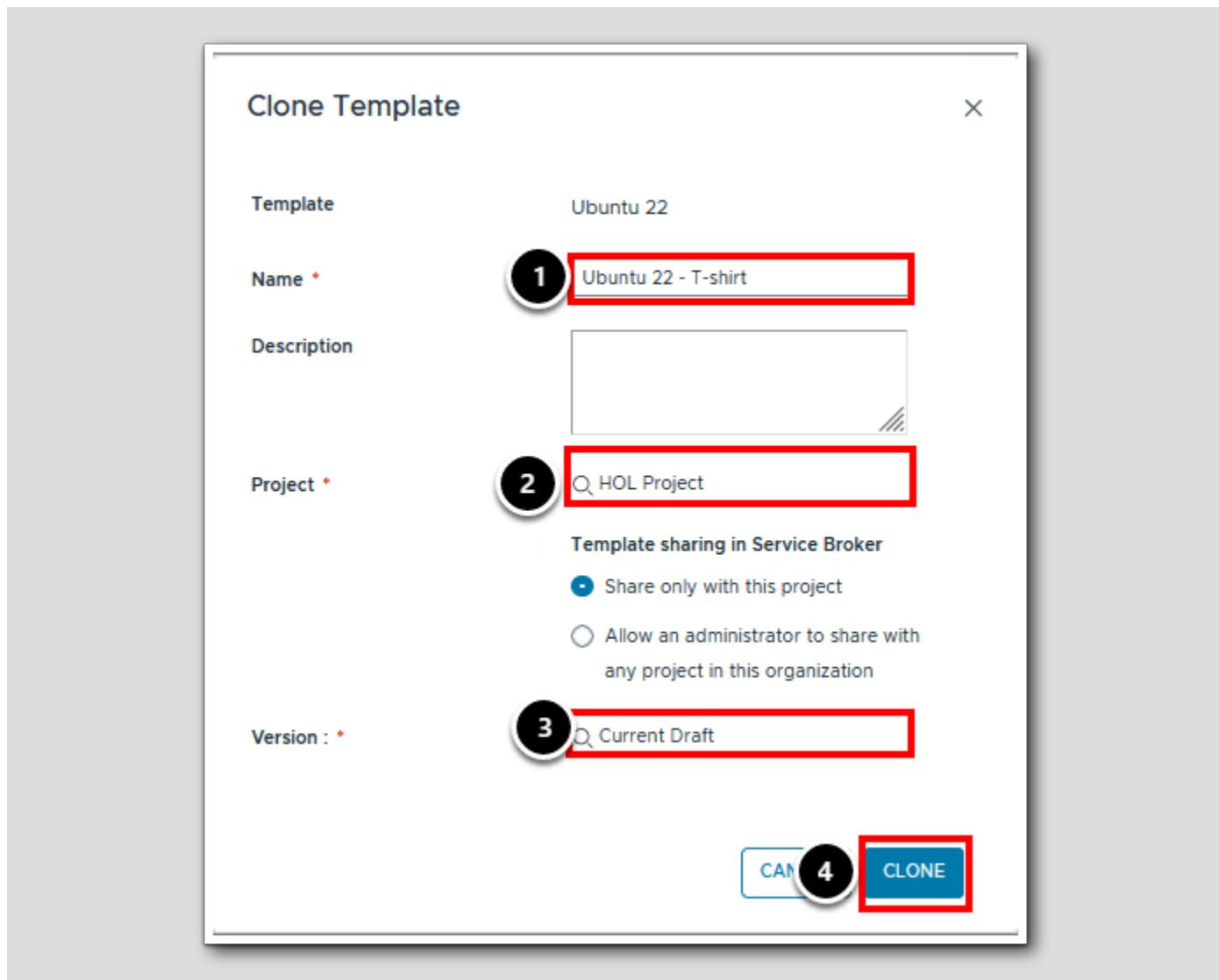
|                                     | Name                            | Source Control | Read-only | Project     | Last Updated              | Updated By           | Released Versions |
|-------------------------------------|---------------------------------|----------------|-----------|-------------|---------------------------|----------------------|-------------------|
| <input type="checkbox"/>            | vSphere VM with Comm...         | ...            |           | HOL Proj... | Jul 23, 2024, 4:12:06 ... | holadmin@vcf.holo... | 0 out of 1        |
| <input type="checkbox"/>            | <u>Windows with cloudbas...</u> | ...            |           | HOL Proj... | Jul 23, 2024, 4:11:38 ... | holadmin@vcf.holo... | 1 out of 3        |
| <input type="checkbox"/>            | <u>vSphere Machine</u>          | ...            |           | HOL Proj... | Jul 20, 2024, 1:25:01 ... | system-user          | 1 out of 1        |
| <input checked="" type="checkbox"/> | <b>Ubuntu 22</b>                | ...            |           | HOL Proj... | Jul 20, 2024, 1:25:01 ... | system-user          | 1 out of 1        |
| <input type="checkbox"/>            | Cloud VM with Form              | ...            |           | HOL Proj... | Jul 20, 2024, 1:25:01 ... | system-user          | 1 out of 1        |

Next we will duplicate an existing cloud template to add an additional input and declare vSphere tags on the virtual machine.

1. Click Design button
2. Select the template Ubuntu 22 by check the checkbox next to it
3. Click the Clone button

## Configure Duplicate Options

[264]



1. Enter Ubuntu 22 - T-shirt as the name for the cloned copy of the template
2. Select HOL Project
3. Select Current Draft as the version to copy
4. Click CLONE

## Open Cloud Template

|                          | Name                       | Source Control | Read-only | Project     | Last Updated              | Updated By           | Released Versions |
|--------------------------|----------------------------|----------------|-----------|-------------|---------------------------|----------------------|-------------------|
| <input type="checkbox"/> | » Ubuntu 22 - T-shirt      |                |           | HOL Proj... | Jul 27, 2024, 8:06:34...  | holadmin@vcf.holo... | 0 out of 0        |
| <input type="checkbox"/> | » vSphere VM with Comm...  | 🔥              |           | HOL Proj... | Jul 23, 2024, 4:12:06 ... | holadmin@vcf.holo... | 0 out of 1        |
| <input type="checkbox"/> | » Windows with cloudbas... | 🔥              |           | HOL Proj... | Jul 23, 2024, 4:11:38 ... | holadmin@vcf.holo... | 1 out of 3        |
| <input type="checkbox"/> | » vSphere Machine          | 🔥              |           | HOL Proj... | Jul 20, 2024, 1:25:01 ... | system-user          | 1 out of 1        |
| <input type="checkbox"/> | » Ubuntu 22                | 🔥              |           | HOL Proj... | Jul 20, 2024, 1:25:01 ... | system-user          | 1 out of 1        |
| <input type="checkbox"/> | » Cloud VM with Form       | 🔥              |           | HOL Proj... | Jul 20, 2024, 1:25:01 ... | system-user          | 1 out of 1        |

1. Open the cloned cloud template by selecting Ubuntu 22 - T-shirt

## Add an Input to the Cloud Template

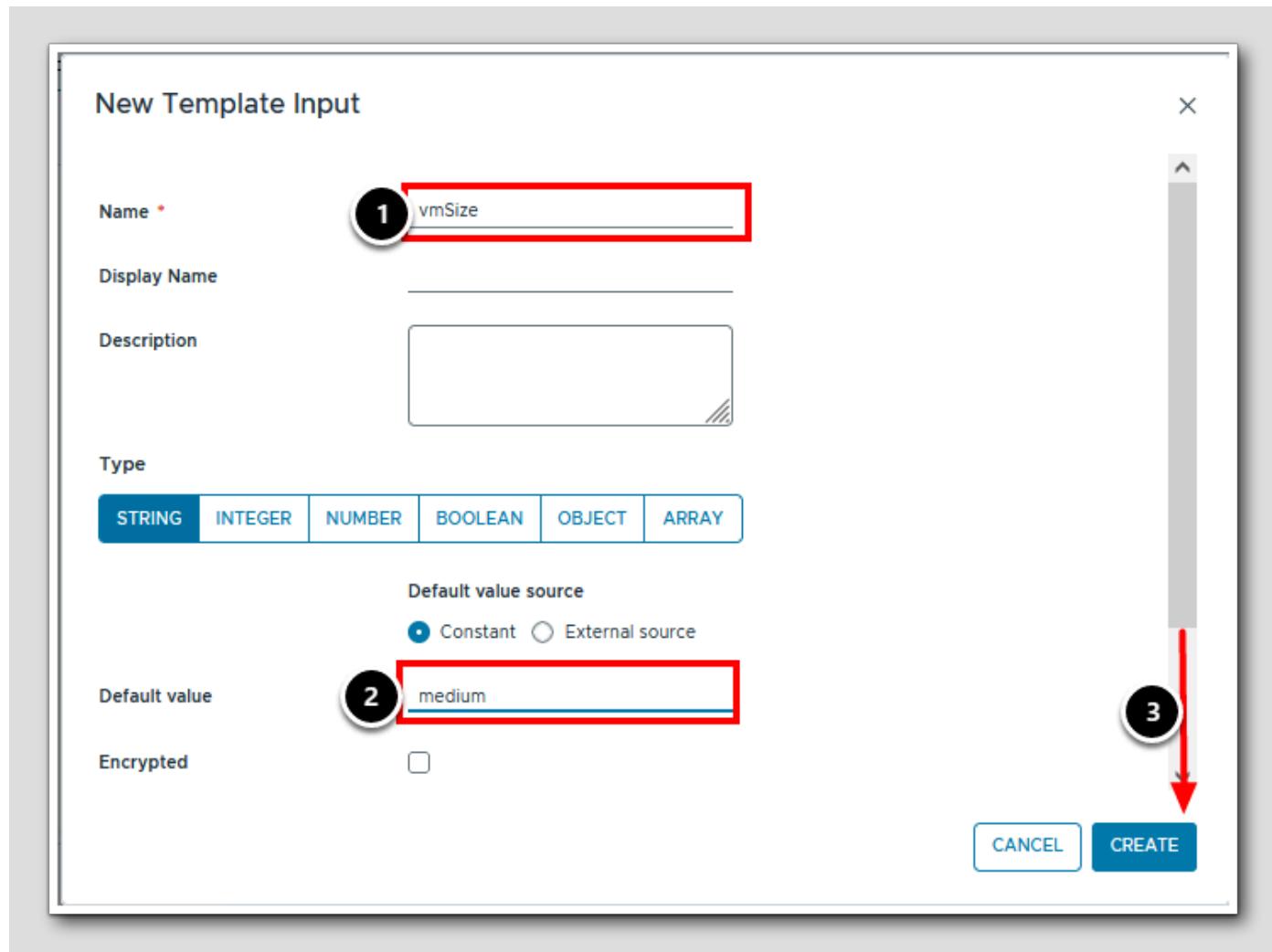
You can add a new input to a cloud template by directly editing the YAML, or using the GUI to help guide you. We will use the GUI this time, to help simplify the process.

The screenshot shows a modal window titled 'Template Inputs' with a status of '0 items'. At the top right, there is a tab labeled 'Inputs' which is highlighted with a red box and circled with a number '1'. Below the tabs, there is a button labeled '+ NEW TEMPLATE INPUT' which is also highlighted with a red box and circled with a number '2'. The main area of the window displays the message 'No Template Input Pro' followed by a 'Manage Columns' button.

1. Navigate to the Inputs tab
2. Click the + NEW TEMPLATE INPUT button

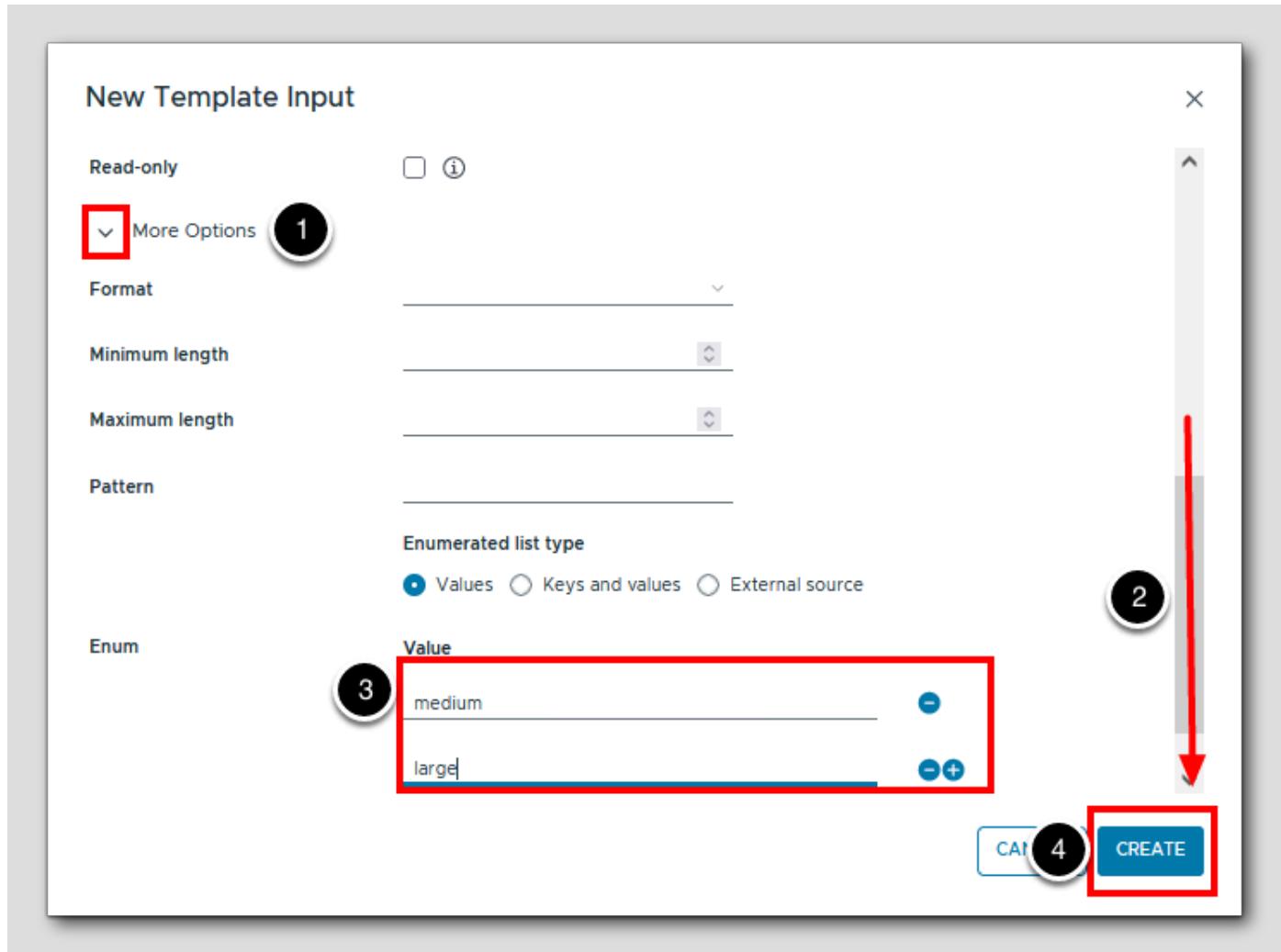
## Configure Input

[267]



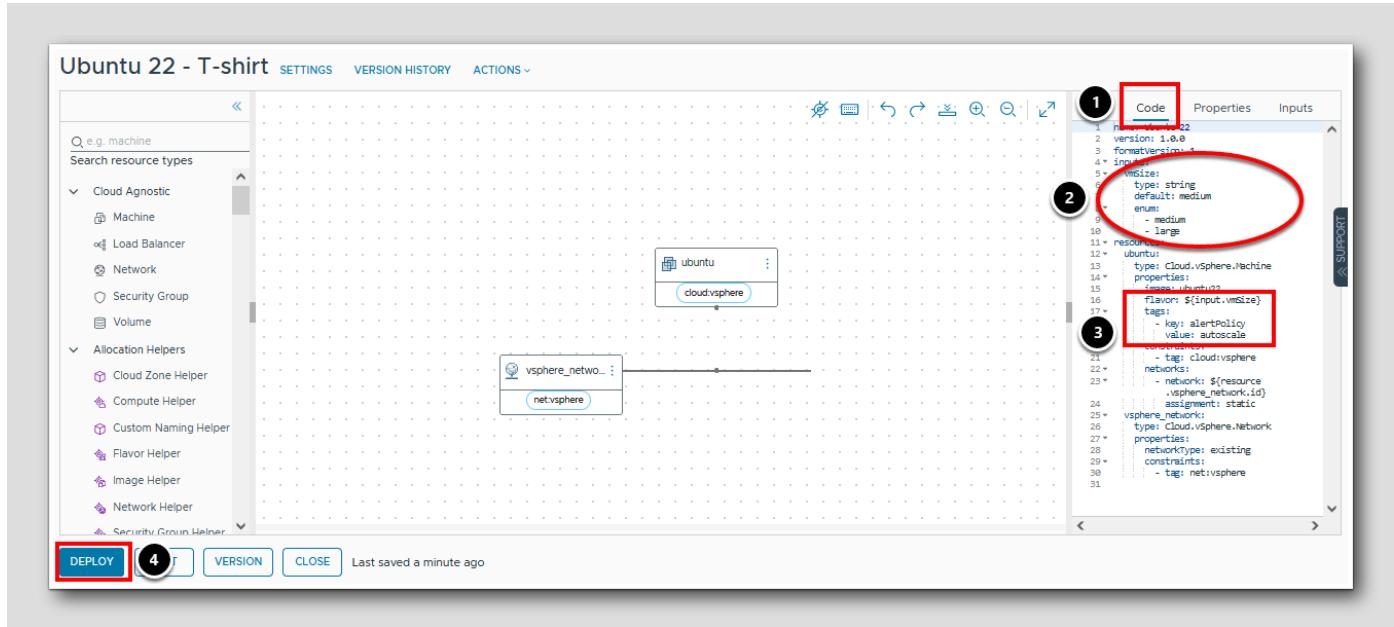
1. Enter **vmSize** in the name field
2. Provide **medium** as the Default value
3. If needed, scroll down

## Configure Input continued



1. Expand the more options by clicking the downward carrot
2. If needed, scroll down
3. Enter **medium** and **large** as values for Enum. You can add a new line by clicking on the + button to the right of the first line
4. Click **CREATE**

## Modify Template



1. Navigate back to the **Code** tab
2. Notice how our actions to add an input updated the code of our template
3. Delete the existing **flavor** property on line 16, and replace it with the following code:

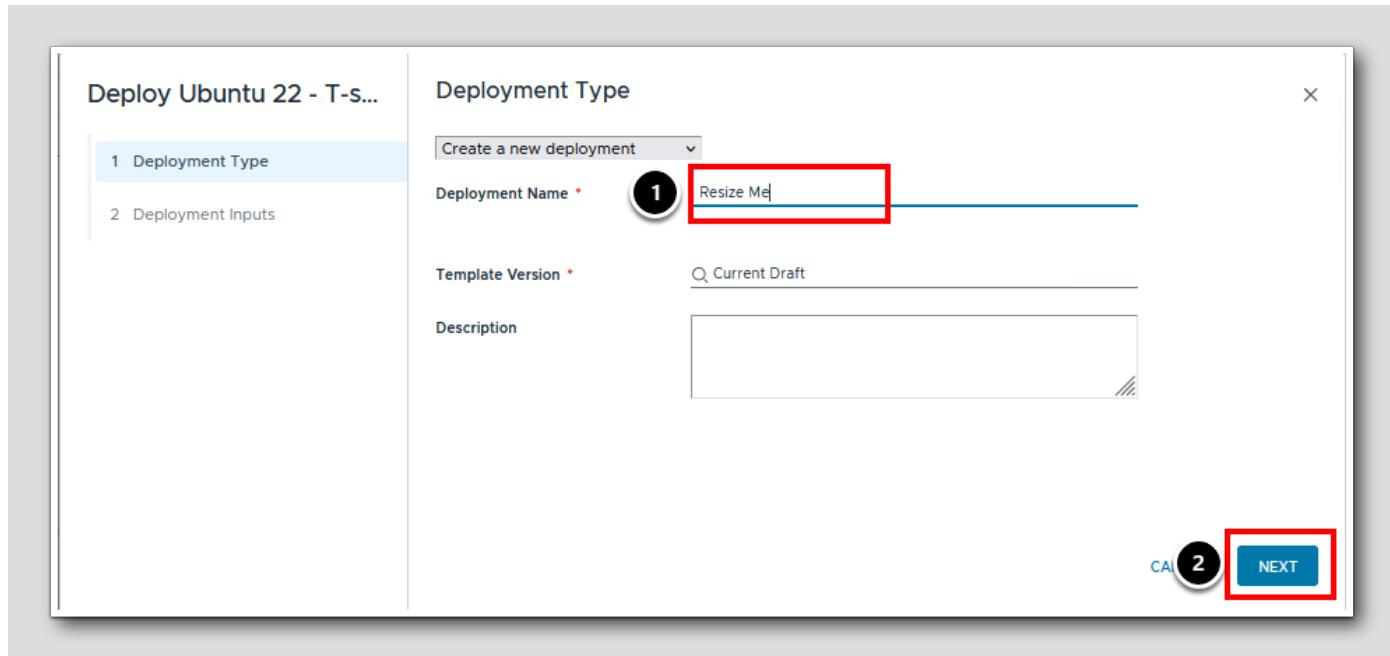
```

flavor: ${input.vmSize}
tags:
  - key: alertPolicy
    value: autoscale
  
```

The YAML used within the Cloud Template is sensitive to indentation. Ensure that the YAML code matches the screenshot and code snippet above, and that no error banner is present at the top of the screen before continuing.

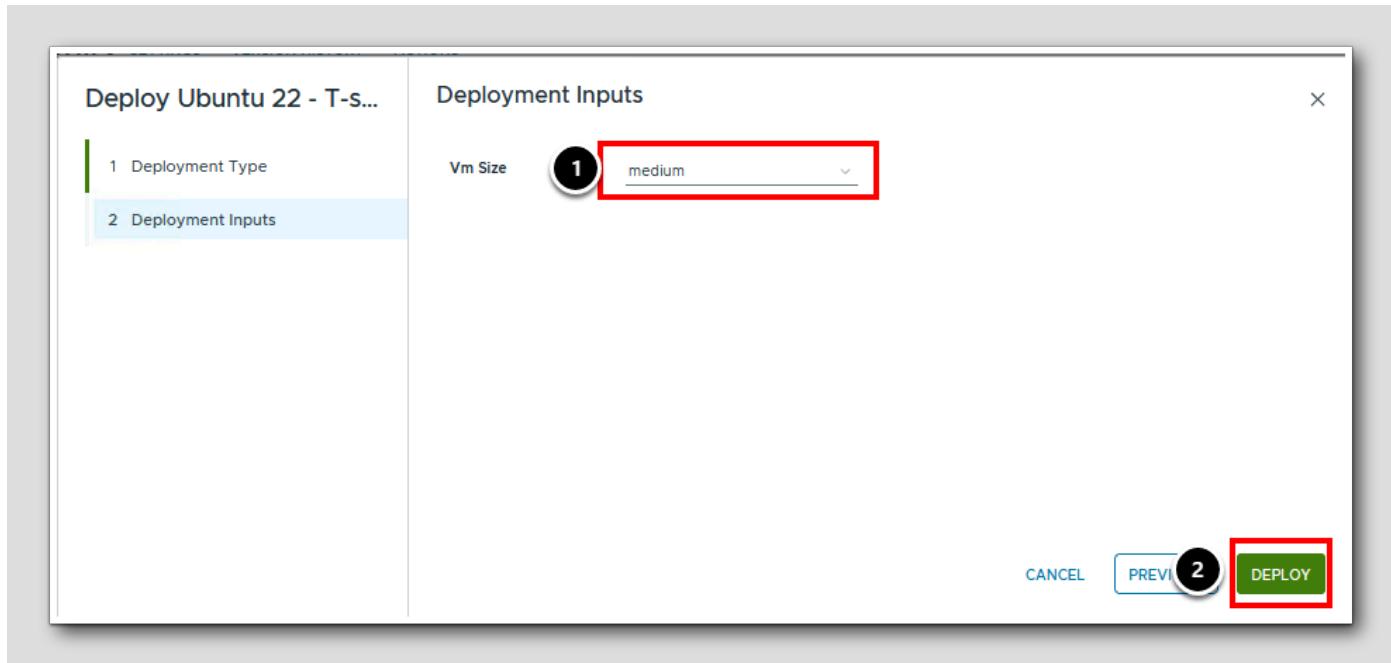
1. Start a deployment by clicking the **DEPLOY** button

## Deploy VM



1. Enter a name for the deployment as **Resize Me**
2. Click **NEXT**

## Deploy VM continued



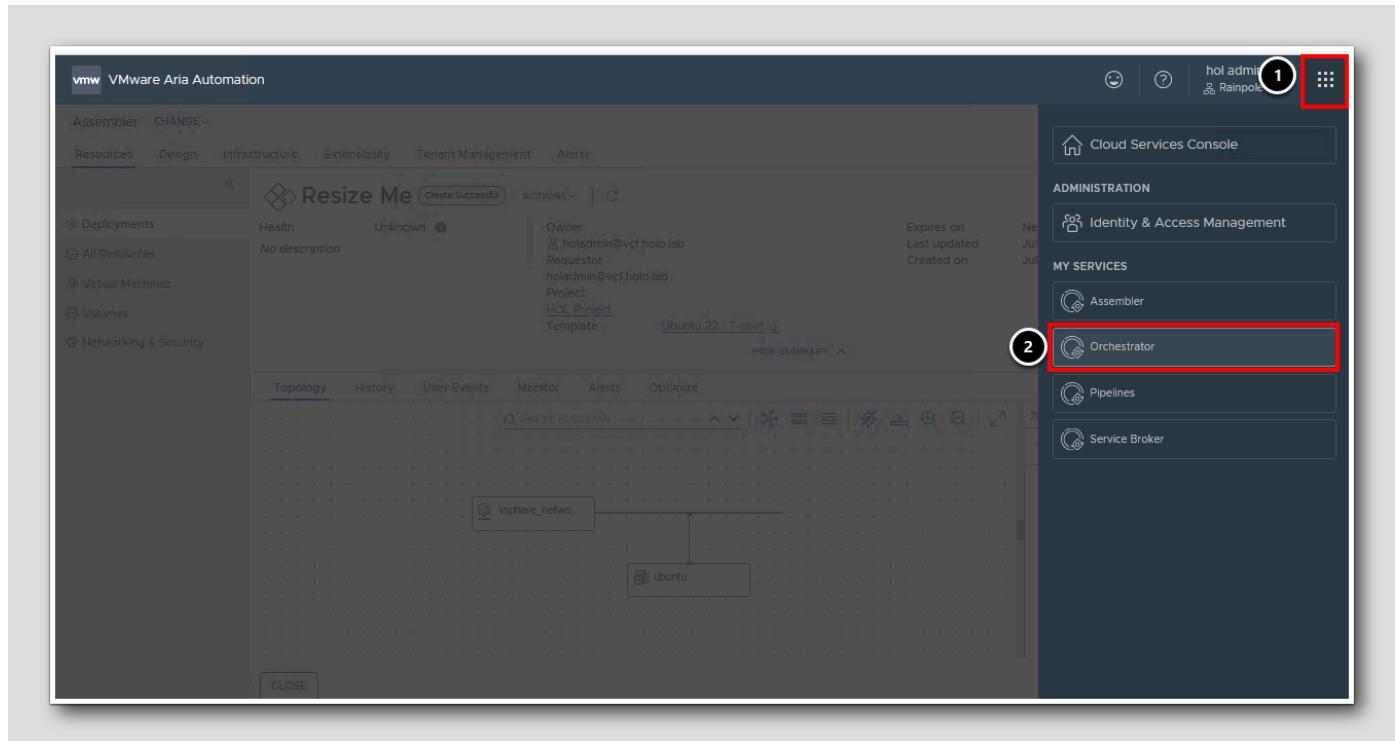
1. Choose **medium** for for Vm Size.
2. Initiate the deployment by clicking **DEPLOY**

In this lesson we accomplished the steps needed to deploy a virtual machine using a cloud template that supports multiple t-shirt sizes. The cloud template also tags the virtual machine so that Aria Operations can apply a specialized monitoring policy, limiting the scope of it's automatic remediation.

## Review Aria Orchestrator Workflow

While we wait for our deployment to complete we will review the Aria Orchestrator workflows used to scale the deployment up or down. There are two workflows: one is used in response to the low CPU usage alert to scale the virtual machine down and the other is used in response to the high CPU usage alert to scale the virtual machine up. We will review the **AutoScale-in** workflow as the **AutoScale-out** workflow is functionally identical except for a switch case that determines the new size of the deployment. It is entirely possible to perform this action with a single workflow, but it would increase the complexity of the integration beyond the scope of this lab.

## Navigate to Aria Orchestrator



1. Expand the Cloud Services Menu
2. Select Orchestrator

## Open Workflow

[274]

The screenshot shows the VMware Aria Automation Orchestrator interface. The left sidebar has 'Workflows' selected. The main area shows a search bar with 'Any : autoscale' and two workflow cards: 'AutoScale-In' and 'AutoScale-Out'. The 'AutoScale-In' card has an 'OPEN' button highlighted.

1. Under Library, select **Workflows**
2. Filter the list of workflows by entering **autoscale** and pressing enter
3. Open the AutoScale-in workflow by clicking on **OPEN**

## Review Variables

[275]

The screenshot shows the 'Variables' tab for the 'AutoScale-In' workflow. It lists two variables: 'AriaAutoUrl' (string, value: https://auto.vcf.sddc.lab) and 'vmname' (string, value: ).

| Name        | Type   | Description | References                           |
|-------------|--------|-------------|--------------------------------------|
| AriaAutoUrl | string |             | AutoScale_In_AriaAuto                |
| vmname      | string |             | AutoScale_In_AriaAuto.Get Input Info |

- Click on **Variables** to change tabs

In Orchestrator variables are values available internally to the workflow. This workflow has two defined variables:

- **AriaAutoUrl**
  - This variable provides a reference to the Aria Automation instance. The workflow will use this during execution to locate the Aria Automation API endpoint. By defining this as a variable rather than in the code of a later scripting step, the workflow is more easily ported to a new Automation and Orchestrator instance
- **vmname**
  - This variable does not have a default value and is dynamically derived during execution.

## Review Input/Outputs

[276]

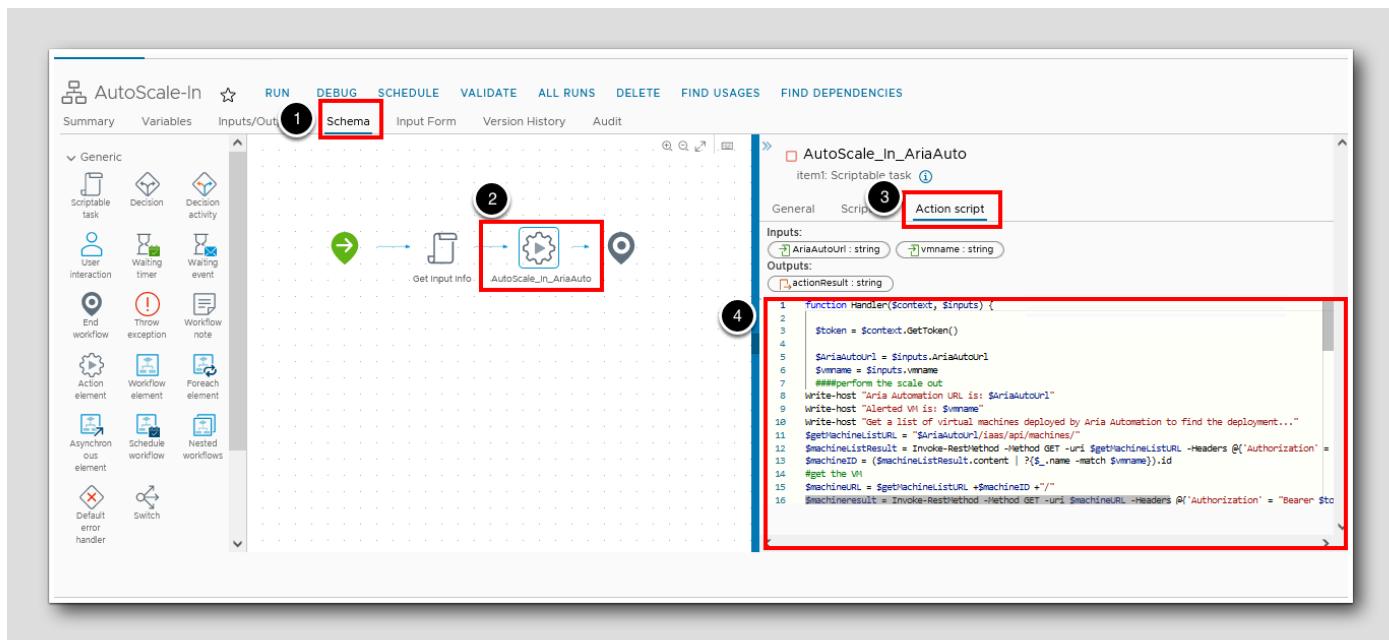
| Name         | Type              | Direction | Description | References                            |
|--------------|-------------------|-----------|-------------|---------------------------------------|
| vm           | VC:VirtualMachine | Input     |             | <a href="#">Get Input Info</a>        |
| actionResult | string            | Output    |             | <a href="#">AutoScale_In_AriaAuto</a> |

- Click on **Inputs/Outputs** to change tabs

Inputs/Outputs are similar to variables, except they are either provided (inputs) by the calling entity, or accessible (outputs) to the calling entity. In this case the calling entity is Aria Operations, which does not check for a result, but it is useful to have in case we wanted to call this work flow as part of a larger workflow or via API.

- **vm**
  - This is a vCenter VM object passed from Aria Operations. We will see how we configure the workflow inputs in Aria Operations and associate it with the virtual machine object type in a later lesson.
- **actionResult**
  - This is the result of the workflow and any return codes it may offer.

## Review Schema



1. Click on **Schema** to change to the schema tab
2. Select **AutoScale\_in\_AriaAuto** to select the action
3. Change to the **Action Script** tab
4. Scroll through and review the code in the script.

The script queries the Aria Automation API to determine which deployment the affected virtual machine belongs to. Once that is determined it queries the target deployment, and depending on the current flavor size submits an update to a larger (or smaller size).

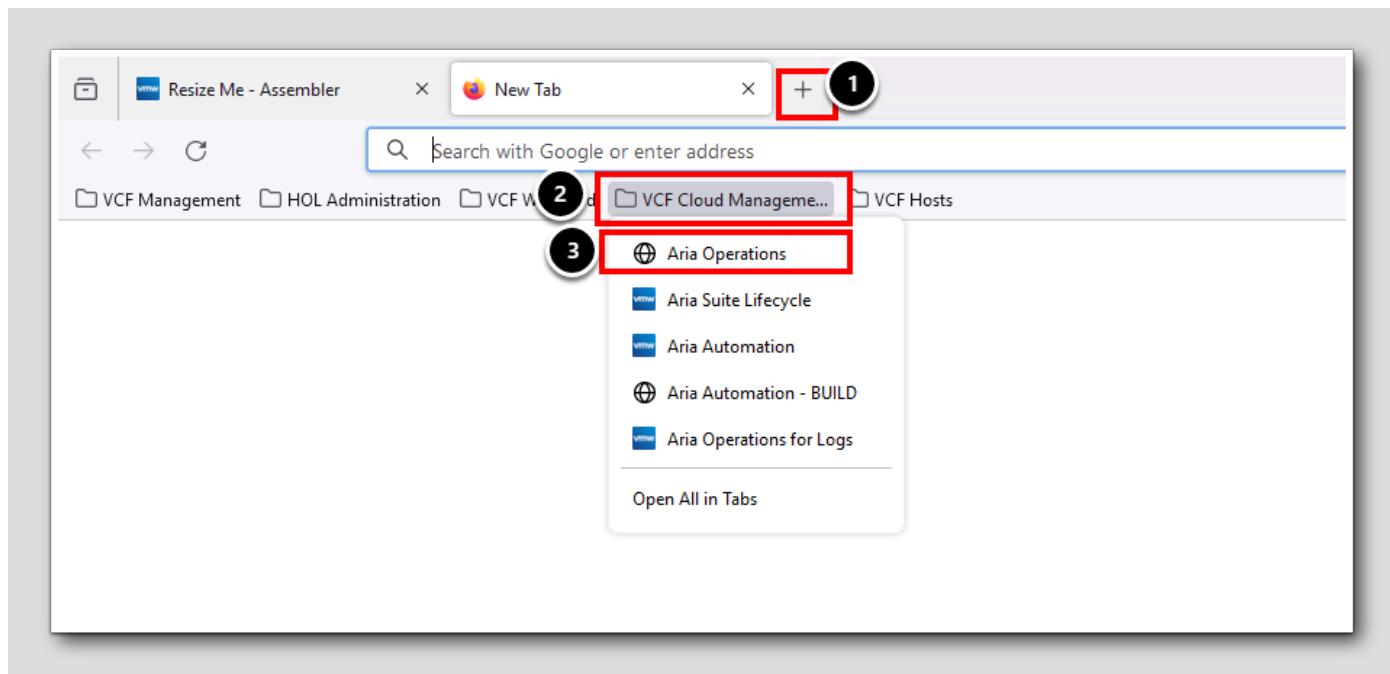
In this section we reviewed the workflows that will execute the updates to remediate the alert triggered by Aria Automation.

## Configure Aria Operations for Orchestrator Management Pack

Aria Operations leverages Actions to facilitate auto-remediation of alerts. Most management packs, such as the one for vCenter, include actions to allow execution of tasks such as power-on, power-off, or snapshot cleanup, via a push button remediation, or automated through policies. We can supplement these built-in actions, with Aria Orchestrator workflows. This gives us the power to create complex flows to perform tasks. To make Orchestrator workflows available to Operations, we must use the Aria Operations for Orchestrator Management Pack. This management pack allows us to import workflows for use as actions. Once imported, we must configure the workflows by associating them with an object type and input.

Let's get started!

## Launch Aria Operations



1. Open a new tab by clicking the + side in the browser.
2. Open the VCF Cloud Management shortcut folder.
3. Launch Aria Operations

Log in to Operations

[280]



We will utilize our existing vIDM session to log into Aria Operations.

1. Select **Workspace ONE Access** in the authentication type pulldown
2. Click **REDIRECT**

After logging in, we will configure package discovery to import our workflows.

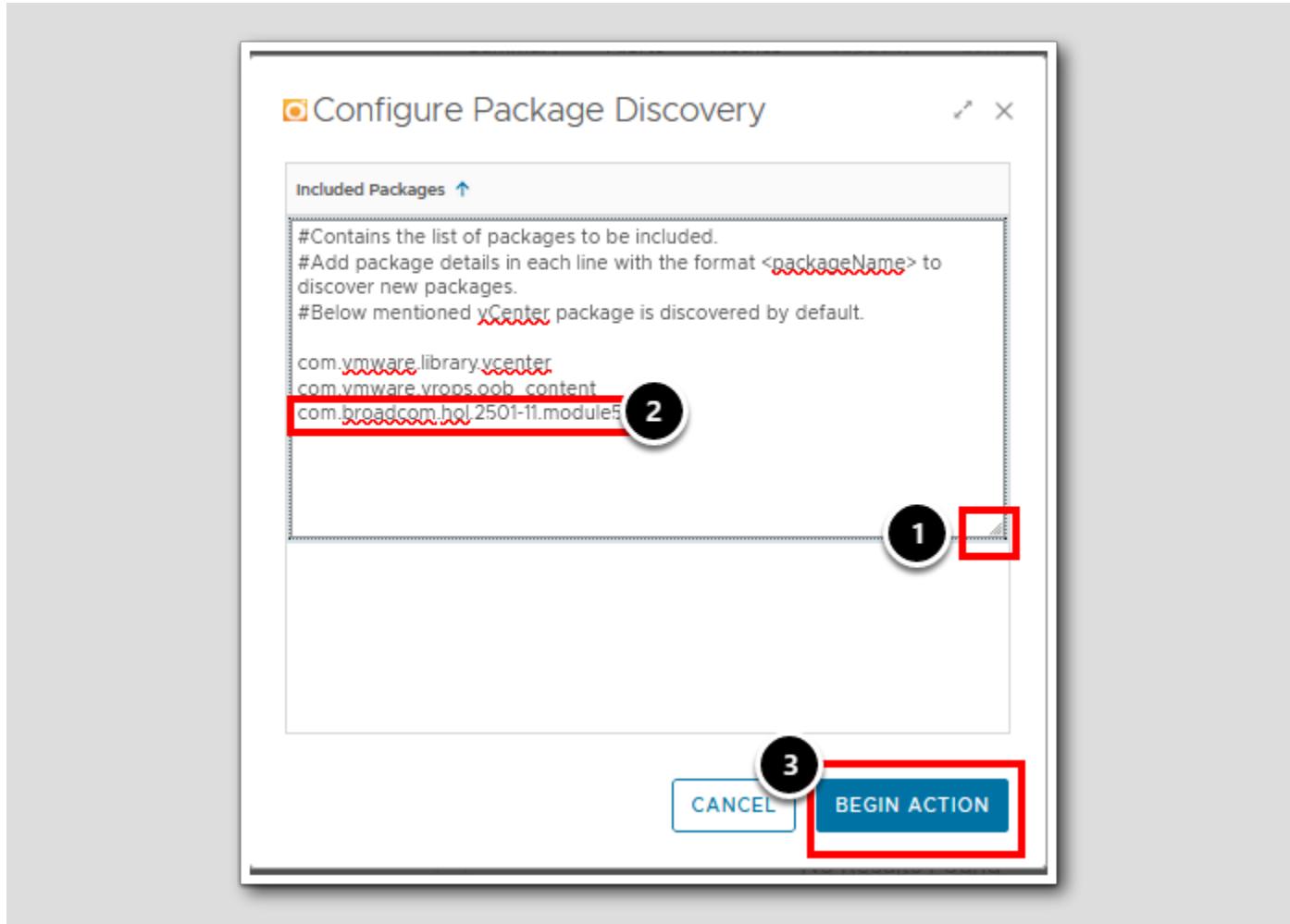
## Configure Package Discovery

The screenshot shows the VMware Aria Operations interface. The left sidebar has 'Environment' selected (step 1). The 'Object Browser' pane (step 2) is open, showing 'VMware Aria Automation Orchestrator' expanded (step 3), then 'Orchestrator Workflows' (step 4), and finally 'HOL Orchestrator' (step 5). The main pane shows 'HOL Orchestrator' details (step 6), and the 'Actions' menu is open with 'Configure Package Discovery' highlighted (step 7).

Packages are utilized by Aria Orchestrator to group workflows, actions, and other constructs into an easily exportable container. Most often this is used to share content between Orchestrator instances and identify dependencies between constructs such as workflows and actions. Conveniently, packages provide an easy container for us to identify and limit which workflows are available as actions in Aria Operations.

1. Expand Environment in the navigation pane.
2. Select Object Browser
3. In the Object Browser pane that opens, expand VMware Aria Automation Orchestrator
4. Open Orchestrator Workflows
5. Open the orchestrator instance labeled HOL Orchestrator
6. Click on the Actions menu
7. Select Configure Package Discovery

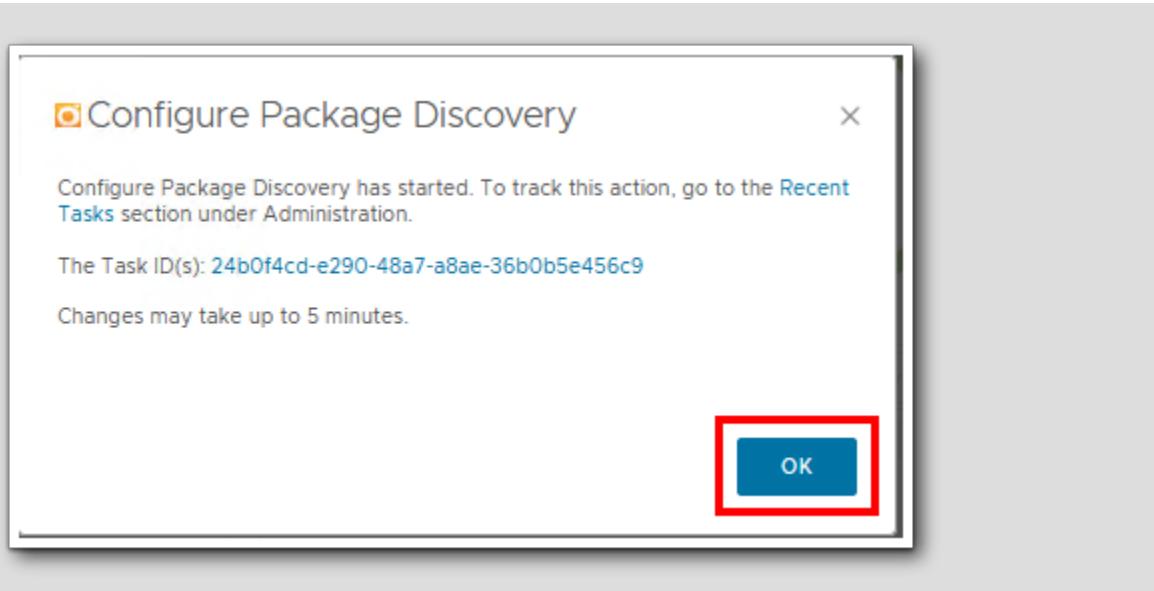
## Configure Package Discovery continued



The Configure Package Discovery window allows us to configure which packages Operations should import from Orchestrator. Notice that `com.vmware.library.vcenter` and `com.vmware.vrops.oob_content` are included.

1. Expand the included packages input box, by dragging the expansion corner to the desired size.
2. Add `com.broadcom.hol.2501.11.module5` on a new line, below the two existing entries
3. Click BEGIN ACTION

## Confirmation



1. Click OK

It may take up to 5 minutes for Operations to import the workflows from Orchestrator, it depends on the current workload of the operations cluster.

## Interlude

[284]

The screenshot shows the vSphere Web Client interface for a virtual machine named "ubuntu-000308". The "Summary" tab is selected. On the left, there's a summary box with the following details:

- IP Address: 10.64.12
- Number of virtual CPUs: 1
- Memory: 1GB
- Disk Space: 25 GB
- VMware tools: Tools Version

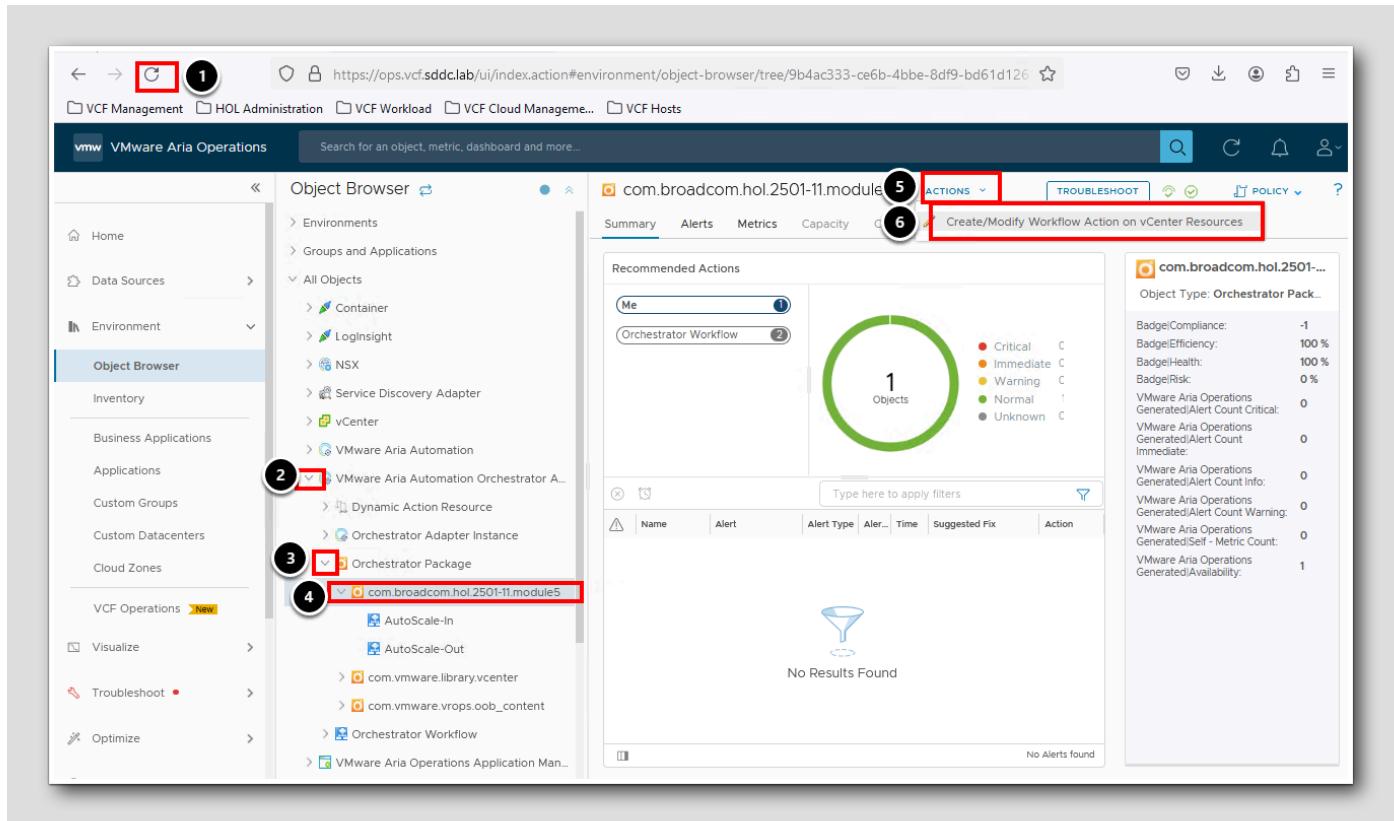
A message box indicates "Time Remaining > 1 Year". Below it, under "Utilization", there are sections for CPU Usage, Free Memory, Guest Page In Rate per second, and Page Out Rate per second.

The "ACTIONS" menu is open, listing various operations:

- Add a new vNic to Virtual Machine
- Delete Unused Snapshots for VM
- Execute Script
- Get Top Processes
- Manage Snapshots for VM
- Migrate VM
- Migrate Virtual Machine with vMotion
- Move VM
- Power Off VM
- Reboot Guest OS For VM
- Reconfigure VM
- Set CPU Count and Memory for VM
- Set CPU Count for VM
- Set CPU Resources for VM
- Set Memory Resources for VM
- Set Memory for VM
- Shut Down Guest OS for VM
- Suspend VM
- Update CPU Reservation
- Update CPU Reservation Limit
- Update Memory Reservation
- Upgrade Guest Tools for VM
- VM Power Operation
- Open Virtual Machine in vSphere Web Client...

Our next step is to associate the imported actions with a **Resource Type** and **Target Resource Type**. Resource Type indicates to operations, which objects should this action be available on, while target resource instructs Operations what type of object should be passed as an input to the workflow. Generally, these should be the same object type, however there are scenarios where you may want to create a workflow that performs an action on all virtual machines running on a given vSphere Host. In that case you would select Host System as the Resource Type, and Virtual Machine as the Target Resource Type.

## Configure Workflow Inputs

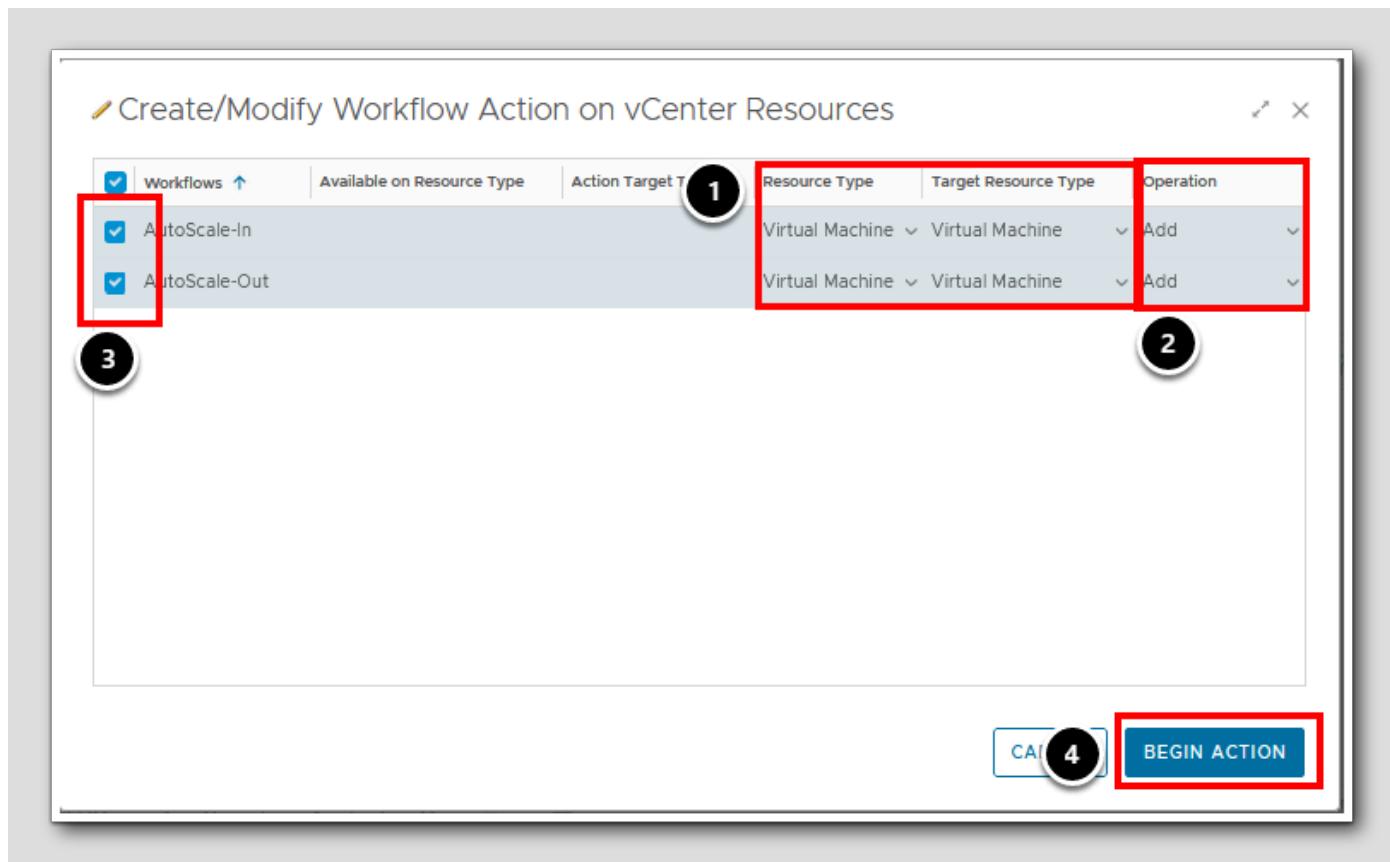


As mentioned previously, depending on the load on the Operations cluster, the workflow input may take up to 5 minutes.

1. Click the browser refresh button to reload the page
2. If needed, expand **VMware Aria Automation Orchestrator Adapter**
3. Expand **Orchestrator Package**
4. Select **com.broadcom.hol.2501-11.module5**
5. Open the **ACTIONS** context menu
6. Select **Create/Modify Workflow Action on vCenter Resources**

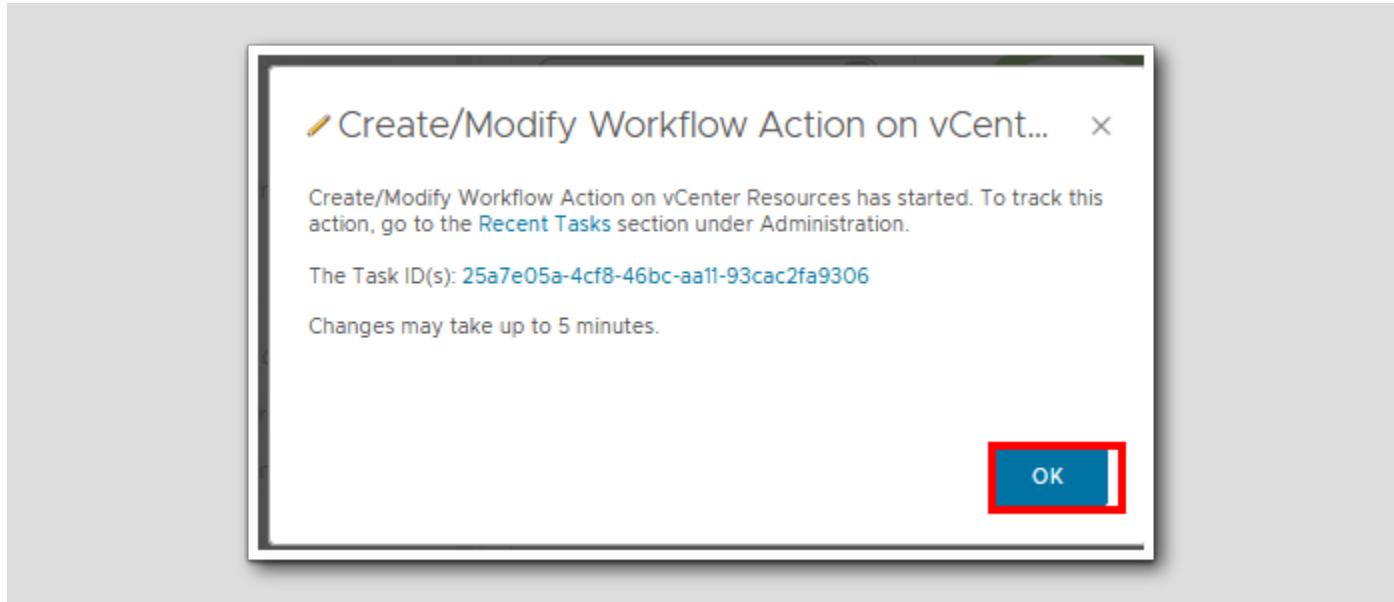
Next we will assign the target types for our workflows.

## Create/Modify Workflow Action



1. Configure Resource Type and Target Resource Type for both workflows to Virtual Machine
2. Set Operation to Add for both workflows, this instructs operations to add the association to the target objects.
3. Select both workflows
4. Click BEGIN ACTION

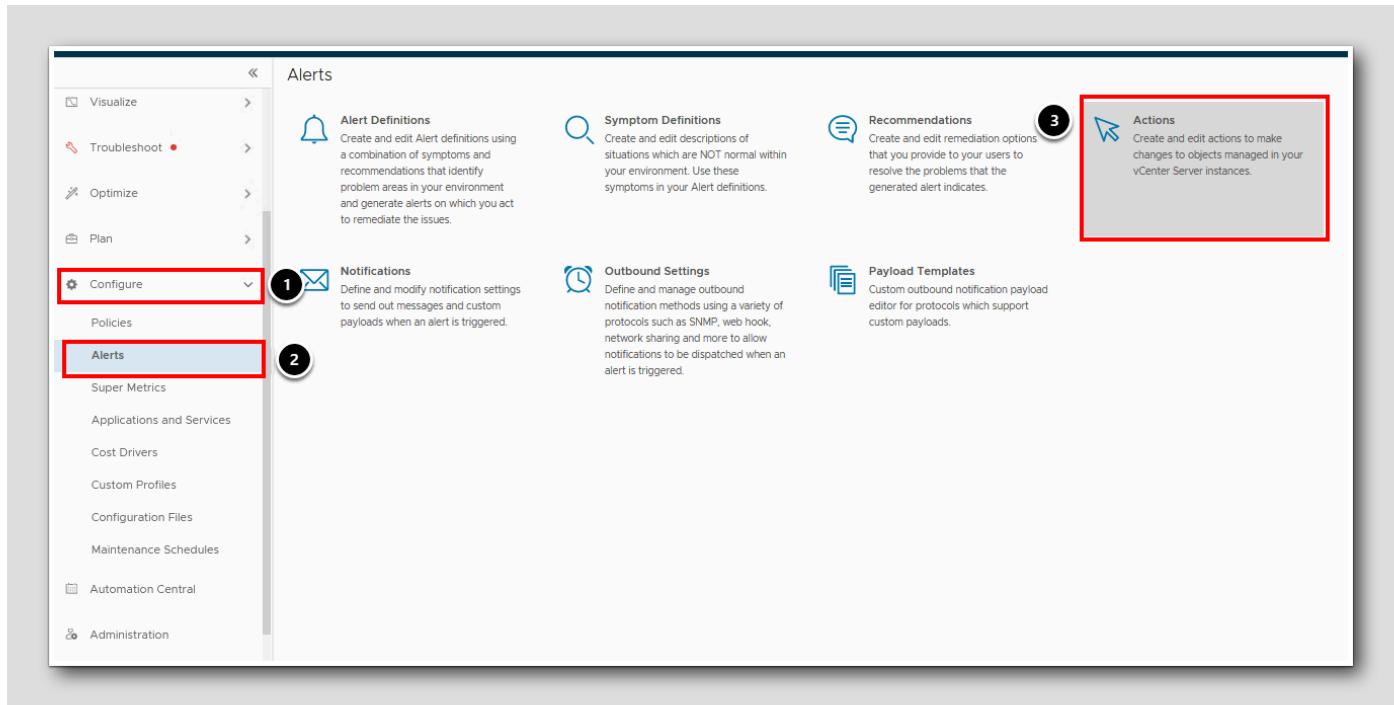
## Confirmation



1. Acknowledge the confirmation prompt by clicking OK.

Finally, let's navigate to the actions section and validate that the actions are available to Aria Operations.

## Navigate to Alert Configuration



1. Click **Configure** to expand the configuration section of the navigation pane
2. Select **Alerts** to open the alerts menu
3. Click **Actions**

## Validate Actions

[289]

The screenshot shows the 'Actions' page in Aria Operations. At the top, there is a breadcrumb navigation: Home / Alerts / Actions. Below the header is a search bar with the text 'autoscale' typed into it. A red box highlights this search bar. To the left of the search bar is a circular icon containing the number '1'. To the right is a 'Clear' button ('X') and a dropdown arrow. The main area is a table with the following columns: Action Name, Action Type, Adapter Type, Resource Adapter Type, Associated Object Types, and Recommendations. There are two rows of data:

| Action Name   | Action Type | Adapter Type                             | Resource Adapter Type | Associated Object Types | Recommendations |
|---------------|-------------|--|-----------------------|-------------------------|-----------------|
| AutoScale-In  | update      | VMware Aria Automation Orchestrator A... | vCenter               | Virtual Machine         | 0               |
| AutoScale-Out | update      | VMware Aria Automation Orchestrator A... | vCenter               | Virtual Machine         | 0               |

At the bottom of the table, it says '1 - 2 of 2 items'.

1. Type **autoscale** in the filter box
2. Press Enter (not shown)

Validate that the actions, AutoScale-In and AutoScale-Out appear with the correct type associations.

If the actions are not present immediately press Enter again after a few minutes to refresh the page. It may take up to five minutes for the actions to appear.

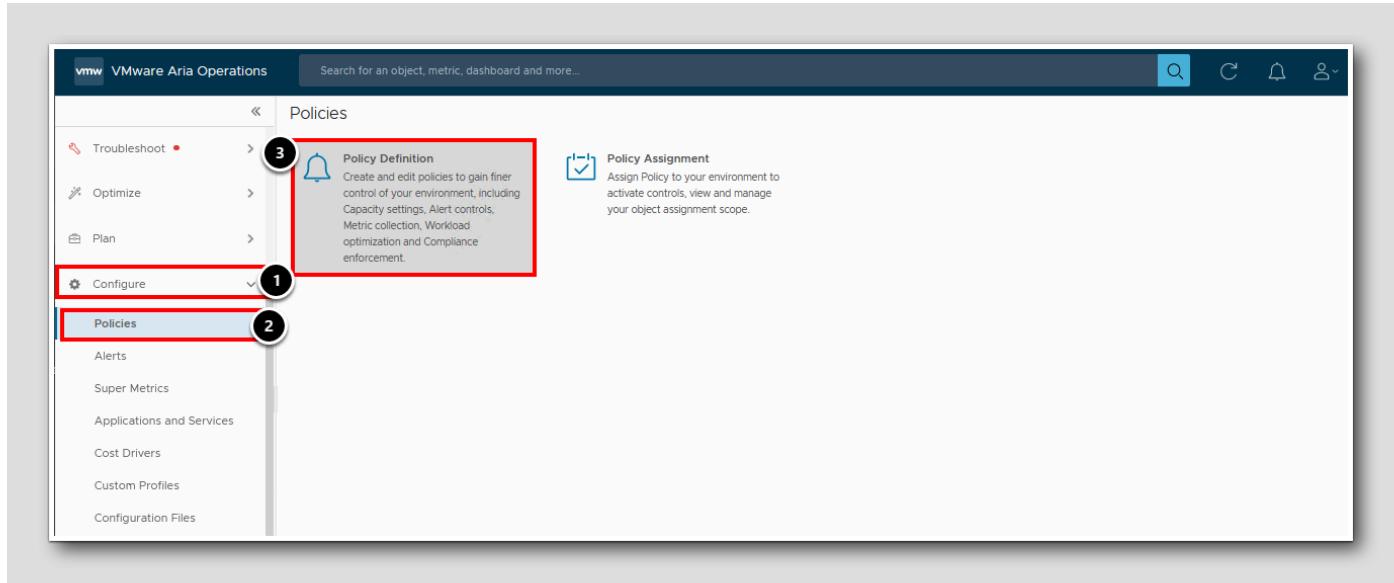
Congratulations Aria Operations is now configured to import actions from Aria Orchestrator for use as part of automated remediation. The final step in the configuration of the integration between Orchestrator and Operations is to configure alerts and monitoring policy to utilize the actions and automatically remediate issues that arise.

## Configure Aria Operations Monitoring Policy

[290]

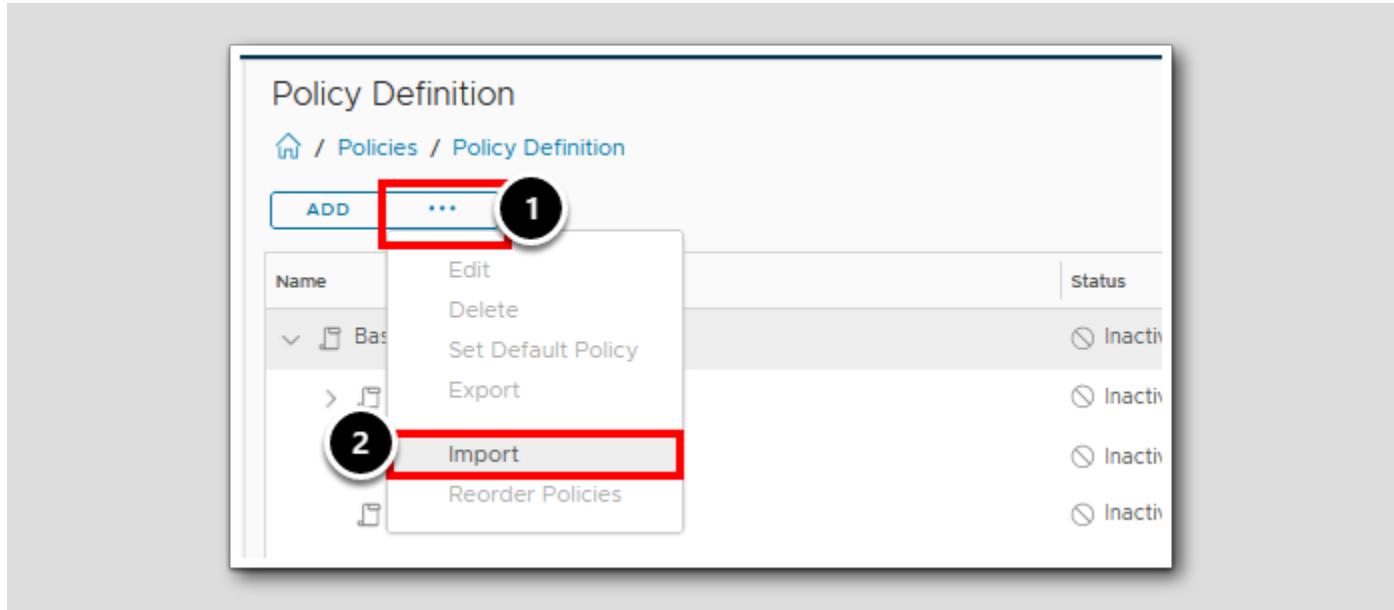
The next step in configuring the integration between Operations and Orchestrator is to enable the alerts to monitor our target VMs for high or low CPU utilization. In this lab we will be importing a Aria Operations Policy containing alerts, recommendations, and a monitoring policy configured with those alerts enabled. Additionally, we will be importing a Custom Group monitored by the imported policy. The Custom Group will dynamically include virtual machines with the `alertPolicy:autoscale` tag.

## Navigate to Operations Policy Configuration



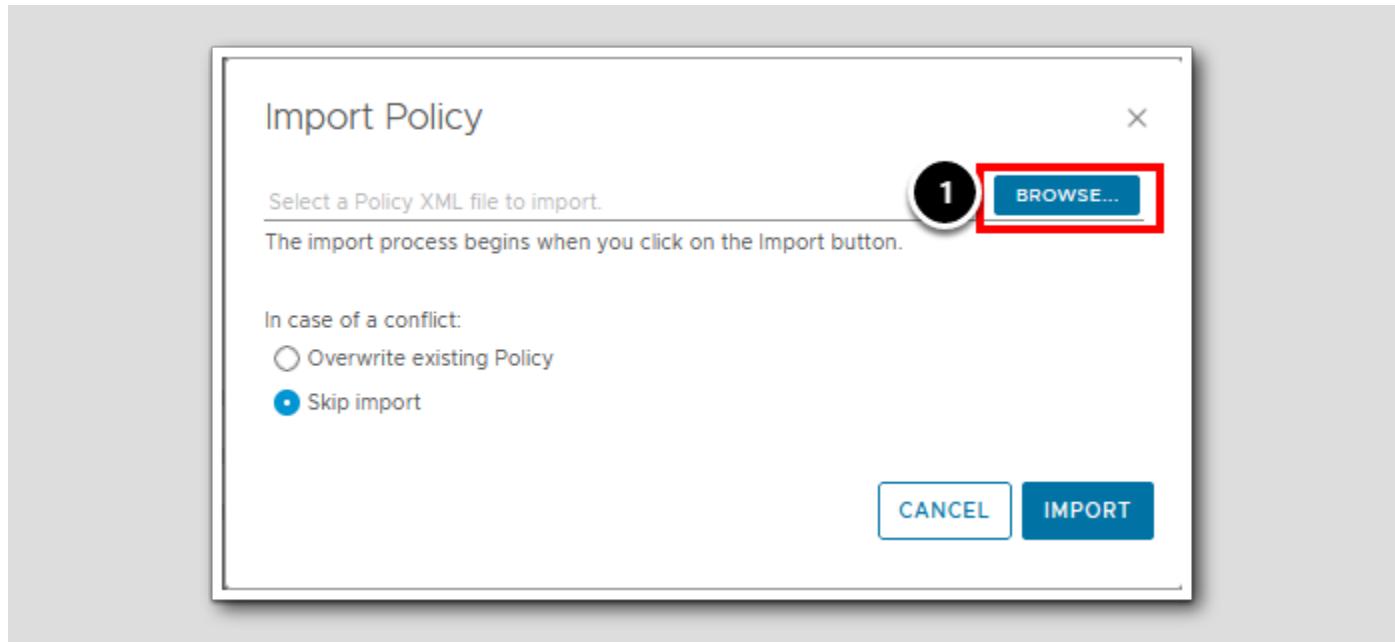
1. Expand **Configure**
2. Select the **Policies** section of the configure menu
3. Select **Policy Definition**

## Import a new Policy



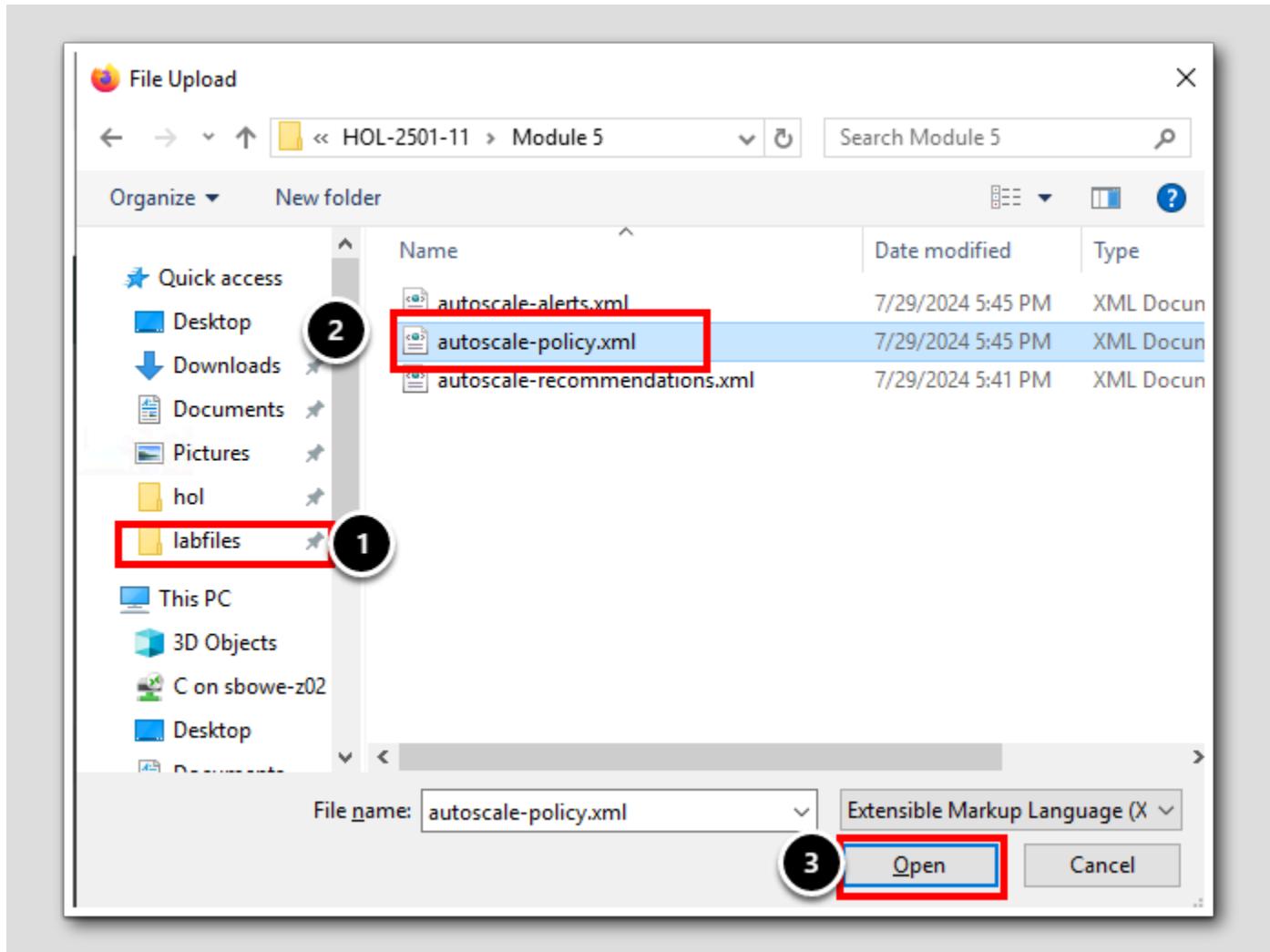
1. Click the ... button to open a context menu
2. Select Import to launch the import policy wizard

## Select Policy



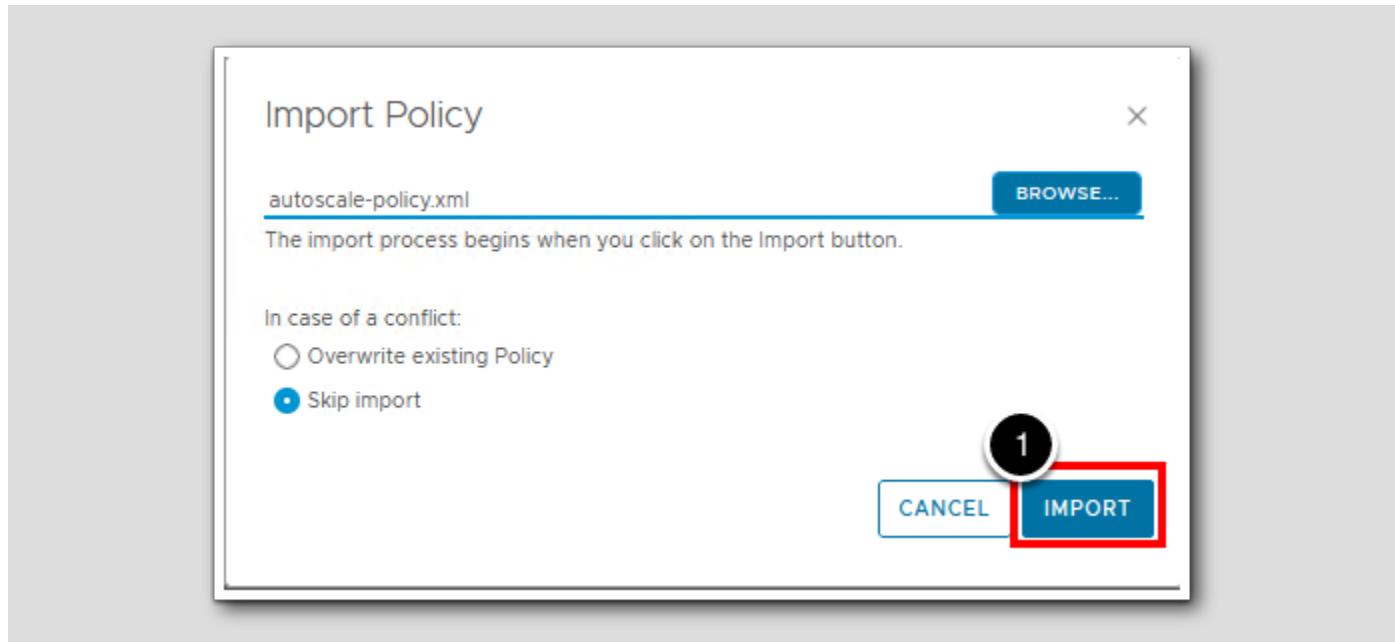
1. Click BROWSE...

## Select Policy Continued



1. Select labfiles in the Quick Access Menu
  - Open the folder HOL-2501-11 (not shown)
  - Open the folder Module 5 (not shown)
2. Select autoscale-policy.yml
3. Click the Open button

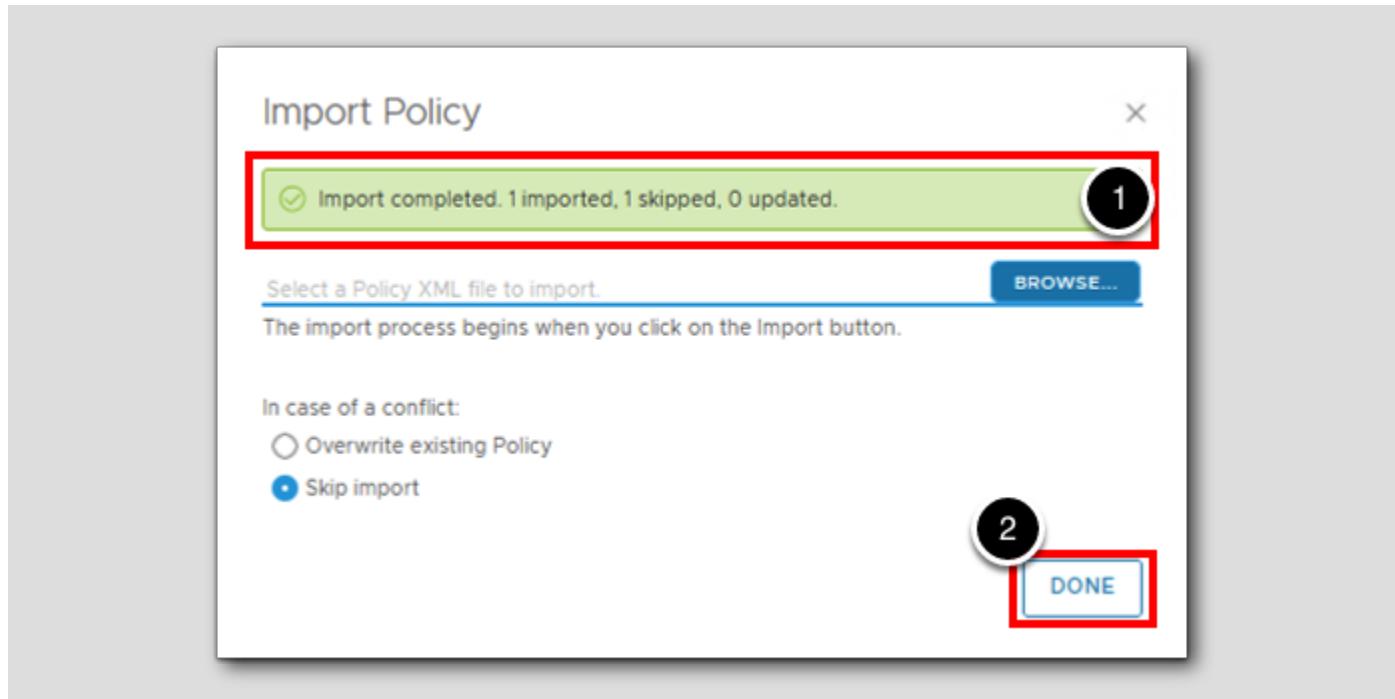
## Import Selected Policy



1. Click IMPORT

As mentioned previously, importing a policy imports not just the monitoring rules, but associated alerts, recommendations, etc. Let's validate that everything was imported properly!

## Import Policy Continued



1. When the import completes a green banner is displayed confirming the success of the import. If the content is already present within the environment the import is skipped, as we told Aria Operations not to overwrite in the case of a conflict when starting the import.
2. Click DONE to close the dialog box.

## Validate Import

[297]

The screenshot shows the 'Policy Definition' page in the vSphere Web Client. The table lists various policies under the 'Base Settings' category. The 'vSphere Solution's Default Policy' is expanded, revealing its child policy, 'HOL Autoscale Policy'. Both policies are currently inactive.

| Name  | Status   | Priority | Description  | Last Modified  | Modified By        |
|---|----------|----------|--|----------------|--------------------|
| Base Settings   | Inactive |          |  | 22 days ago    |                    |
| Config Wizard Based Policy  | Inactive |          |  | 22 days ago    | VMware             |
| Default Policy  | Inactive |          |  | 2 months ago   |                    |
| Foundation Policy   | Inactive |          |  | 22 days ago    | VMware             |
| HOL Policy  | Inactive |          |  | 28 days ago    | holadmin           |
| NSX Security Configuration Guide                                    | Inactive |          | NSX Security Configuration Guide. Recommendations av...    | 22 days ago    | VMware             |
| Policy for Virtual Machines - Risk Profile 1                        | Inactive |          | This policy applies vSphere Security Configuration Guid... | 22 days ago    | VMware             |
| Policy for Virtual Machines - Risk Profile 2                        | Inactive |          | This policy applies vSphere Security Configuration Guid... | 22 days ago    | VMware             |
| Policy for Virtual Machines - Risk Profile 3                        | Inactive |          | This policy applies vSphere Security Configuration Guid... | 22 days ago    | VMware             |
| vSAN Security Configuration Policy                                  | Inactive |          |  | 22 days ago    | vSAN Security C... |
| <b>1 vSphere Solution's Default Policy (Jul 1, 2024 3:12:17 PM)</b> | Active   |          |  | 28 days ago    | holadmin           |
| <b>2 HOL Autoscale Policy</b>                                       | Inactive |          |  | 21 seconds ago | holadmin@vcf.h...  |

1. Expand vSphere Solution's Default Policy (Jul 1, 2024 3:12:17 PM)
2. If the policy imported successfully, we will see a child policy underneath it named HOL Autoscale Policy.

Next let's check and make sure the recommendations were successfully imported.

## Navigate to recommendations

[298]

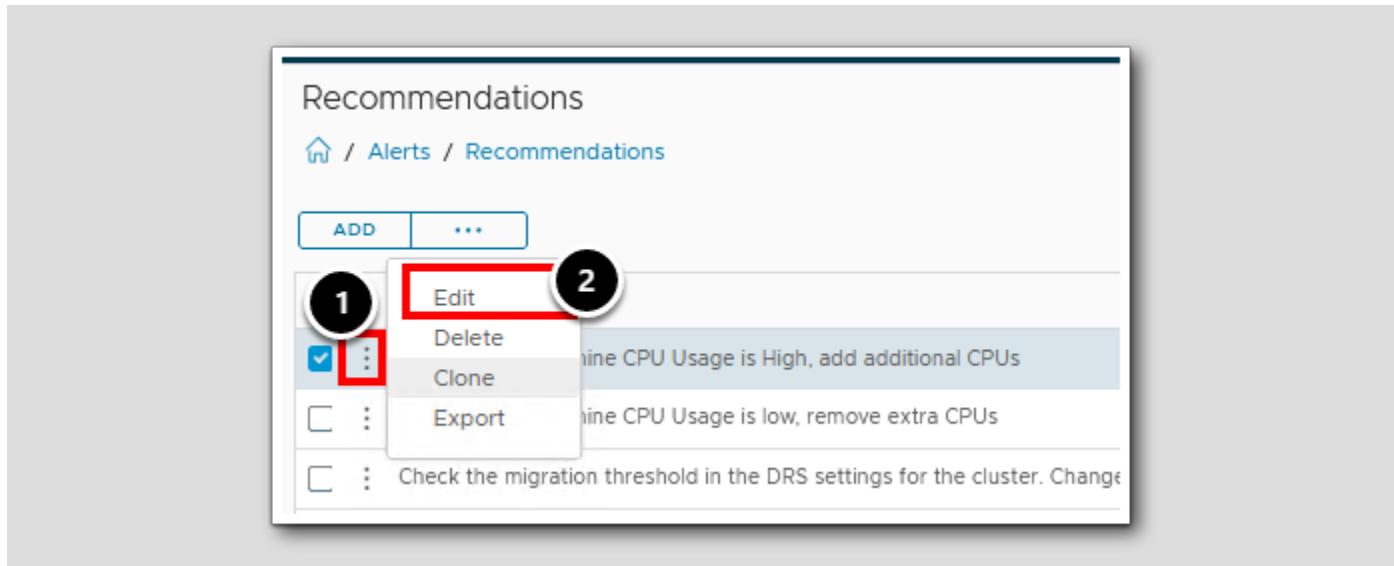
The screenshot shows the VMware Aria Operations interface. The left sidebar has several sections: Environment, Visualize, Troubleshoot, Optimize, Plan, Configure (highlighted with a red box and number 1), Policies, and Alerts (highlighted with a red box and number 2). The main content area is titled 'Alerts' and contains the following sections:

- Alert Definitions**: Create and edit Alert definitions using a combination of symptoms and recommendations that identify problem areas in your environment and generate alerts on which you act to remediate the issues.
- Symptom Definitions**: Create and edit descriptions of situations which are NOT normal within your environment. Use these symptoms in your Alert definitions.
- Recommendations**: Create and edit remediation options that you provide to your users to resolve the problems that the generated alert indicates. This section is highlighted with a red box and number 3.
- Notifications**: Define and modify notification settings to send out messages and custom payloads when an alert is triggered.
- Outbound Settings**: Define and manage outbound notification methods using a variety of protocols such as SNMP, web hook, network sharing and more to allow notifications to be dispatched when an alert is triggered.
- Payload Templates**: Custom outbound notification payload editor for protocols which support custom payloads.
- Actions**: Create and edit actions to make changes to objects managed in your vCenter Server instances.

Recommendations are a way to associate guidance to resolve a problem or an action to execute either on demand or automatically, in order to resolve a triggered alert. To execute our workflows we must associate them to the recommendations.

1. Expand Configure
2. Open Alerts by selecting it.
3. Click on Recommendations

## Edit High CPU recommendation



At the top of the recommendations list, should be two recommendations [HOL] Virtual Machine CPU Usage is High, add additional CPUs and [HOL] Virtual Machine CPU Usage is low, remove extra CPUs. The naming of them should cause them to be sorted to the top, if not you can search for them by typing HOL into the filter box in the top right corner of the screen.

1. Click the **three vertical dots** to expand a context menu on [HOL] Virtual Machine CPU Usage is High, add additional CPUs
2. Select **Edit**

## Select Action

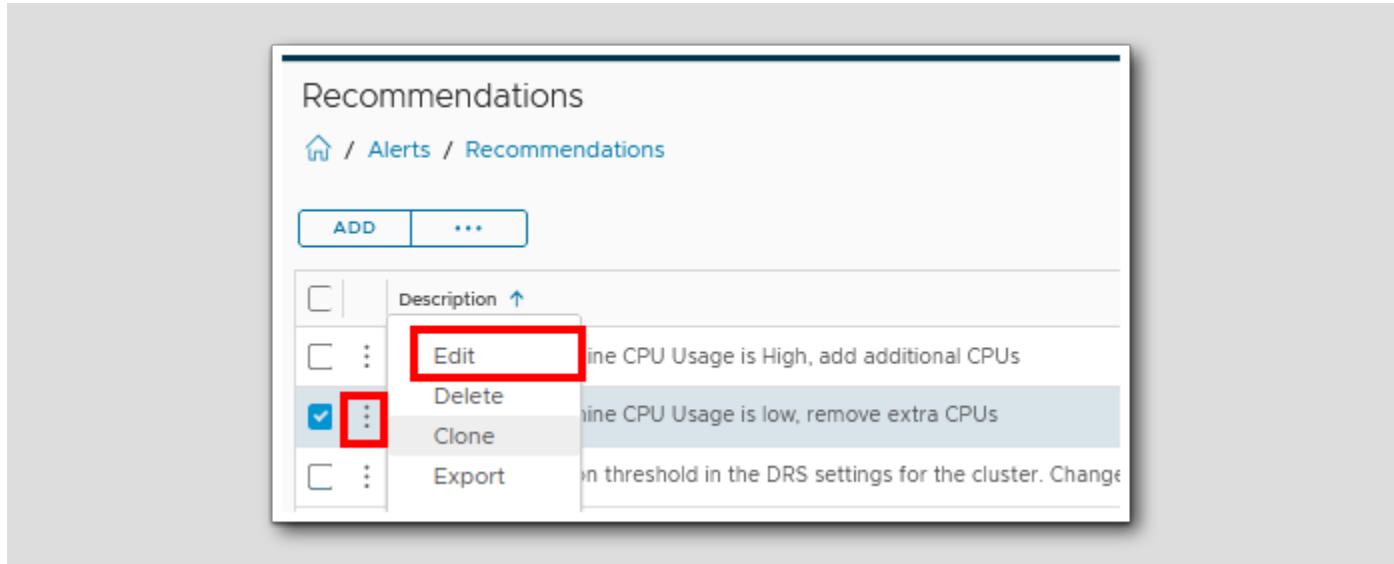
The screenshot shows the 'Edit Recommendation' dialog box. At the top, there's a breadcrumb navigation: Home / Alerts / Recommendations. Below that, a placeholder text says 'Add a description and select an action to your new recommendation.' Under 'Description', there's a text input field containing '[HOL] Virtual Machine CPU Usage is High, add additional CPUs'. In the 'Alert Definitions (1)' section, there's a 'REMOVE FROM ALL' button and a table with one row: Name ↑ [HOL] CPU Usage is High. The 'Action (Optional)' section has an 'Adapter Type' dropdown set to '--All--' and an 'Action' dropdown where 'AutoScale-Out' is selected. A red box labeled '1' highlights the 'Action' dropdown. At the bottom, there are two buttons: 'SAVE' (highlighted with a red box and labeled '2') and 'CANCEL'.

1. Select AutoScale-Out

2. Click SAVE

Next, we need to update the Low CPU recommendation with the appropriate workflow.

## Edit Low CPU recommendation



1. Click the three vertical dots for [HOL] Virtual Machine CPU Usage is low, remove extra CPUs to expand a context menu
2. Select Edit

## Select Action

Edit Recommendation

[Home](#) / [Alerts](#) / [Recommendations](#)

Add a description and select an action to your new recommendation.

Description [🔗](#)

[HOL] Virtual Machine CPU Usage is low, remove extra CPUs

Alert Definitions (1)

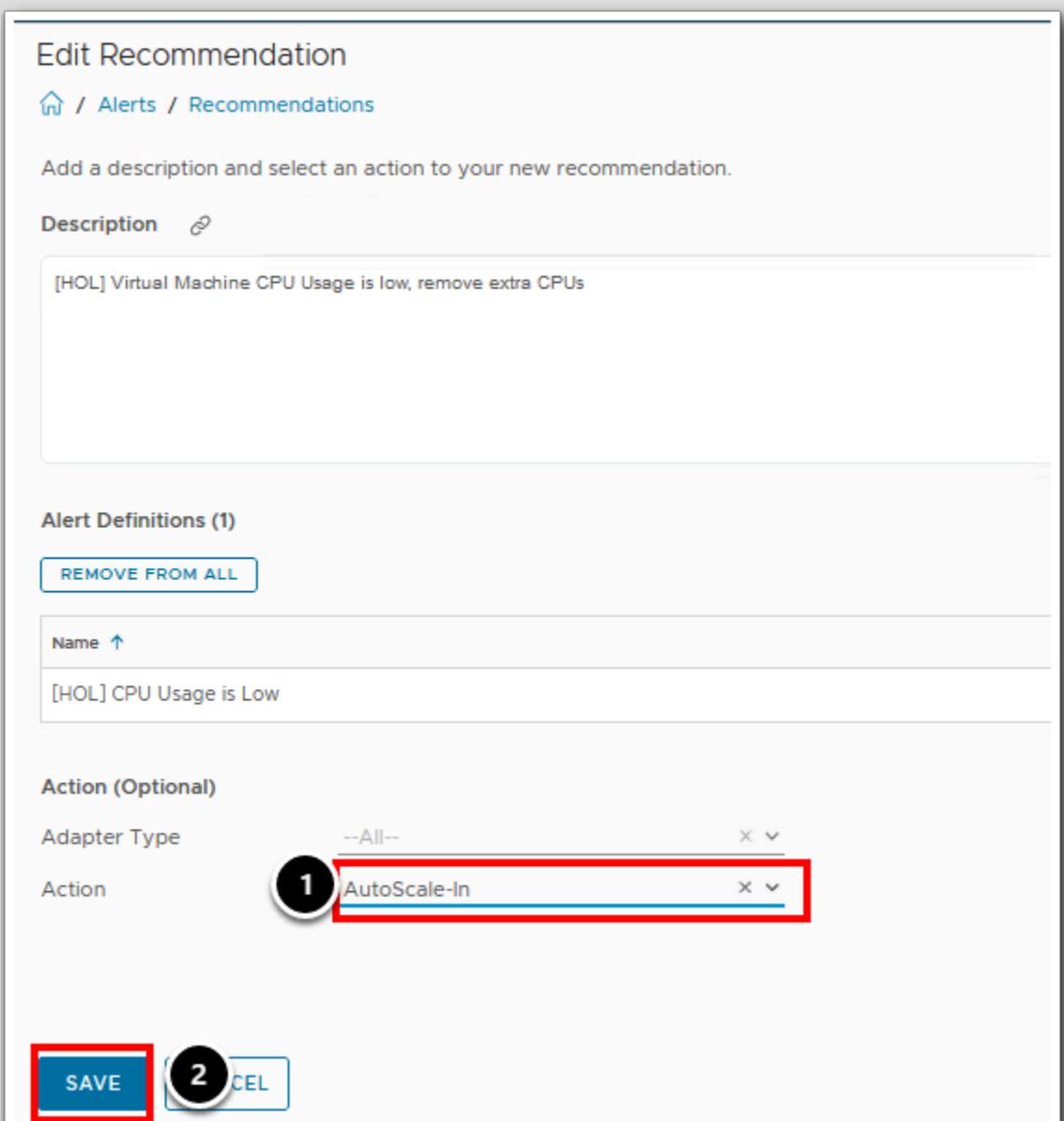
[REMOVE FROM ALL](#)

| Name ↑                 |
|------------------------|
| [HOL] CPU Usage is Low |

Action (Optional)

Adapter Type --All--

Action **1** AutoScale-In **2** CANCEL

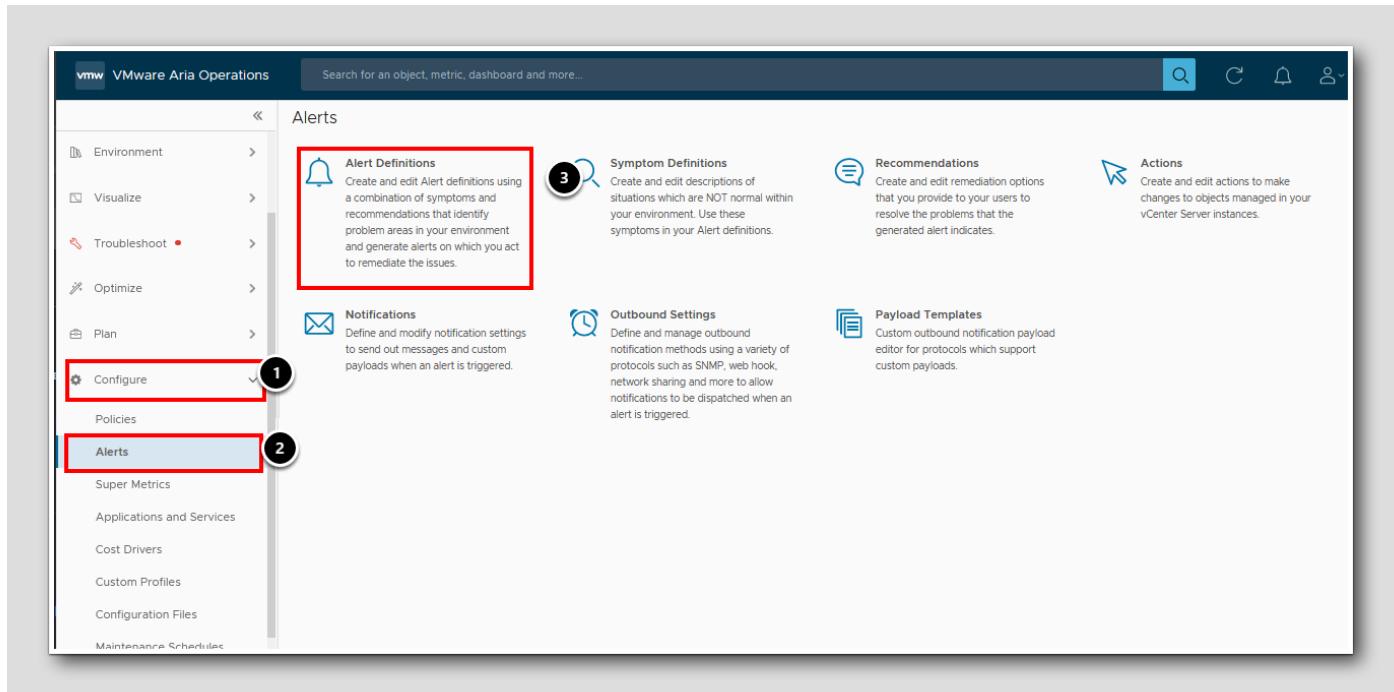


1. Select AutoScale-In

2. Click SAVE

The final step to validate and configure our recommendations and alerts is to validate the Alert Definitions themselves.

## Navigate to Alerts



1. Expand **Configure**
2. Open **Alerts** by selecting it.
3. Click on **Alert Definitions**

## Validate Alert Import

The screenshot shows the 'Alert Definitions' page in the vRealize Operations Management Cloud interface. On the left, a sidebar lists various management categories like Environment, Visualize, Troubleshoot, Optimize, Plan, Configure, Policies, and Alerts. The 'Alerts' category is selected. The main pane displays a list of alert definitions. One alert, '[HOL] CPU Usage is High', is highlighted with a red box and has a red circle with the number '1' over it. To the right of the list is a detailed view of this specific alert. This view includes fields for 'Name' ('[HOL] CPU Usage is High'), 'Impact' (Health), 'Alert Type / Subtype' (Application: Performance), and 'Wait / Cancel Cycle' (1/1). Below these are sections for 'Symptoms / Conditions (1)' and 'Conditions'. Under 'Conditions', there is a condition: 'If CPUUsage (%) > 85 mark as Critical'. The next section, 'Recommendations (1)', is also highlighted with a red box and has a red circle with the number '2' over it. It contains a single recommendation: '[HOL] Virtual Machine CPU Usage is High, add additional CPUs'. At the bottom, there is a section for 'Policies Enabled (0)'. The status bar at the bottom of the interface shows 'hol'.

1. Select [HOL] CPU Usage is High by clicking on it
2. Expand Recommendations
  - Validate the High CPU Usage recommendation is associated with the alert
3. Repeat the previous two steps for [HOL] CPU Usage is Low (not shown)
  - Validate the Low CPU Usage recommendation is associated with the alert

Our required alert definitions are included in our policy import. The final piece we need to import is our group definition. This Custom Group will include all deployed virtual machines with the `alertPolicy:autoscale` tag, and will monitor included objects with our HOL Autoscale Policy. This Custom Group is critical to ensure that we do not attempt to run our workflows against unintended targets!

## Import Group

The screenshot shows the 'Custom Groups' page in the vSphere Web Client. The left sidebar has 'Environment' selected. In the main pane, there's a table of groups. A context menu is open over the 'Import' option in the 'Actions' column of the first row. The menu items are numbered 1 through 4: 1 points to 'ADD', 2 points to 'Custom Groups', 3 points to the ellipsis button, and 4 points to 'Import'.

|                          |                        | Health             | Group Type  | Description                              | Members Count | Policy                               | Dynamic Member... | Defined by      |
|--------------------------|------------------------|--------------------|-------------|--|---------------|--------------------------------------|-------------------|-----------------|
| <input type="checkbox"/> | Import                 | <span>Green</span> | Environment | List of vSphere datastores that a...     | 0             | vSphere Solution's Default Policy... | true              | vSAN            |
| <input type="checkbox"/> | NSXT World             | <span>Green</span> | NSX World   | This is a built-in group, hence it is... | 1             | vSphere Solution's Default Policy... | false             | NSX             |
| <input type="checkbox"/> | Operating System World | <span>Green</span> | Environment | The top level group for VMware ...       | 0             | vSphere Solution's Default Policy... | true              | OS and Appli... |
| <input type="checkbox"/> | Universe               | <span>Green</span> | Universe    | The highest level group in Opera...      | 13            | vSphere Solution's Default Policy... | false             | System          |
| <input type="checkbox"/> | VCF Management World   | <span>Green</span> | Environment | VCF Management World                     | 1             | vSphere Solution's Default Policy... | true              | VMware Cloud... |
| <input type="checkbox"/> | VCF Workload World     | <span>Green</span> | Environment | VCF Workload World                       | 0             | vSphere Solution's Default Policy... | true              | VMware Cloud... |
| <input type="checkbox"/> | VCF World              | <span>Green</span> | VCF World   | This is a built-in group, hence it is... | 1             | vSphere Solution's Default Policy... | false             | VMware Cloud... |
| <input type="checkbox"/> | VMC World              | <span>Green</span> | VMC World   | This is a built-in group, hence it is... | 23            | vSphere Solution's Default Policy... | false             | VMware Cloud... |

1. Expand Environment
2. Open Custom Groups
3. Click the ... button to launch a context menu
4. Select Import to launch the Group import wizard

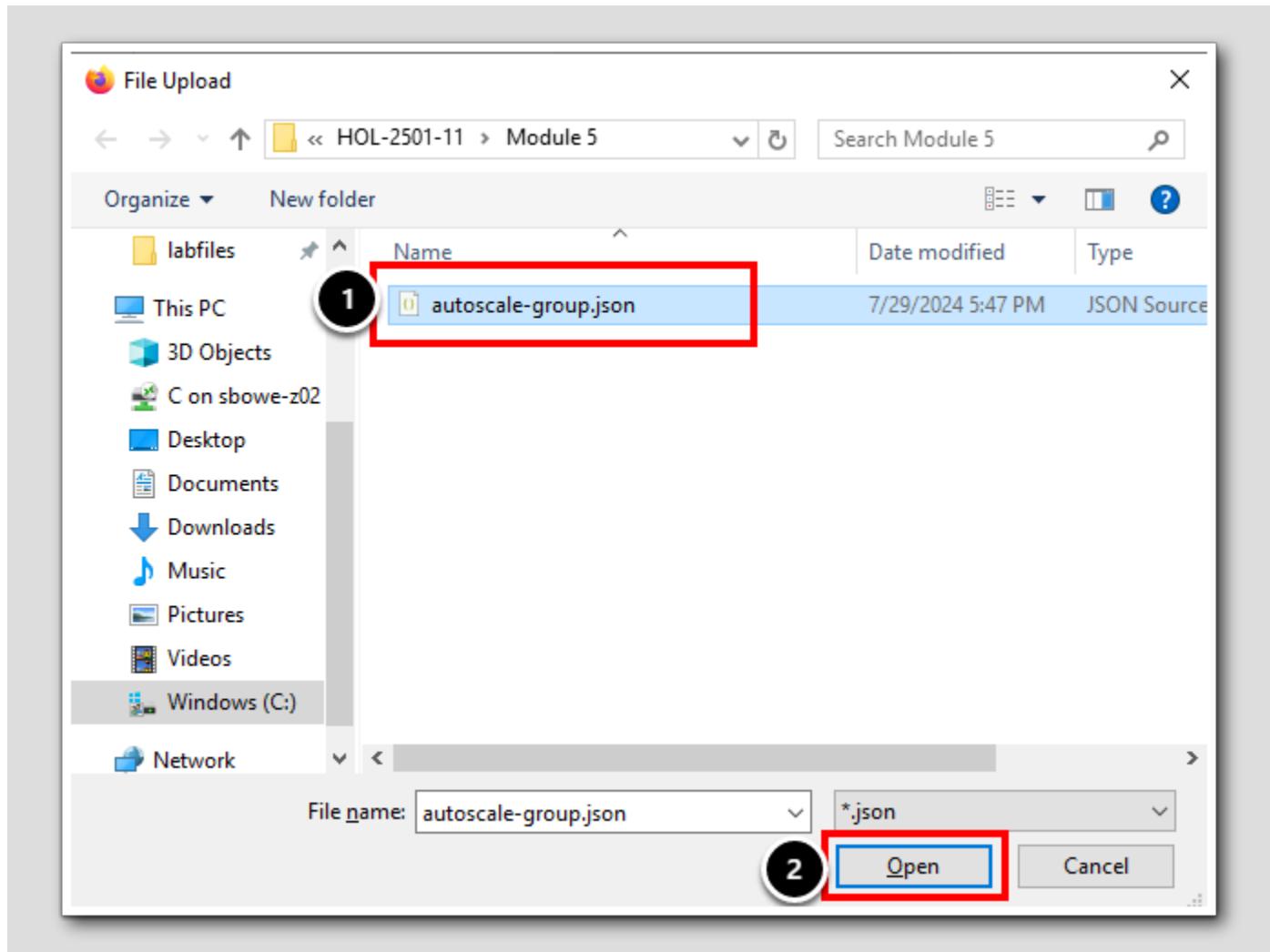
## Browse for Group

The screenshot shows the 'Import Custom Group(s)' dialog box. It has a 'Select a JSON file to import' input field and a note: 'The import process begins when you click on the Import button.' Below that, it says 'In case of a conflict:' with two radio button options: 'Overwrite existing Custom Group' (selected) and 'Skip import'. At the bottom are 'CANCEL' and 'IMPORT' buttons.

1. Click the BROWSE button

## Select Group

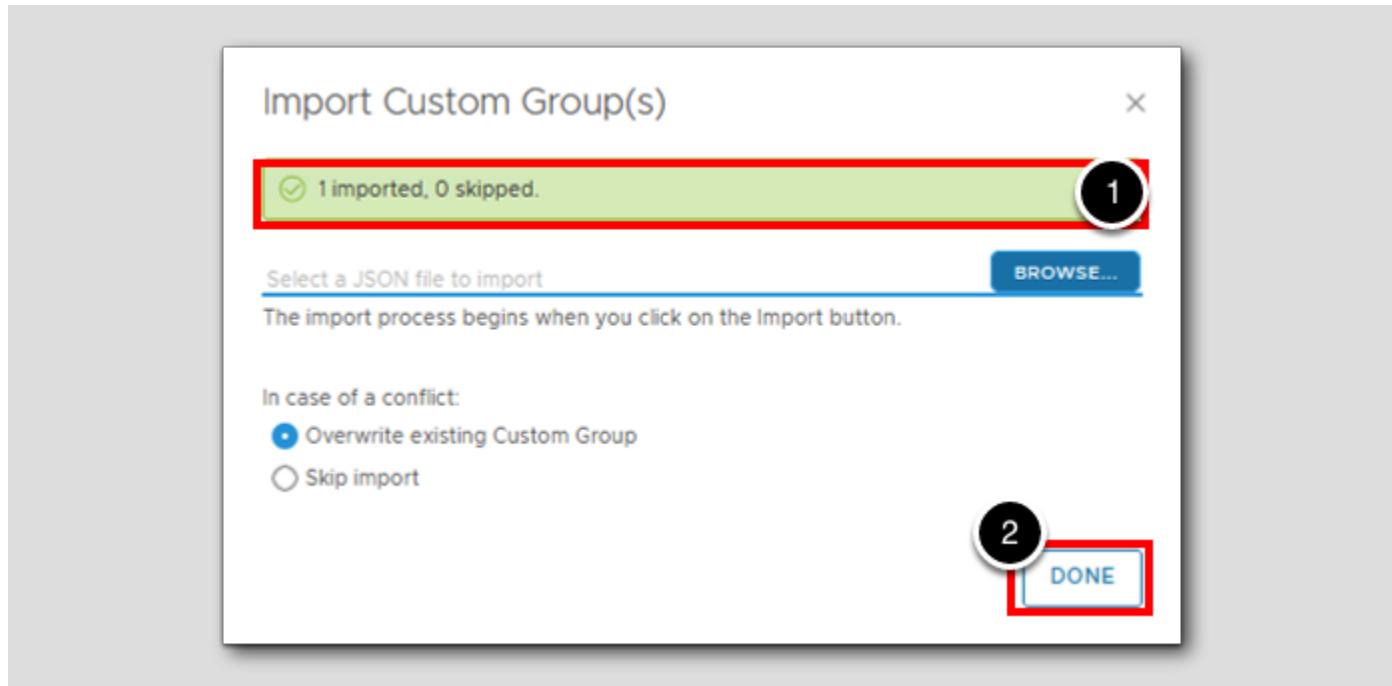
[307]



1. Select autoscale-group.json
2. Click the Open button
3. Click the IMPORT button (not shown)

After importing the group we must validate that it imported correctly.

## Complete the Import



1. When the import has completed a green banner will be displayed confirming the success of the operation.
2. Click DONE to close the dialog box

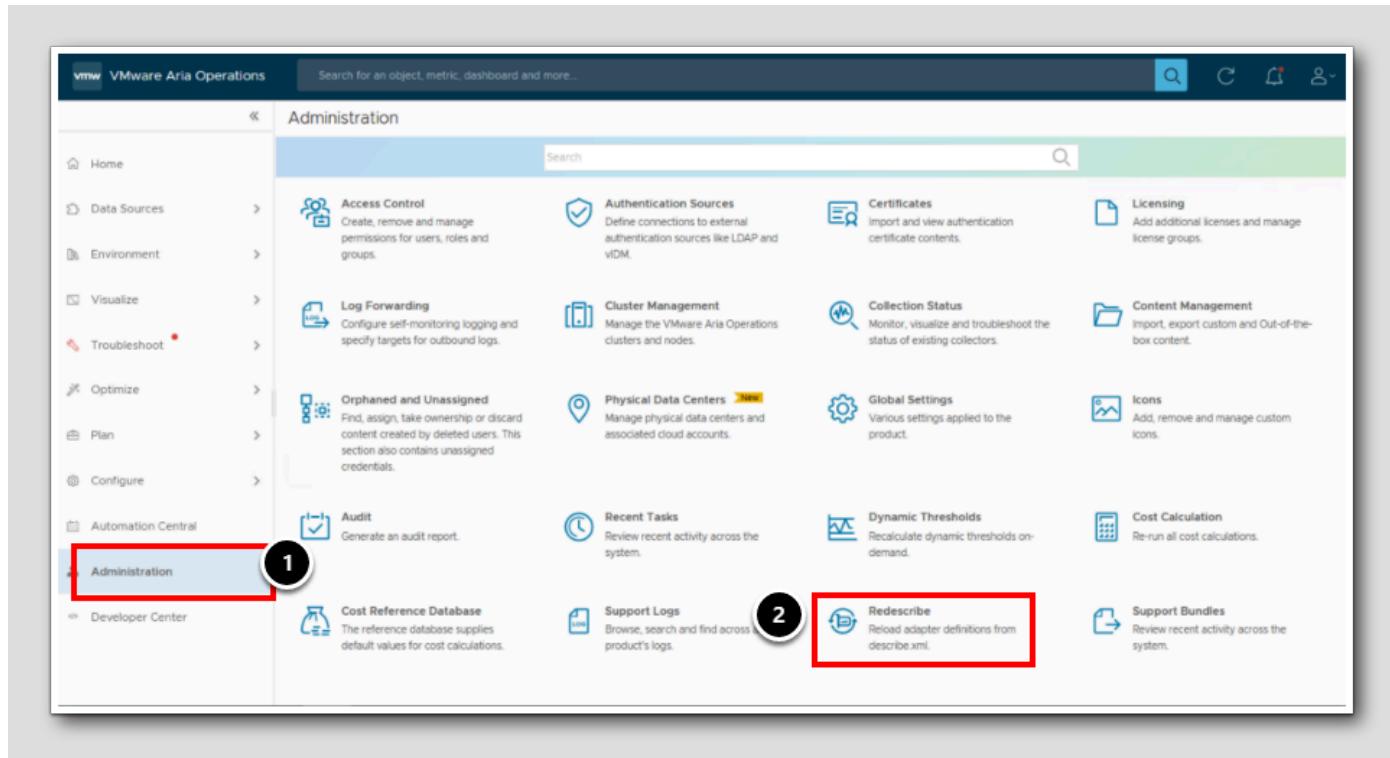
## Validate Group

The screenshot shows the 'Custom Groups' interface in Aria Operations. On the left, a list of groups is shown, with 'AutoScale Policy Group' selected (marked with a red box and number 1). On the right, the details for 'AutoScale Policy Group' are displayed. Under 'Defined membership criteria', there is an entry 'Tag: alertPolicy contains autoscale' (marked with a red box and number 3). In the top right corner of the details pane, the associated policy is listed as 'HOL Autoscale Policy' (marked with a red circle and number 4).

1. Select the group AutoScale Policy Group by clicking on it
2. Expand Defined membership criteria
3. Validate it references the alertPolicy tag category and the value autoscale
4. Validate that the associated policy is HOL Autoscale Policy

The last step before we can test our new automated alert, is to enable it for automation in Aria Operations. We do this by updating our monitoring policy. Because we are importing alerts and policies, rather than creating them in this environment, we must instruct Aria Operations to reload its alert and adapter configurations. We do this by performing a **Redescribe**.

## Open Redescribe Wizard



1. Click on **Administration** in the navigation pane, to open the Administration menu.
2. Select **Redescribe**

A redescription causes Aria Operations to reload the configuration of all adapters. As part of this process, things like alert definitions are reloaded. Without reloading or updating our imported alerts, we would be unable to enable automation on them.

## Trigger Redescribe

[311]

Redescribe

Rerun the describe process on an adapter that was installed or updated.

**REDESCRIBE**

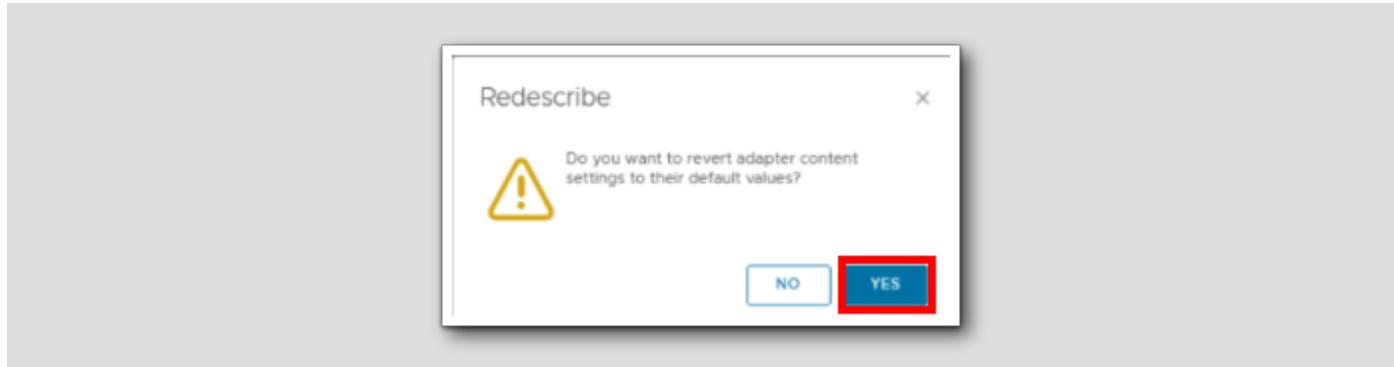
1 operation completed successfully.

| Name   | Status | Describe Version | Adapter Version | Me |
|--|--------|------------------|-----------------|----|
| Container  | ✓      | 64               | -               |    |
| NSX  | ✓      | 31               | 8.17.23642267   |    |
| Service Discovery Adapter                          | ✓      | 52               | 8.17.23642265   |    |
| vCenter  | ✓      | 945              | 8.17.23642280   |    |
| VMware Aria Automation                             | ✓      | 13               | 8.17.23642245   |    |
| VMware Aria Automation Orchestrator Adapter        | ✓      | 17               | 3.3.21620483    |    |
| VMware Aria Operations Application Management P... | ✓      | 93               | 8.17.23642275   |    |
| VMware Aria Operations for Logs Adapter            | ✓      | 17               | 8.17.23642277   |    |
| VMware Cloud Foundation                            | ✓      | 11               | 8.17.23642252   |    |
| VMware Cloud on AWS                                | ✓      | 19               | 8.17.23642258   |    |
| VMwareAriaOperationsAPI                            | ✓      | 1                | -               |    |
| vRealize Operations Adapter                        | ✓      | 342              | 2.0.0           |    |
| vSAN Adapter                                       | ✓      | 31               | 8.17.23642273   |    |

1. Click the REDESCRIBE button

## Confirm Redescribe

[312]

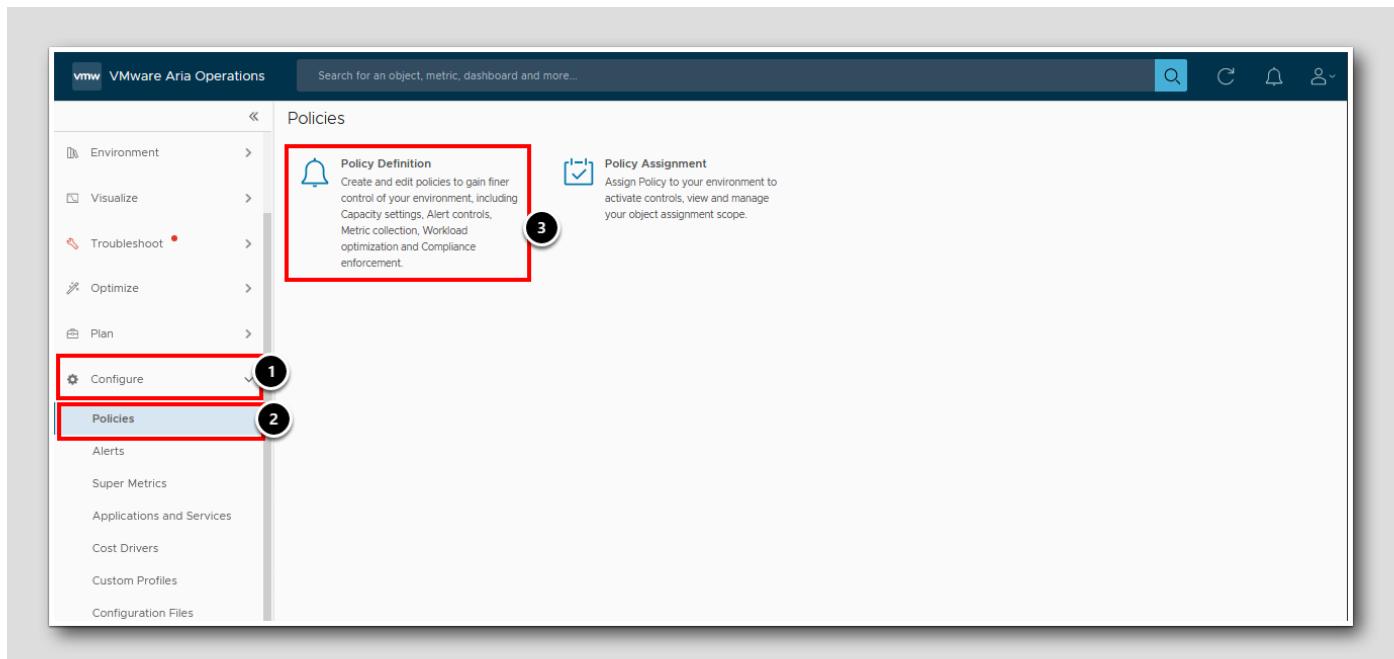


1. Click YES to confirm the redescribe and reload the adapter settings.

The redescribe will take 1 to 2 minutes. While you can continue to the next step, to prevent extra work or waiting - just wait until the screen refreshes.

## Navigate to Policies

[313]



1. Expand the Configure Menu
2. Open Policies
3. Click Policy Definition

## Edit Policy

[314]

The screenshot shows the 'Policy Definition' screen in the vSphere web interface. On the left, there's a navigation pane with an 'ADD' button and a search bar. The main area displays a list of policies under 'Base Settings'. A policy named 'HOL Autoscale Policy' is selected, highlighted with a red box and numbered '2'. To the right of the list, detailed information about the selected policy is shown: 'Parent policy: vSphere Solution's Default Policy (Jul 1, 2024 3:12:17 PM)' and 'Priority: 1'. Below this, several policy sections are listed with their respective counts and edit icons: Metrics and Properties (Local: 0 Attributes), Alerts and Symptoms (Local: 2 Alerts / 0 Symptoms), Capacity (Local: 0 Policy elements), Maintenance Schedule (Local: 0 Policy elements), Compliance (Local: 0 Policy elements), Workload Automation (Local: 0 Policy elements), and VC Pricing (Local: 0 Pricing Sections). An 'EDIT POLICY' button is located at the top right of the policy details section, also highlighted with a red box and numbered '3'.

1. Expand the vSphere Solution's Default Policy by clicking the arrow
2. Select HOL AutoScale Policy by clicking on it
3. Click EDIT POLICY

## Select Alerts and Symptoms

HOL Autoscale Policy [Edit]

Name: HOL Autoscale Policy

Description: - None -

Inherit From: vSphere Solution's Default Policy (Jul 1, 2024 3:12:17 PM)

Metrics and Properties 1

Locally defined attributes None

Alerts and Symptoms

Locally defined alerts 2

Locally defined symptoms None

Capacity

Locally defined policy elements None

Maintenance Schedule

Locally defined policy elements None

Compliance

Locally defined policy elements None

Workload Automation

Locally defined policy elements None

The 'Alerts and Symptoms' section is highlighted with a red box and a circled '1' above it.

1. Open Alerts and Symptoms

## Automate Alerts

[316]

HOL Autoscale Policy [Edit]

Alerts and Symptoms

Select Object Type: **Virtual Machine** 1

Type here to apply filters 🔍

| Alert Definition   | State  | Automate  | Symptoms / Conditions | Criticality                         |
|--|--|---|-----------------------|-------------------------------------|
| [HOL] CPU Usage is High  | Activated  | <span style="border: 1px solid red; padding: 2px;">Activated</span> <span style="border: 1px solid black; border-radius: 50%; padding: 5px;">2</span> | 1                     | <span style="color: red;">!!</span> |
| [HOL] CPU Usage is Low   | Activated  | <span style="border: 1px solid red; padding: 2px;">Activated</span> <span style="border: 1px solid black; border-radius: 50%; padding: 5px;">3</span> | 1                     | <span style="color: red;">!!</span> |
| Hands-On Labs High CPU Alert   | Deactivated <span style="border: 1px solid #ccc; border-radius: 50%; padding: 2px;">Inherited</span> | Not Applicable  | 1                     | <span style="color: red;">!!</span> |
| Not enough resources for vSphere HA to start the virtual machine                 | Activated <span style="border: 1px solid #ccc; border-radius: 50%; padding: 2px;">Inherited</span>   | Deactivated <span style="border: 1px solid #ccc; border-radius: 50%; padding: 2px;">Inherited</span>  | 1                     | <span style="color: red;">!!</span> |
| One or more monitored service(s) are unavailable on the virtual machine          | Activated <span style="border: 1px solid #ccc; border-radius: 50%; padding: 2px;">Inherited</span>   | Not Applicable  | 1                     | <span style="color: red;">!!</span> |
| One or more virtual machine guest file systems are running out of disk space     | Activated <span style="border: 1px solid #ccc; border-radius: 50%; padding: 2px;">Inherited</span>   | Not Applicable  | 2                     | <span style="color: red;">!!</span> |
| The Fault tolerance state of the virtual machine has changed to "Needs Seco...   | Activated <span style="border: 1px solid #ccc; border-radius: 50%; padding: 2px;">Inherited</span>   | Not Applicable  | 1                     | <span style="color: red;">!!</span> |
| The Fault tolerance state of the virtual machine has changed to a "Deactivate... | Activated <span style="border: 1px solid #ccc; border-radius: 50%; padding: 2px;">Inherited</span>   | Not Applicable  | 1                     | <span style="color: red;">!!</span> |
| Virtual machine CPU usage is at 100% for an extended period of time              | Activated <span style="border: 1px solid #ccc; border-radius: 50%; padding: 2px;">Inherited</span>   | Not Applicable  | 1                     | <span style="color: red;">!!</span> |
| Virtual machine disk I/O read latency is high                                    | Deactivated <span style="border: 1px solid #ccc; border-radius: 50%; padding: 2px;">Inherited</span> | Not Applicable  | 3                     | <span style="color: red;">!!</span> |

4 SAVE

1. Choose **Virtual Machine** from within the vCenter category as the object type
2. Select **Activated** for the Automate column of the [HOL] CPU Usage is High alert
3. Select **Activated** for the Automate column of the [HOL] CPU Usage is Low alert
4. Click the **SAVE** button

In this section we imported the required alerts, recommendations, group, and policy to allow for an automated remediation. By importing these components we are mirroring real world best practices of creating custom alerts and policies to fit our needs, rather than edit the build in alerts, which could present problems when default content is updated as part of upgrades.

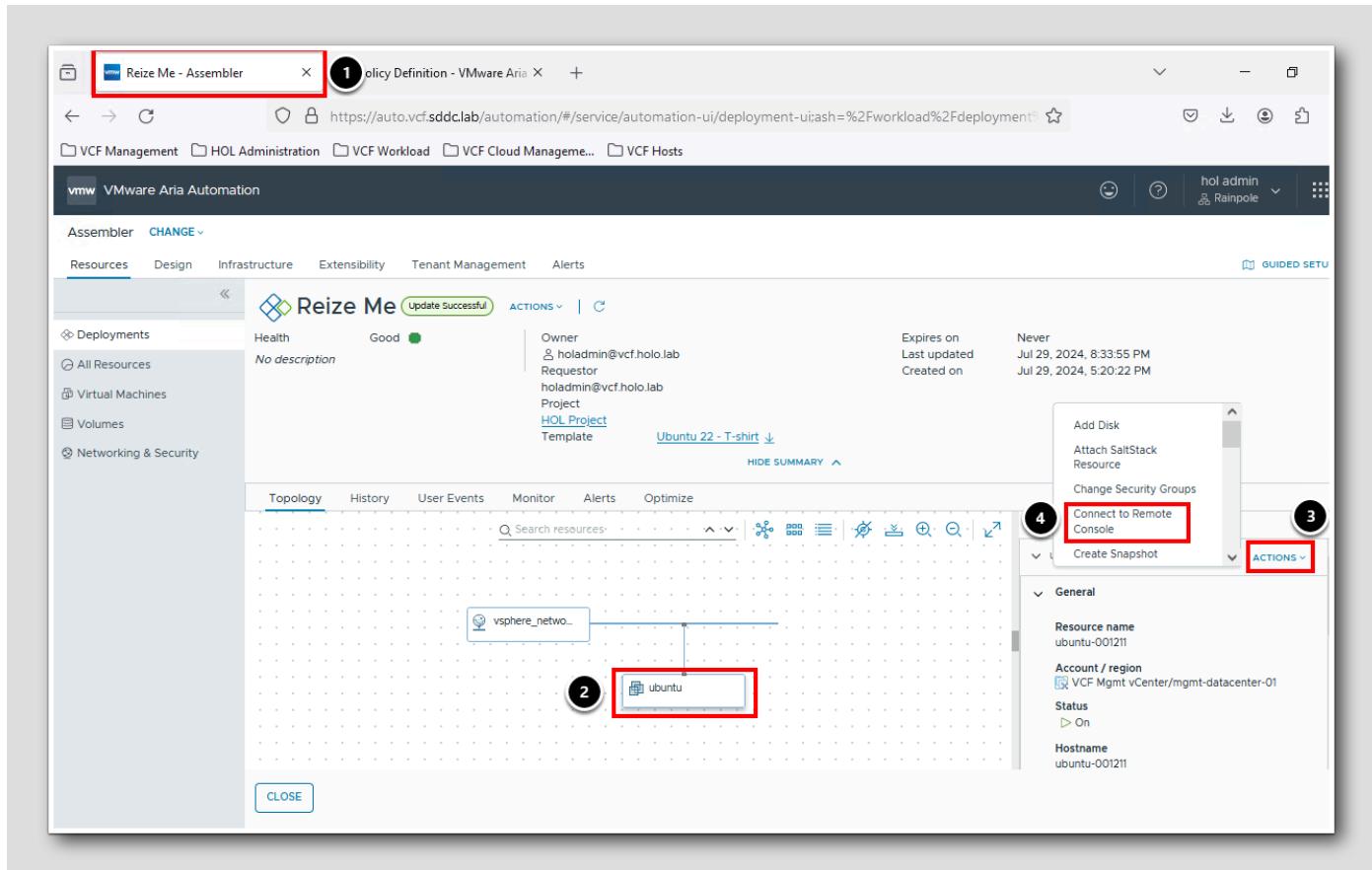
In the next lesson we will trigger the high CPU usage alert on our previously deployed VM.

## Trigger High CPU Alarm

[317]

In this lesson we will test our autoscale workflows by generating a high cpu utilization on our previously deployed virtual machine. We will validate that the utilization triggers our workflows via Aria Operations, causing the virtual machine to be scaled up in Aria Automation.

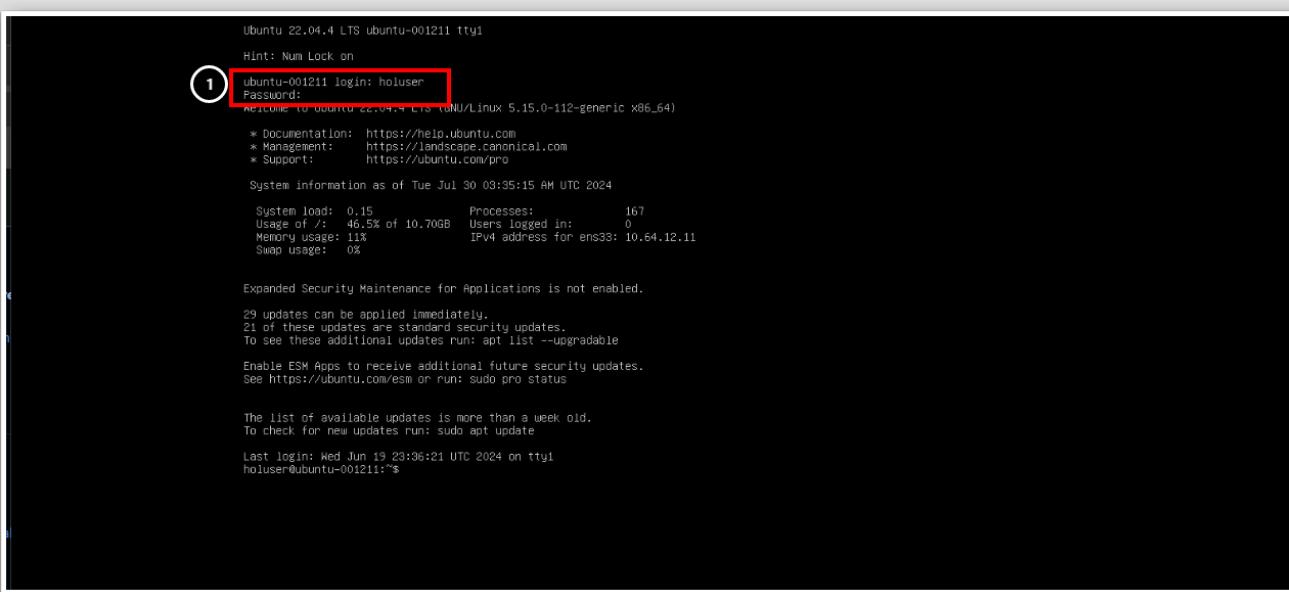
## Launch a remote console



1. Click on the browser tab for Aria Automation. The deployment we started earlier, should be complete.
2. Select the **ubuntu** virtual machine resource on the canvas to change the focus to the virtual machine
3. Expand the **Actions** pull down to launch a context menu of Day 2 operations
4. Launch the remote console by selecting **Connect to Remote Console**, a new browser tab will launch. It will take a few seconds to connect.

After the remote console is connected we will log into the virtual machine and load the CPU by executing a command.

## Login to VM



```

Ubuntu 22.04.4 LTS ubuntu-001211 tty1
Hint: Num Lock on
① ubuntu-001211 login: holuser
Password:
Welcome to Ubuntu 22.04.4 LTS (GNU/Linux 5.15.0-112-generic x86_64)

 * Documentation: https://help.ubuntu.com
 * Management: https://landscape.canonical.com
 * Support: https://ubuntu.com/pro

System information as of Tue Jul 30 03:35:15 AM UTC 2024

System load: 0.15           Processes:          167
Usage of /: 46.5% of 10.70GB  Users logged in: 0
Memory usage: 11%           IPv4 address for ens3: 10.64.12.11
Swap usage: 0%              

Expanded Security Maintenance for Applications is not enabled.

29 updates can be applied immediately.
21 of these updates are standard security updates.
To see these additional updates run: apt list --upgradable

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

The list of available updates is more than a week old.
To check for new updates run: sudo apt update

Last login: Wed Jun 19 23:36:21 UTC 2024 on ttys0
holuser@ubuntu-001211:~$
```

1. Log into the virtual machine using the following credentials

- Username: **holuser**
- Password: **VMware123!**

## Load CPU



```

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

The list of available updates is more than a week old.
To check for new updates run: sudo apt update

Last login: Wed Jun 19 23:36:21 UTC 2024 on ttys0
holuser@ubuntu-001211:~$ cat /dev/urandom > /dev/null
① -
```

To trigger the alert in Aria Operations we need to cause the CPU to be busy. Not always an easy task in a lab environment. Fortunately, Linux has offers a few convenient ways to load a CPU. For this we will utilize /dev/urandom, a file that provides a continuos stream of pseudo random data when accessed. By reading this file and output the result to /dev/null we can generate 100% CPU load.

1. In the command prompt enter `cat /dev/urandom > /dev/null` <press enter>

The execution of this command will generate 100% CPU usage. Next we will return to Aria Automation and monitor the status of the deployment. Depending on the timing, it make take up to 15 minutes for the alert to trigger.

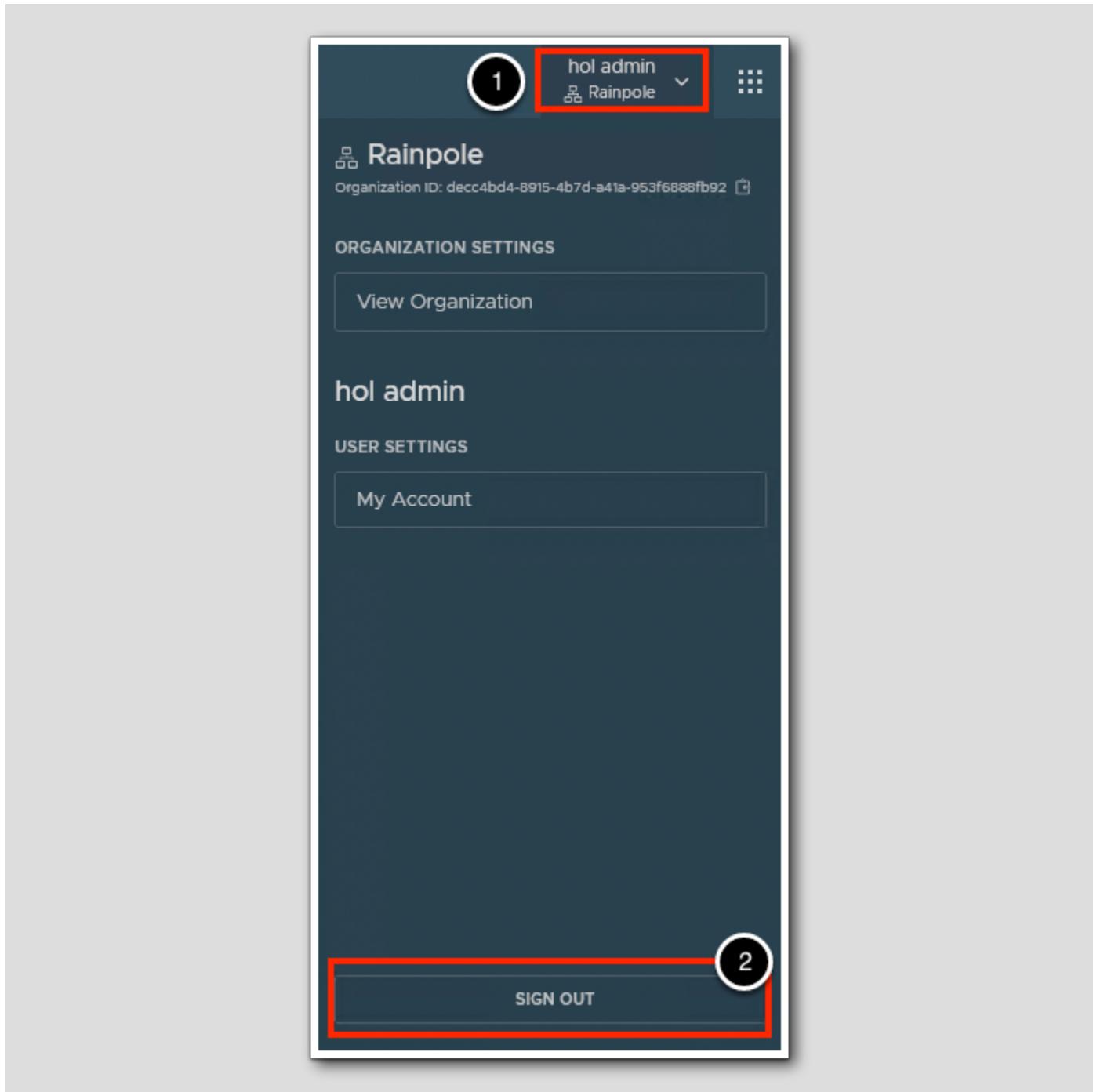
## Monitor Deployment

[321]

1. Return to Aria by clicking on the **Assembler** browser tab
2. Change to the **History** tab to watch for changes and updates to the deployment
  - The Monitor tab pulls performance data from Aria Operations, so you can also monitor the status of CPU consumption there.
3. Change to the **Request Details** tab to watch for changes to the **vmSize** input property
4. Click the **refresh icon** to refresh the page
  - The status should update once the Orchestrator workflow executes, pressing the refresh button can be cathartic while we wait!
5. Notice that once the update is in progress, the **vmSize** input change to **large**

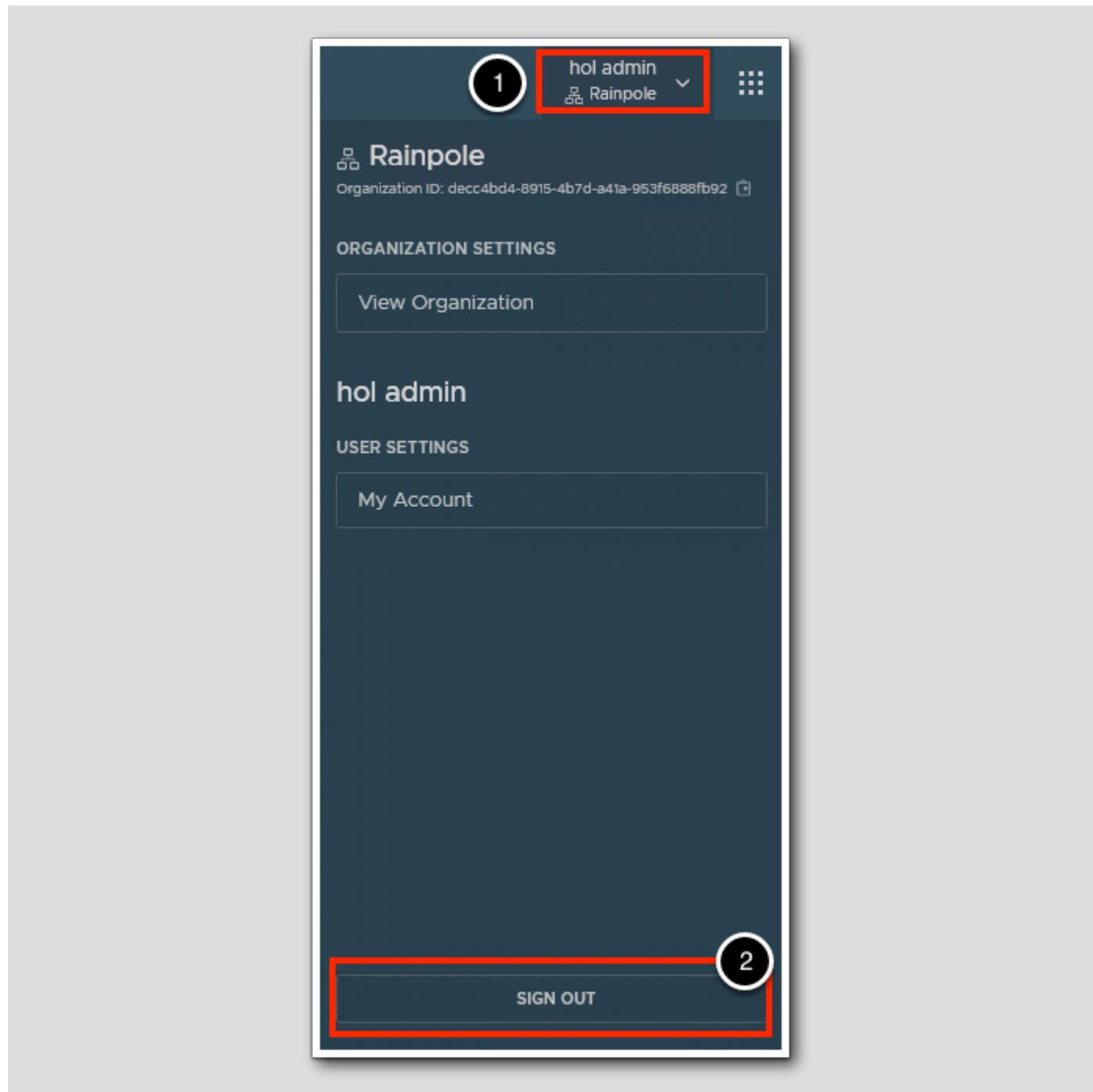
Congratulations on successfully integrating Aria Orchestrator and Aria Operations! Feel free to return to Aria Orchestrator to review the logs for the AutoScale-Out workflow, or wait a while and see what happens when the AutoScale-in workflow is executed due to low CPU usage on our VM. After all, it was rebooted as part of the update, killing our CPU loading command.

## Log Out of Aria Automation Orchestrator



To Log out of Aria Automation Orchestrator:

1. Click **hol admin** to open the organization menu.
2. Click **SIGN OUT**.



## Conclusion

[323]

Integrating Aria Orchestrator with Aria Operations to automate alert remediation through custom workflows provides a robust solution for maintaining operational efficiency and reliability in your vSphere environment. By following the steps outlined in this module, you

have learned how to set up this integration and create workflows that dynamically respond to CPU usage alerts. This automation minimizes manual intervention, reduces downtime, and ensures that your infrastructure is more resilient and adaptive to changing workloads.

To further expand your knowledge and capabilities with Aria Orchestrator and Aria Operations, please refer to the following resources:

- VMware Aria Operations Documentation
- VMware Aria Orchestrator Documentation
- VMware Marketplace - Aria Orchestrator Management Pack

By leveraging these resources, you can continue to optimize and customize your vSphere environment, ensuring a high level of operational efficiency and reliability. Thank you for participating in this module, and we encourage you to explore further integrations and automations to enhance your IT infrastructure.

## Module 6 - Enhance Lifecycle Management using Extensibility in Aria Automation (30 minutes) Intermediate

### Introduction

[325]

In this module, we will quickly cover the basics of using Aria Automation Orchestrator and Aria Automation to perform extensibility tasks.

The lab module will provide a brief overview of the following topics:

- Build a Custom Day 2 Action utilizing an Aria Automation Orchestrator Workflow

Lab Captain(s):

- Fred Hofer, Staff Technical Adoption Manager, Austria

### Log in to Aria Automation as holadmin

[326]

In this lesson we will start by logging into Aria Automation, and launching the Assembler service in preparation for creating our new Extensibility content.

### Open the Firefox Browser from Windows Quick Launch Task Bar

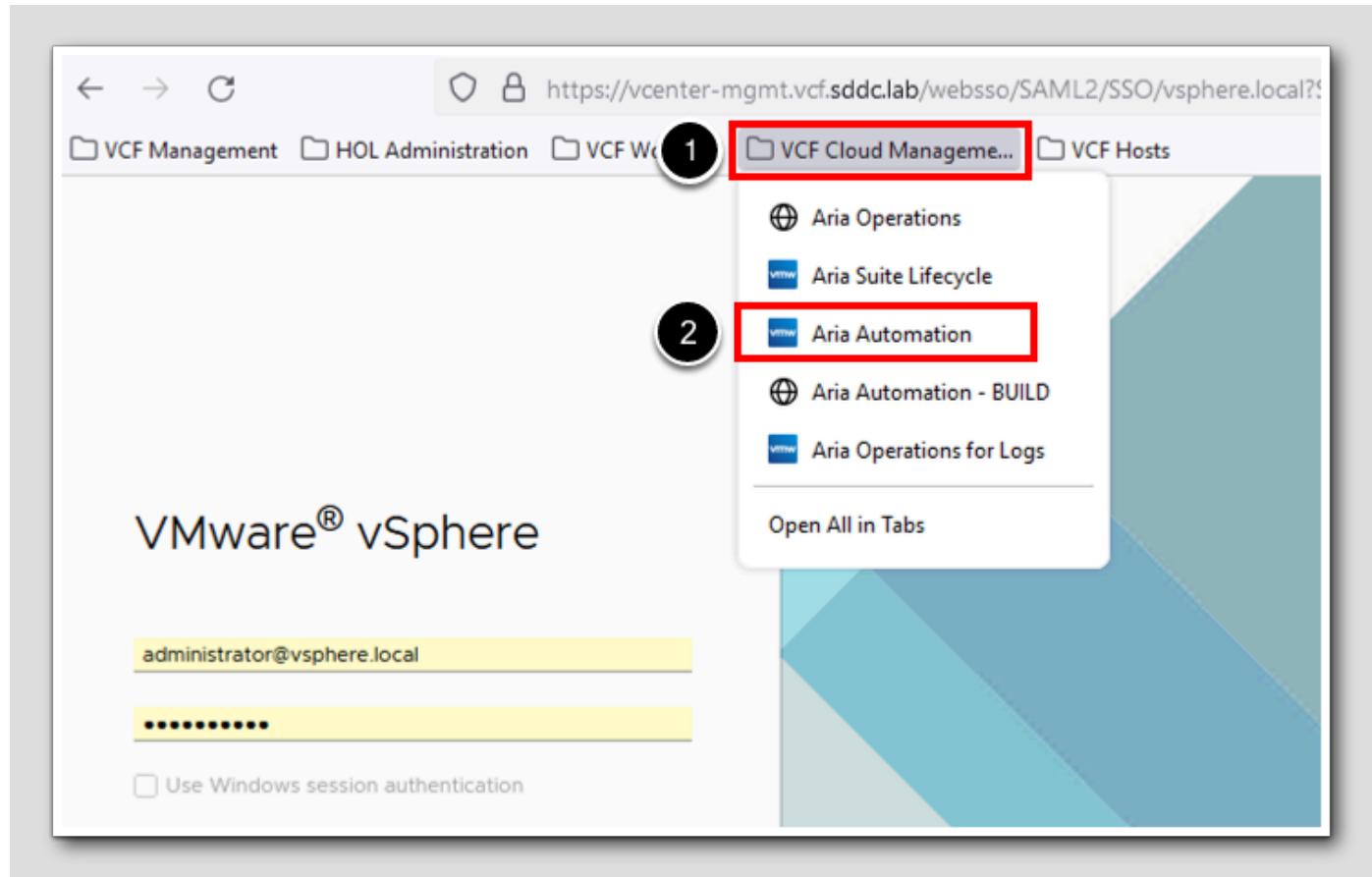
[327]



If the browser isn't already open, launch Mozilla Firefox.

1. Click the Firefox icon on the Windows Quick Launch Task Bar

Open the bookmark for Aria Automation



From within the Firefox web browser:

1. Click VCF Cloud Management from the bookmarks bar to open the folder
2. Click on Aria Automation

## Open the Login Page

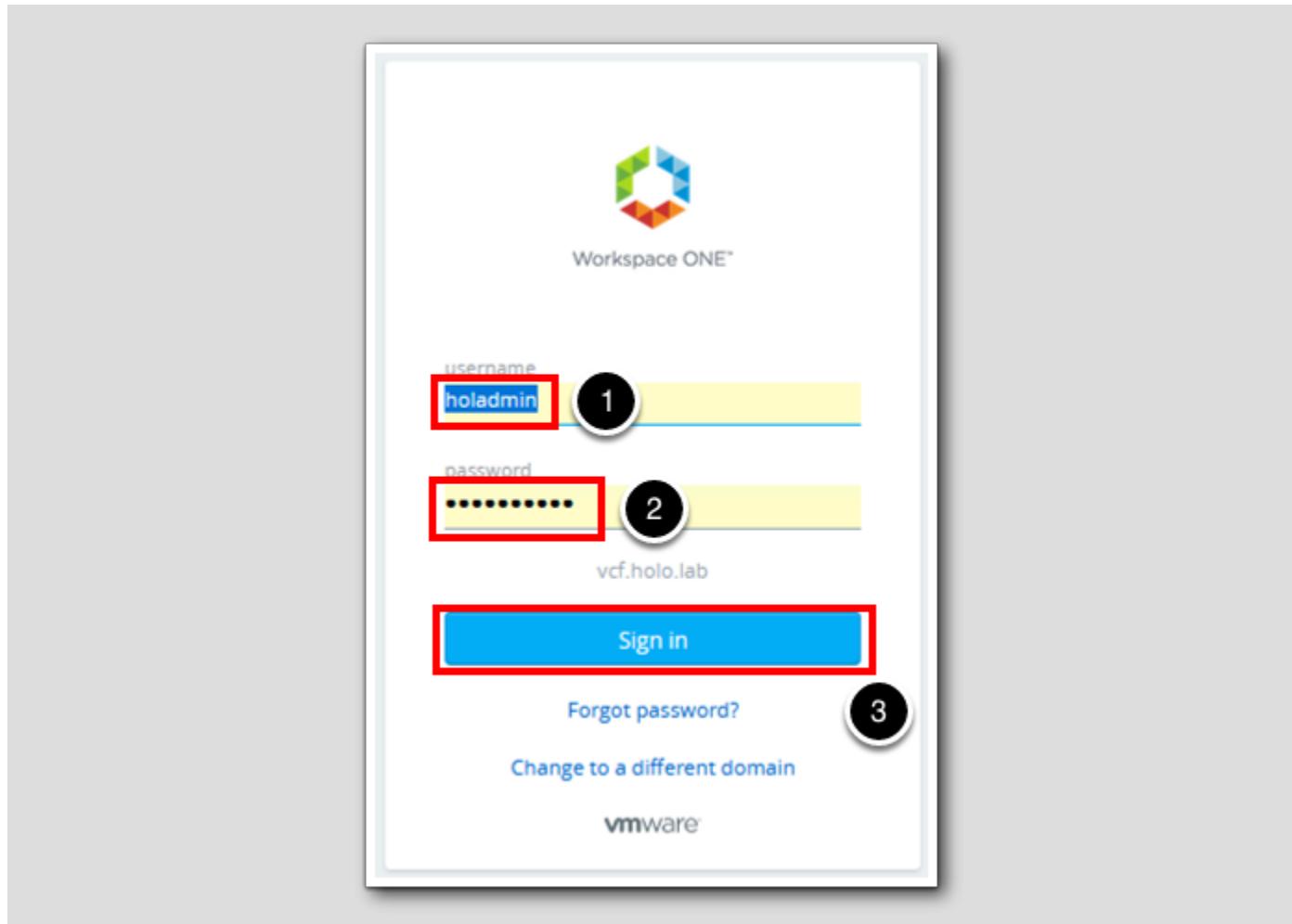
[329]



At the Aria Automation splash screen:

1. Click GO TO LOGIN PAGE

## Log in to Aria Automation

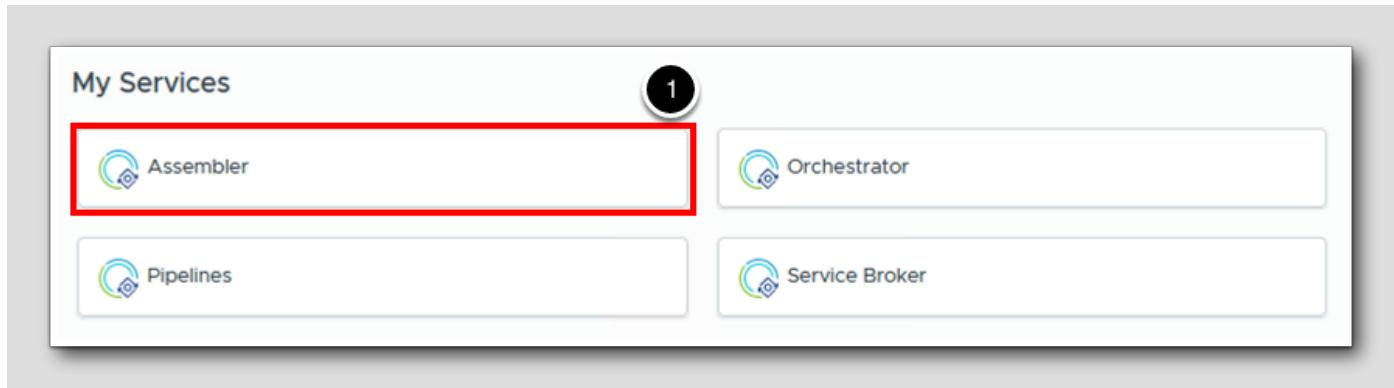


At the **Workspace ONE** login screen:

1. Enter **holadmin** into the username field
2. Enter **VMware123!** into the password field
3. Click **Sign In**

## Launch the Assembler Service

[331]



From within the Cloud Services Console, under **My Services**:

1. Click the **Assembler** service

## Custom Day 2 Action utilizing an Aria Automation Orchestrator Workflow

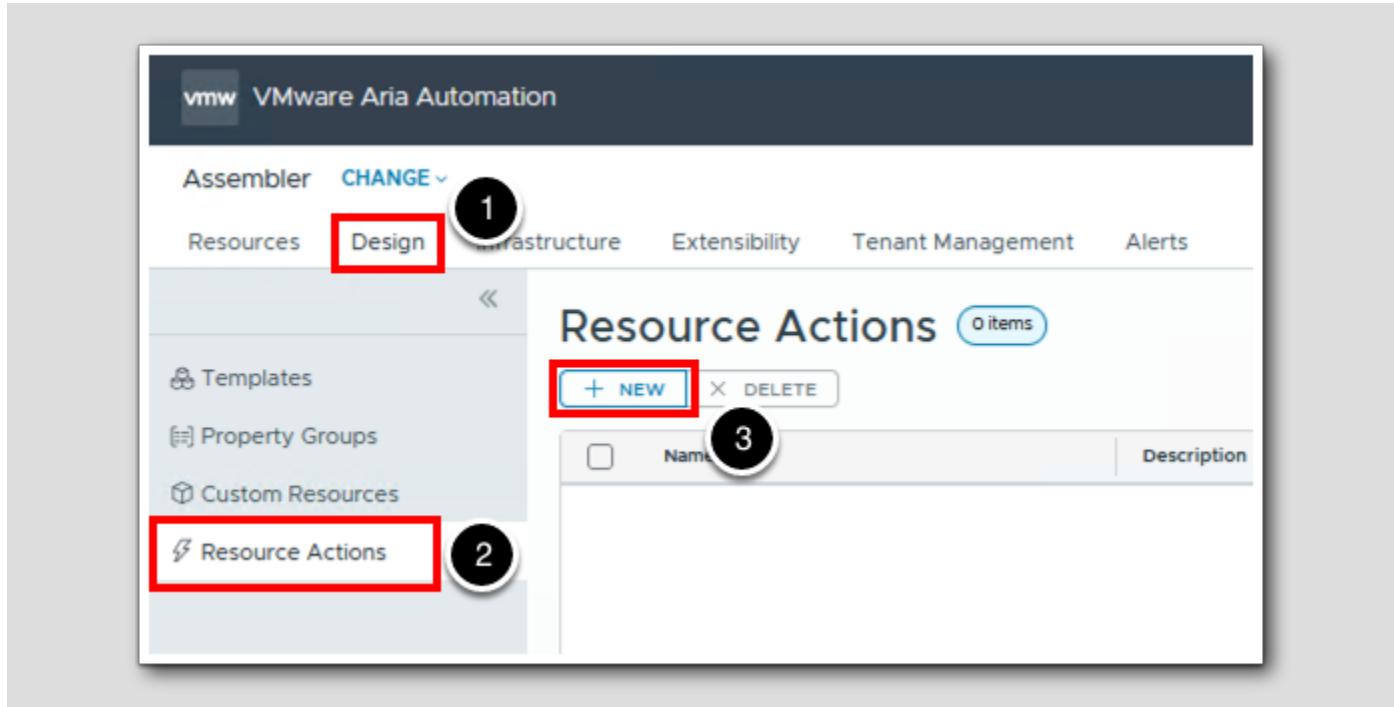
[332]

After we create a deployment based on a cloud template, we can run different lifecycle (day 2) actions on both the deployment and its resources. Aria Automation Assembler includes many Day 2 actions out of the box, but we might want to create others to meet different requirements. We can achieve this by creating custom Resource Actions.

Custom Resource Actions can be based either on Aria Automation Orchestrator workflows or Action Based Extensibility (ABX) actions and can be applied to either embedded or Custom Resources.

In this lesson we will create a custom Resource Action using Aria Automation Orchestrator to move a deployed vSphere Machine resource to a different vSphere folder.

## Navigate to Resource Actions



The first step we need to take is to get to the Resource Actions screen where we can start to create our new custom Resource Action:

1. Click on the Design Tab
2. Select Resource Actions
3. Select + NEW to begin creating our new Resource Action

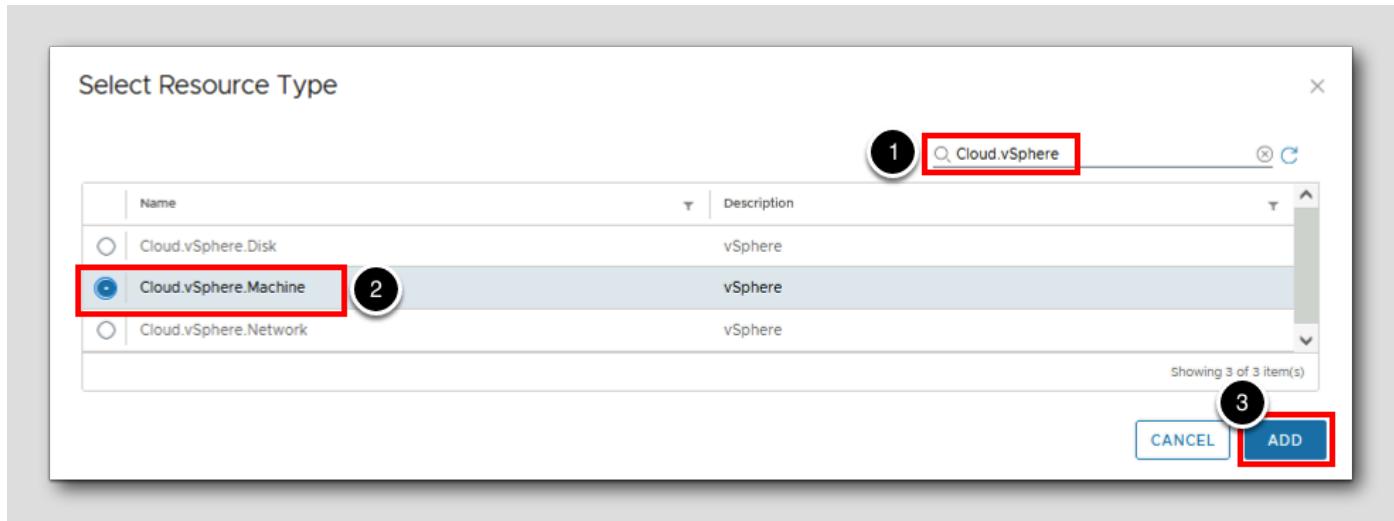
## Configure Resource Action

The screenshot shows the 'New Resource Action' configuration screen. The 'Name' field contains 'Move\_VM\_Folder' (step 1). The 'Display name' field contains 'HOL Move VM to New Folder' (step 2). The 'Activate' section has a slider turned on (step 3). The 'Scope' section shows 'Available for any project' with a note: 'Resource Action will be available for resource types in deployments in any project'. The 'Resource Type' section has a '+ ADD' button highlighted with a red box (step 4).

We will begin to enter the required information to create the Resource Action:

1. Enter **Move\_VM\_Folder** for the **Name** of our Resource Action
2. Enter **HOL Move VM to New Folder**, this will be the **Display name** for our Resource Action
3. Toggle the **Activate** slider so that we enable the Resource Action
4. Click **+ADD** to add the Resource Type that this action is associated with

## Select the Resource Type

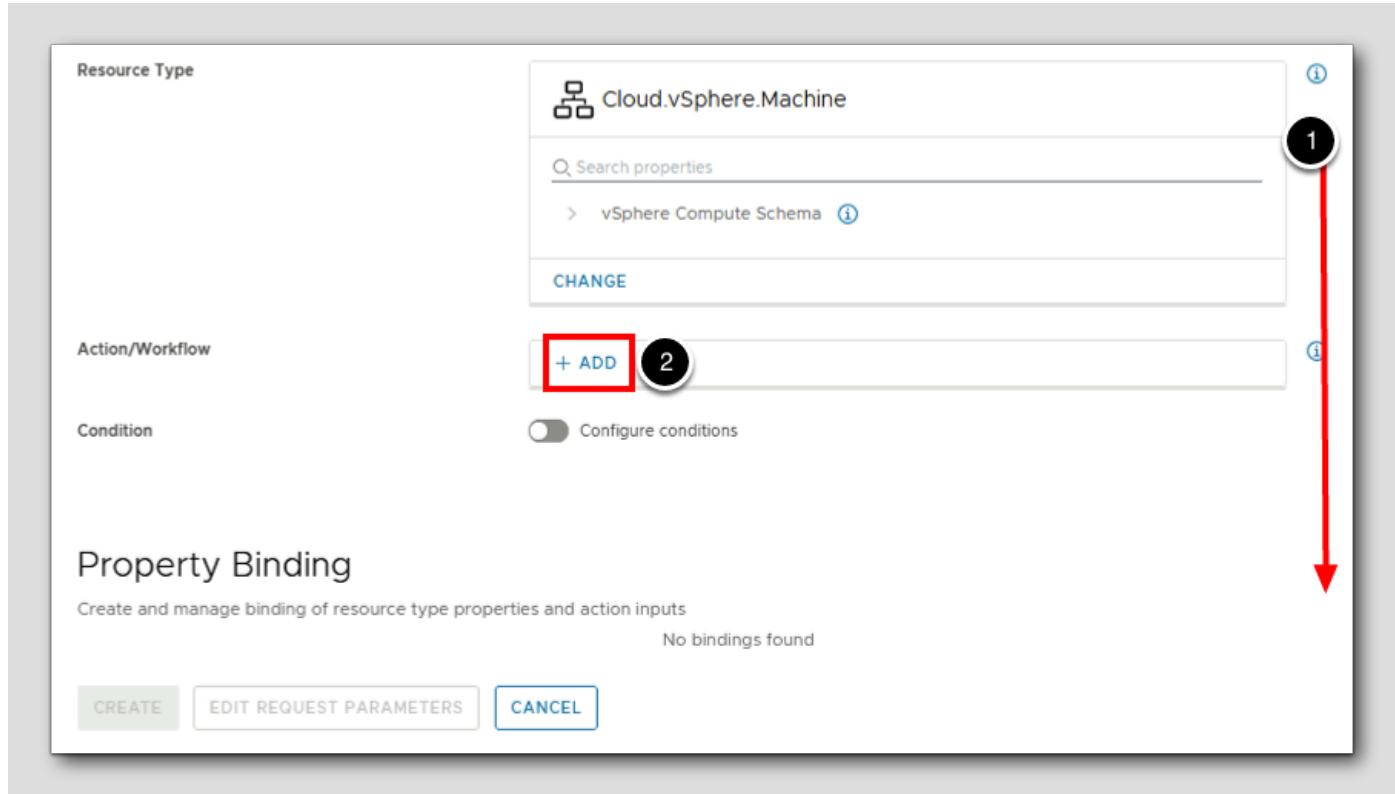


As this Resource action will be for a vSphere Machine we will need to specify this for our Resource Type:

1. Type **Cloud.vSphere** in the **search bar** and then press **Enter** to start the search
2. Select **Cloud.vSphere.Machine** from the search results
3. Click **ADD**

**Note:** A single Resource Action can only be associated to one resource type. However, the same Aria Automation Orchestrator workflow or ABX action can be used in multiple Resource Actions.

## Adding a Workflow

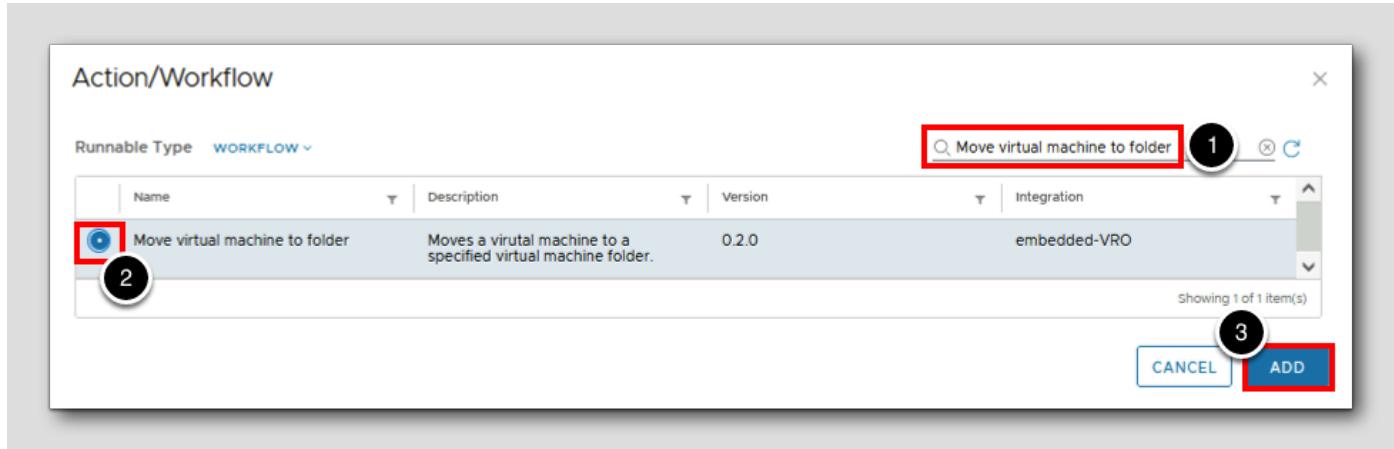


Now that we have selected our Resource Type we now need to select the workflow that we would like to run against it:

1. Scroll down if needed to view the rest of the form
2. Click +ADD in Action/Workflow to add a workflow

**Note:** Remember that Resource Actions can be associated with Aria Automation Orchestrator workflows or ABX Actions, however only a single workflow or action can be selected for each Resource Action. We also cannot combine a workflow and action together within a single Resource Action as we will see in the next step.

## Select the Workflow



Through the out of the box vSphere integration, Aria Automation Orchestrator comes with an existing workflow to move virtual machines to a folder. This means we do not need to create our own:

1. In the search field enter **Move virtual machine to folder**
2. Select the **Move virtual machine to folder** workflow from the search results
3. Click **ADD** to add it to our Resource Action

## Edit the Property Bindings

The screenshot shows a 'Property Binding' configuration screen. At the top, there's a 'CHANGE' button and a 'Condition' section with a toggle switch labeled 'Configure conditions'. A red vertical arrow on the right points downwards from the top of the table area.

**Property Binding**  
Create and manage binding of resource type properties and action inputs

| Action input | Data type         | Binding                    | Description             |
|--------------|-------------------|----------------------------|-------------------------|
| folder       | VC:VmFolder       | <a href="#">in request</a> | Destination folder      |
| vm           | VC:VirtualMachine | <a href="#">in request</a> | Virtual machine to move |

2 objects

At the bottom are three buttons: 'CREATE' (blue), 'EDIT REQUEST PARAMETERS' (light blue), and 'CANCEL' (light gray).

- 1: Points to the 'Configure conditions' toggle switch.
- 2: Points to the 'Action input' column header of the table.
- 3: Points to the 'Binding' column header of the table.

The Property Bindings section on the request form is now populated detailing the Property Bindings.

We use the Property Binding feature to ensure the corresponding workflow or action receives the correct inputs (either manually or automatically) to be able to execute successfully.

There are three Property Binding types:

- **In request** which basically means the value will be set at request time via an interactive dialog
- **Direct** which basically allows us to define the property (schema) of the resource to pass as a binding
- **With binding action** allows us to choose a binding Aria Automation Orchestrator action to run to bind to the Data Type

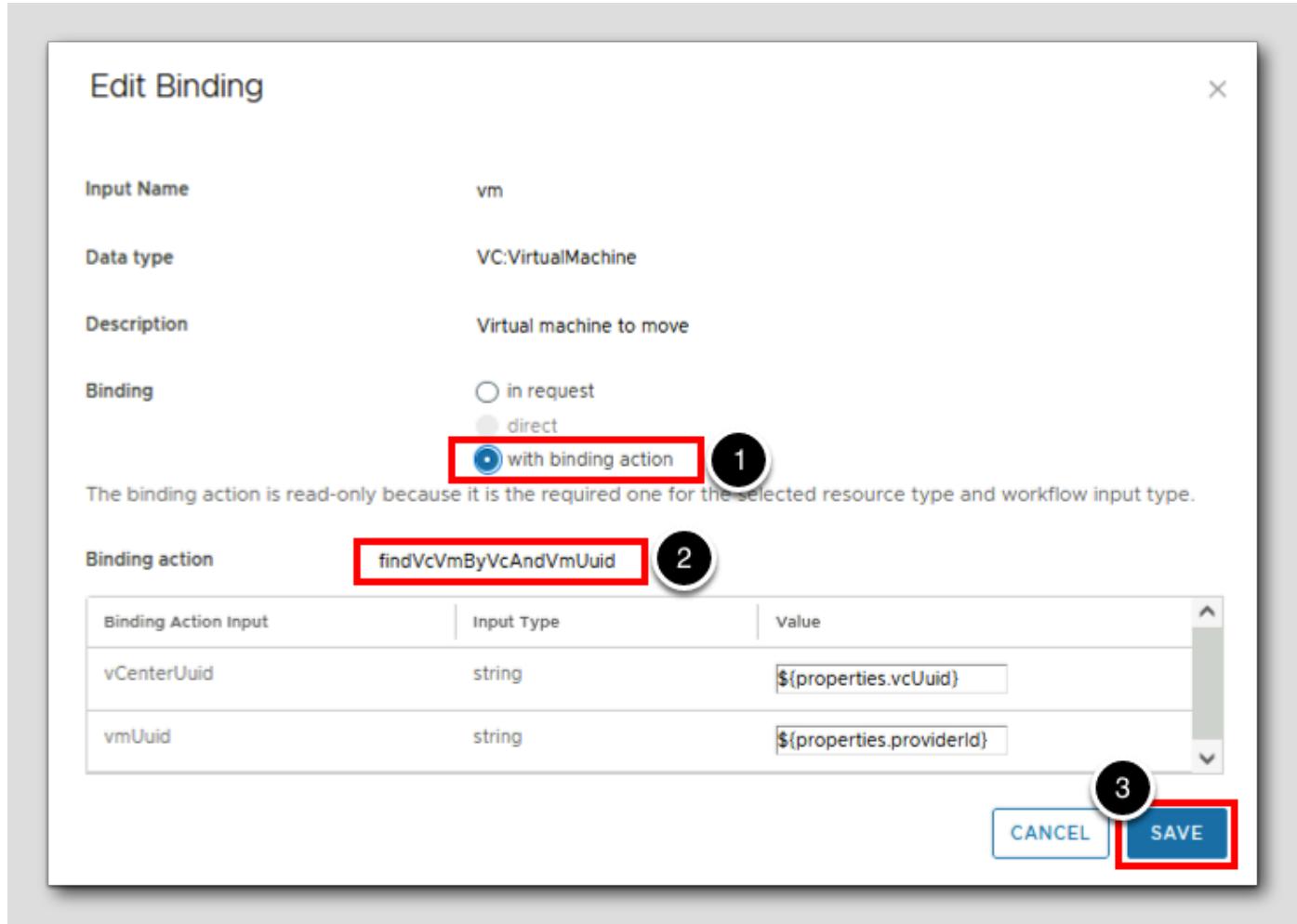
As there are two Property Bindings listed, the workflow requires two inputs:

- **vm** which is of Aria Automation Orchestrator data type, VC:VirtualMachine
- **folder** which is of Aria Automation Orchestrator data type, VC:vmFolder

As this Resource Action is related to the **Cloud.vSphere.Machine** Resource Type, and we need to translate that to the **VC:VirtualMachine** type, we are going to update the Property Binding to use a **binding action** to automatically provide the right resource information for the resource the action is being completed on:

1. Scroll down to view the rest of the form
2. Locate the **vm** Action input column
3. Click **in request** in the Binding column

## Set up the Property Binding



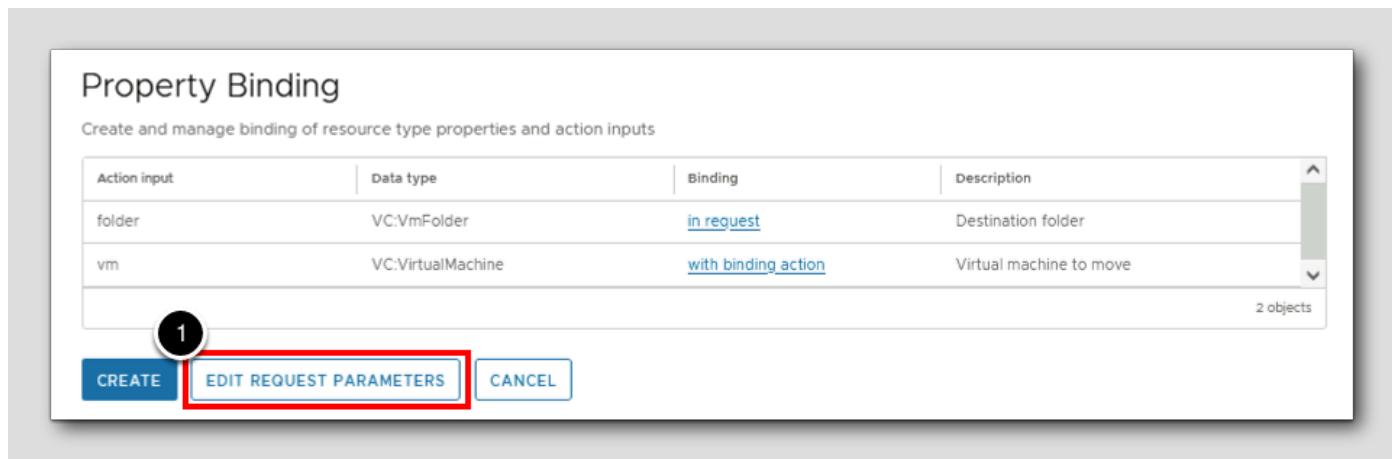
We now need to select the appropriate binding action to complete the mapping of our Virtual Machine types:

1. In Binding select the with binding action radio button
2. The findVcVmByVcAndVmUuidaction is automatically selected
3. Click SAVE

We observed that the `findVcVmByVcAndVmUuid` action was automatically selected and configured. This is due to the fact that when running the embedded version of Aria Automation Orchestrator, the integration with Aria Automation Assembler is automatically configured.

## Edit the Request Parameters

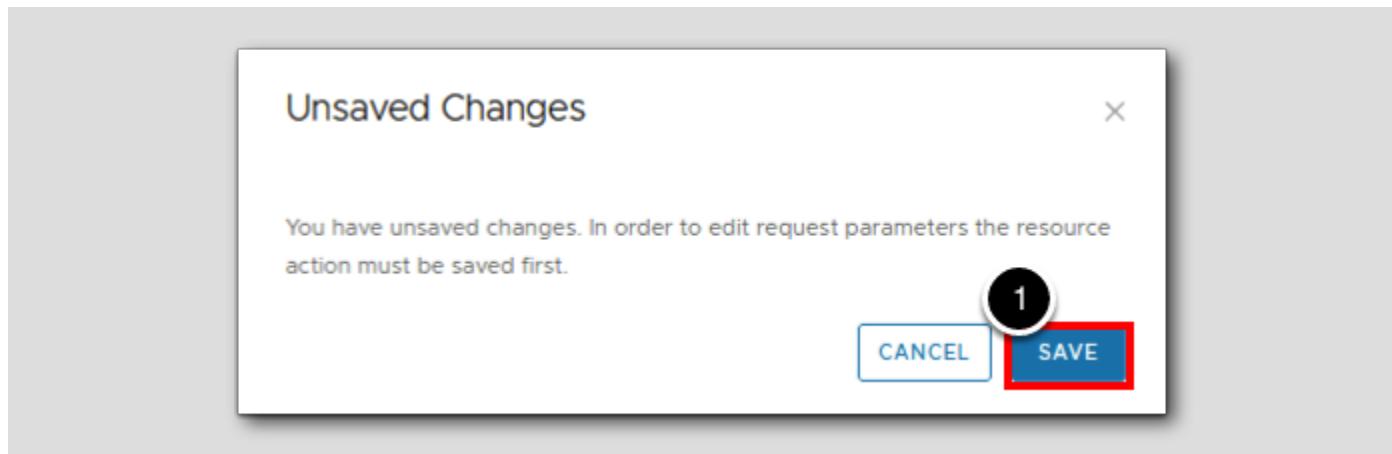
[340]



1. Click EDIT REQUEST PARAMETERS

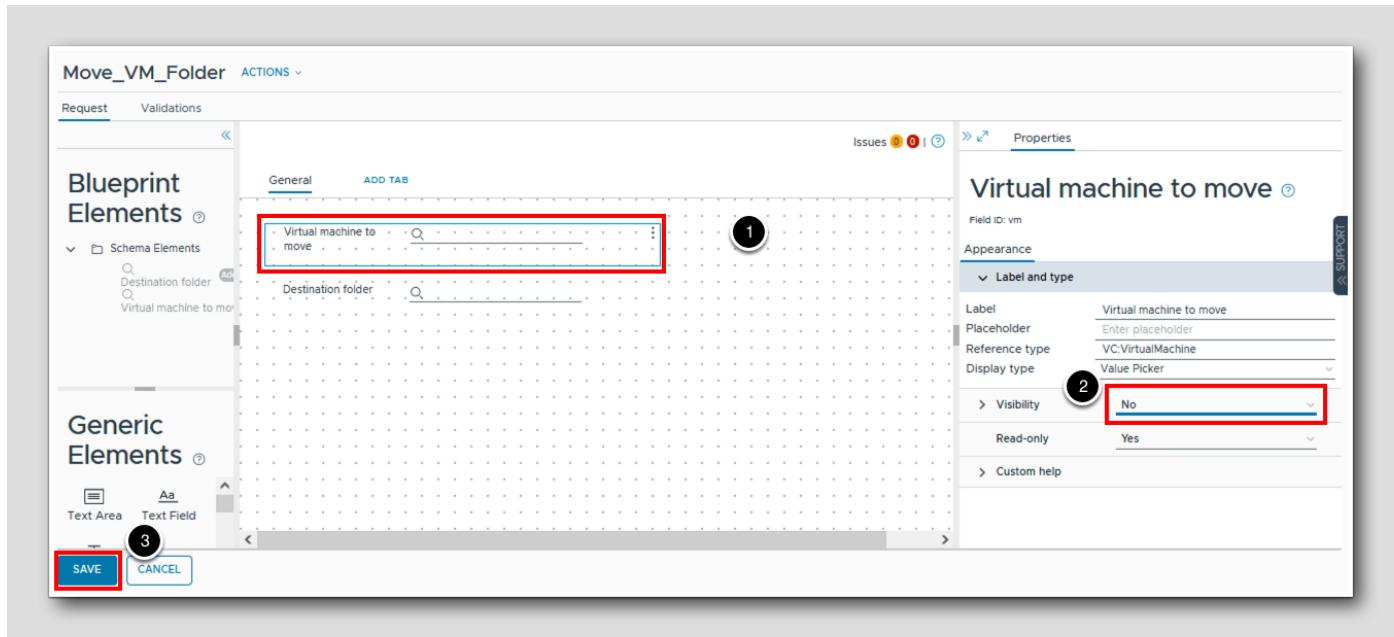
## Save Changes

[341]



1. Click SAVE to save the Resource Action before we edit the Request Parameters form

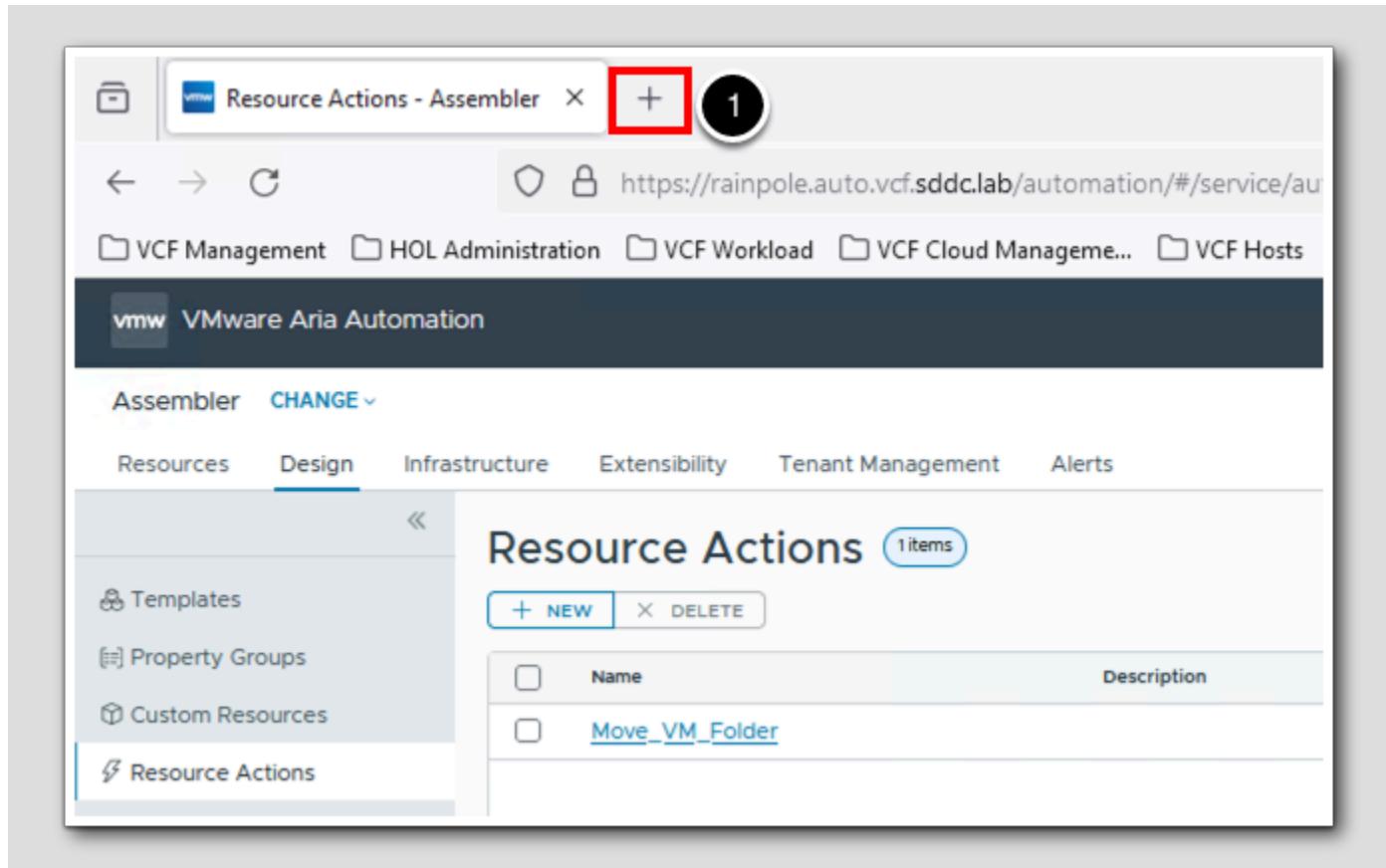
## Update the Request Form



Even though we have configured a binding action, the field for the Virtual Machine object will still be displayed on the form at request time. We can customise the request form for the Resource Action, so that the Virtual Machine field is not displayed:

1. Select the Virtual machine to move field on the Form
2. Set Visibility to No
3. Click SAVE to save the changes

Open a new Browser Tab



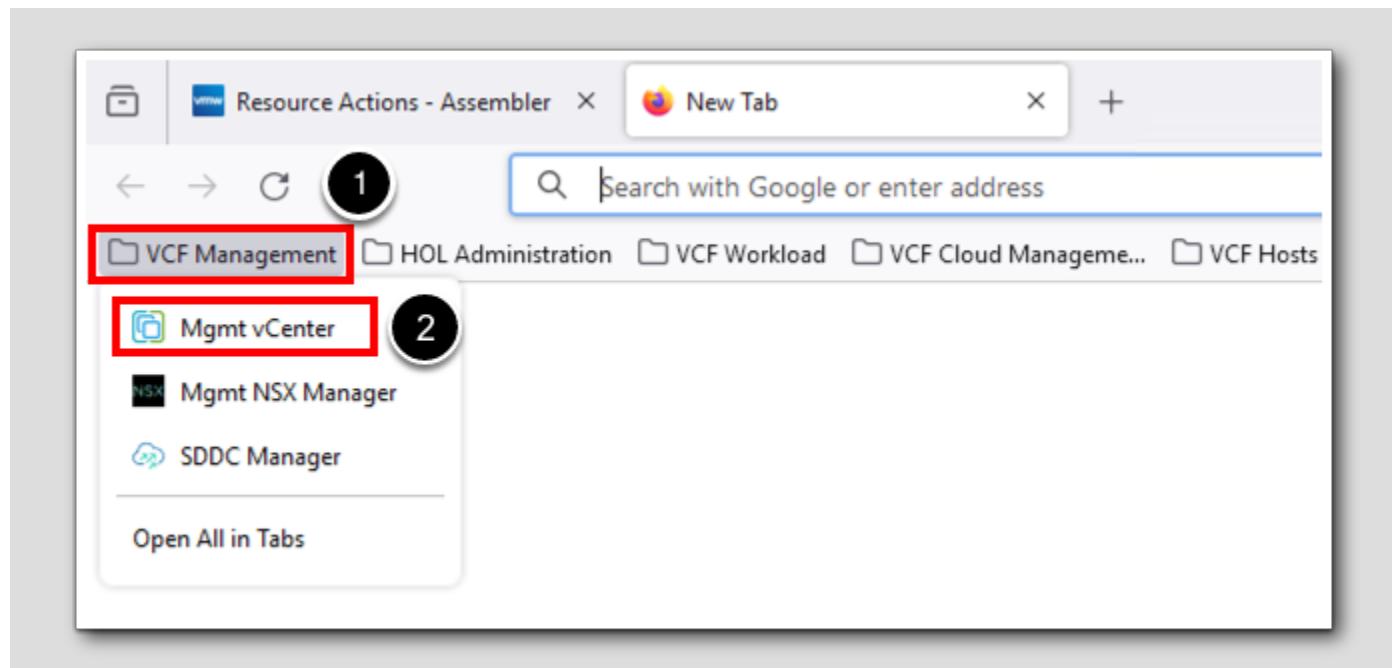
The screenshot shows a browser window titled "Resource Actions - Assembler". The address bar shows the URL: <https://rainpole.auto.vcf.sddc.lab/automation/#/service/au>. The page content is the "Resource Actions" section of the VMware Aria Automation interface. The "+" button in the top right corner is highlighted with a red box. The "Design" tab is selected. The "Resource Actions" table lists one item: "Move\_VM\_Folder".

| Name           | Description |
|----------------|-------------|
| Move_VM_Folder |             |

Let's now open a new tab in the Firefox browser so we can open the vSphere client in preparation for testing our new action:

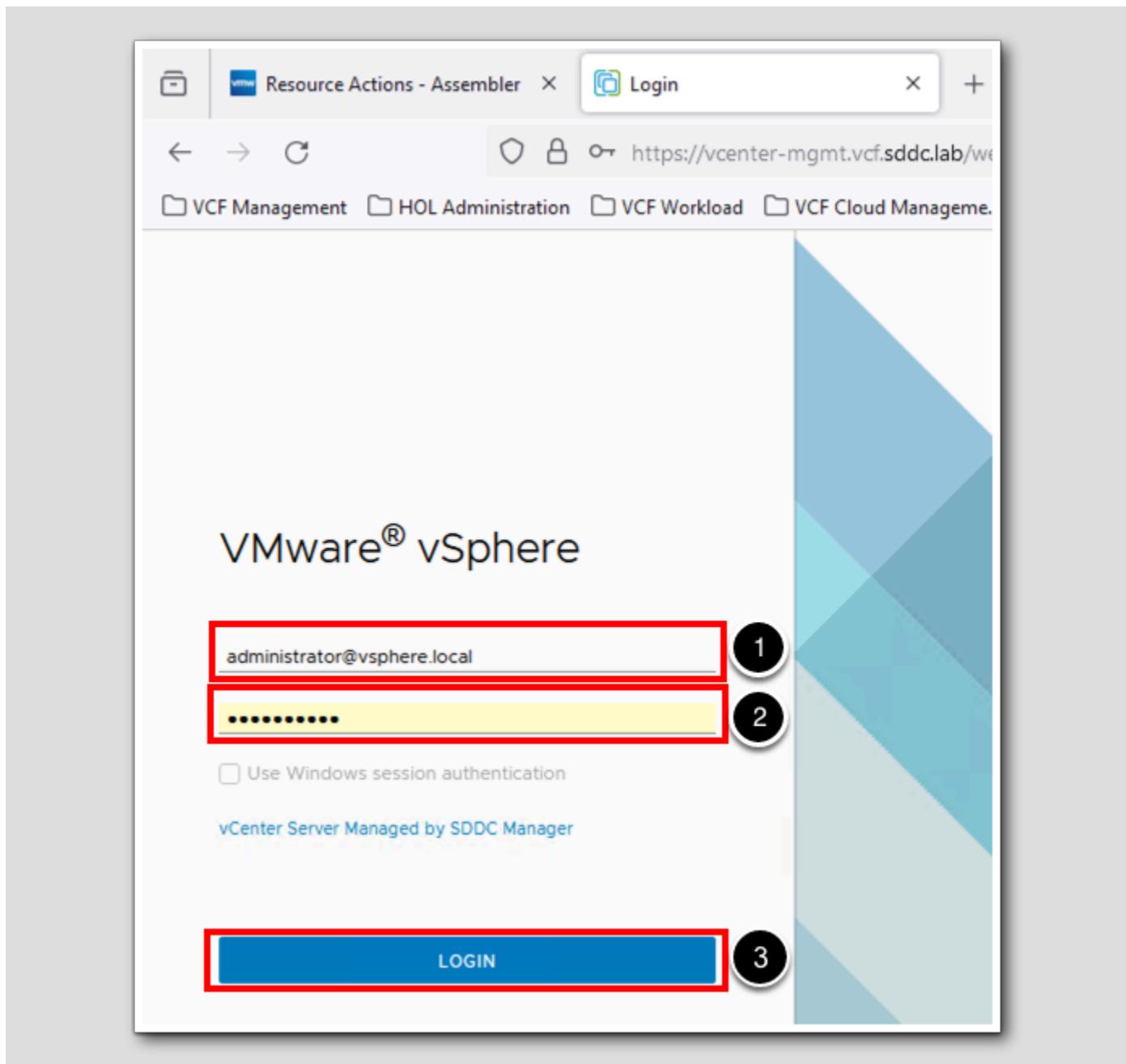
1. Click the + button in Firefox to open a new tab

Open the vSphere Client login screen



1. Click VCF Management bookmark to open the folder
2. Open the vSphere Client by selecting Mgmt vCenter

## Log in to the vSphere Client



1. Enter administrator@vsphere.local into the username field
2. Enter VMware123! into the password field
3. Click LOGIN

## View the Virtual Machine

The screenshot shows the vSphere Client interface with the following highlights:

- Resource Actions - Assembler**: A browser tab at the top left, highlighted with a red box.
- vSphere - vcenter-mgmt.vcf.sddc.lab**: The current browser tab, highlighted with a red box.
- 4**: A number in a circle indicating the count of open tabs.
- 1**: A number in a circle next to the Datacenters icon in the navigation bar.
- 2**: A number in a circle next to the Workloads icon in the navigation bar.
- 3**: A number in a circle next to the selected virtual machine, **ubuntu-000308**.

The main pane displays the **vccenter-mgmt.vcf.sddc.lab** host under the **Summary** tab. An alarm titled **esxi-2.vcf.sddc.lab Host memory usage** is visible in the Issues and Alarms section.

1. Select the VM's and templates view

2. Expand the Workloads Folder

3. Make note of the **ubuntu-xxxxxx** virtual machine, this virtual machine was pre deployed for use in the lab by Aria Automation.

Will will use our Custom Day 2 action to move the **ubuntu-xxxxxx** virtual machine to the **Development** folder

4. Return to the Assembler tab

Due to the naming policies within Aria Automation, the value of XXXX is a number from 000001 to 999999, that may differ for your lab environment. Therefore, the machine name in the screenshot may differ from the live lab environment presented to you.

## Run the Custom Resource Action

[347]

1. Select Resources to find our **hol-linux** deployment

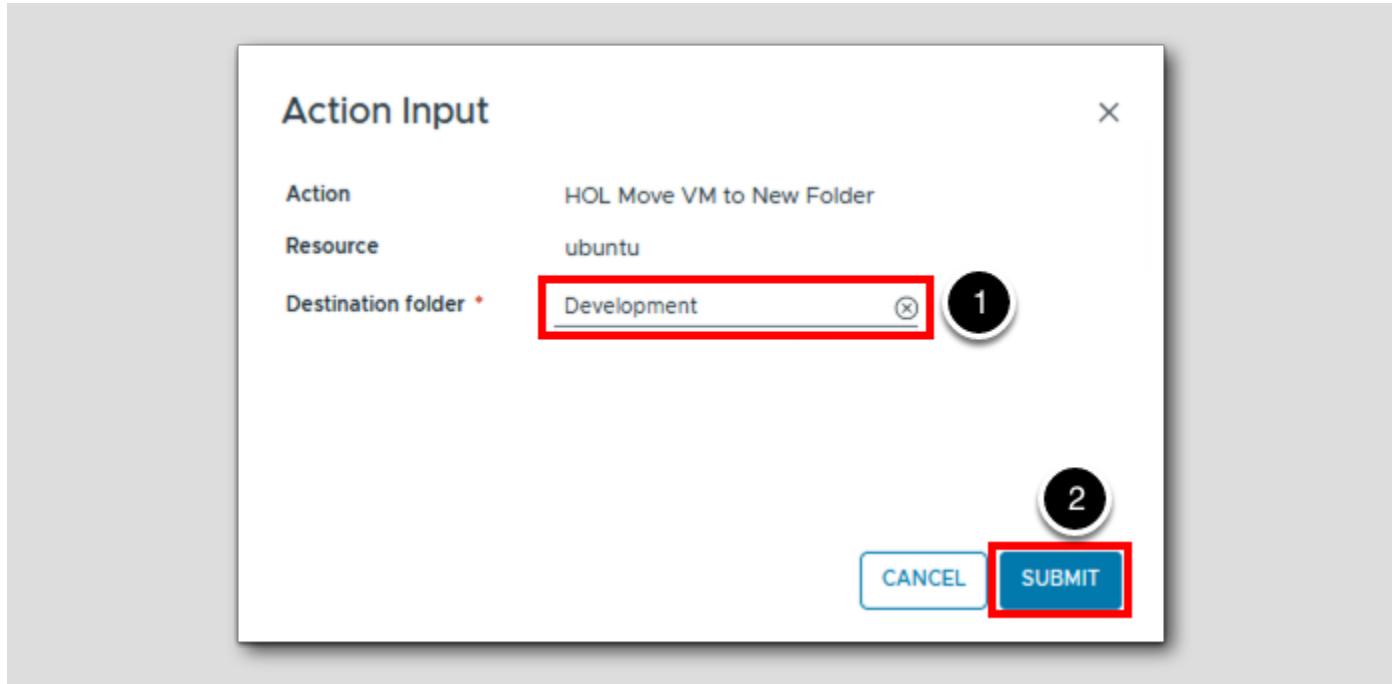
2. Expand the **hol-linux** deployment

3. Click the vertical ellipsis next to the **ubuntu-xxxxxx** machine to view the list of Resource Actions

4. Click on the scroll bar to the right of the actions list to scroll down

5. Notice that our Custom Resource Action appears in the list. Select **HOL Move VM to New Folder**

Provide the Request details



Notice that our request form does not have an input for virtual machine name, this is because we stated visibility = no when customizing the form for that field:

1. In the Destination folder field, type **Development** and then select Development from the drop down list
2. Click **SUBMIT**

## Observe the Status of the Action

The screenshot shows the VMware Aria Automation Assembler interface. On the left, there's a sidebar with categories like Deployments, All Resources, Virtual Machines, Volumes, and Networking & Security. The main area is titled "Deployments" and shows two items: "hol-windows" and "hol-linux". On the right, detailed information for the first item is displayed, including Owner (holadmin@vcf.holo.lab), Project (HOL Project), and a red-highlighted "Status" field which is currently "CANCEL". Below this, there's a table with columns for Resource Name, Address, Status, and Resource Type, containing entries for "ubuntu-000308" and "hol-prod".

1. The action will now run and the deployment screen will automatically update with a **Status** field showing the progress of the action.

## Wait for the Action to Complete

[350]

The screenshot shows the VMware Aria Automation Assembler interface. On the left, there's a sidebar with categories like Deployments, All Resources, Virtual Machines, Volumes, and Networking & Security. The main area is titled "Deployments" and shows two items: "hol-windows" and "hol-linux". To the right of the list is a detailed view of the first item, "hol-windows". The "Status" field is highlighted with a red box and contains a green banner with the text "HOL Move VM to New Folder Successful". There's also a small circular badge with the number "1" in the top right corner of the status box. Below the status are fields for Owner (holadmin@vcf.holo.lab), Project (HOL Project), Price (Month to date \$5.06), Expires on (Never), and Created on (Jul 19, 2024, 1:16:30 PM). At the bottom, there's a table with columns for Resource Name, Address, and Status, showing entries for "ubuntu-000308" and "hol-prod".

1. Wait while the action runs, the **Status** field will automatically update with a green banner and a value of **HOL Move VM to New Folder Successful**

Confirm the Request was successful

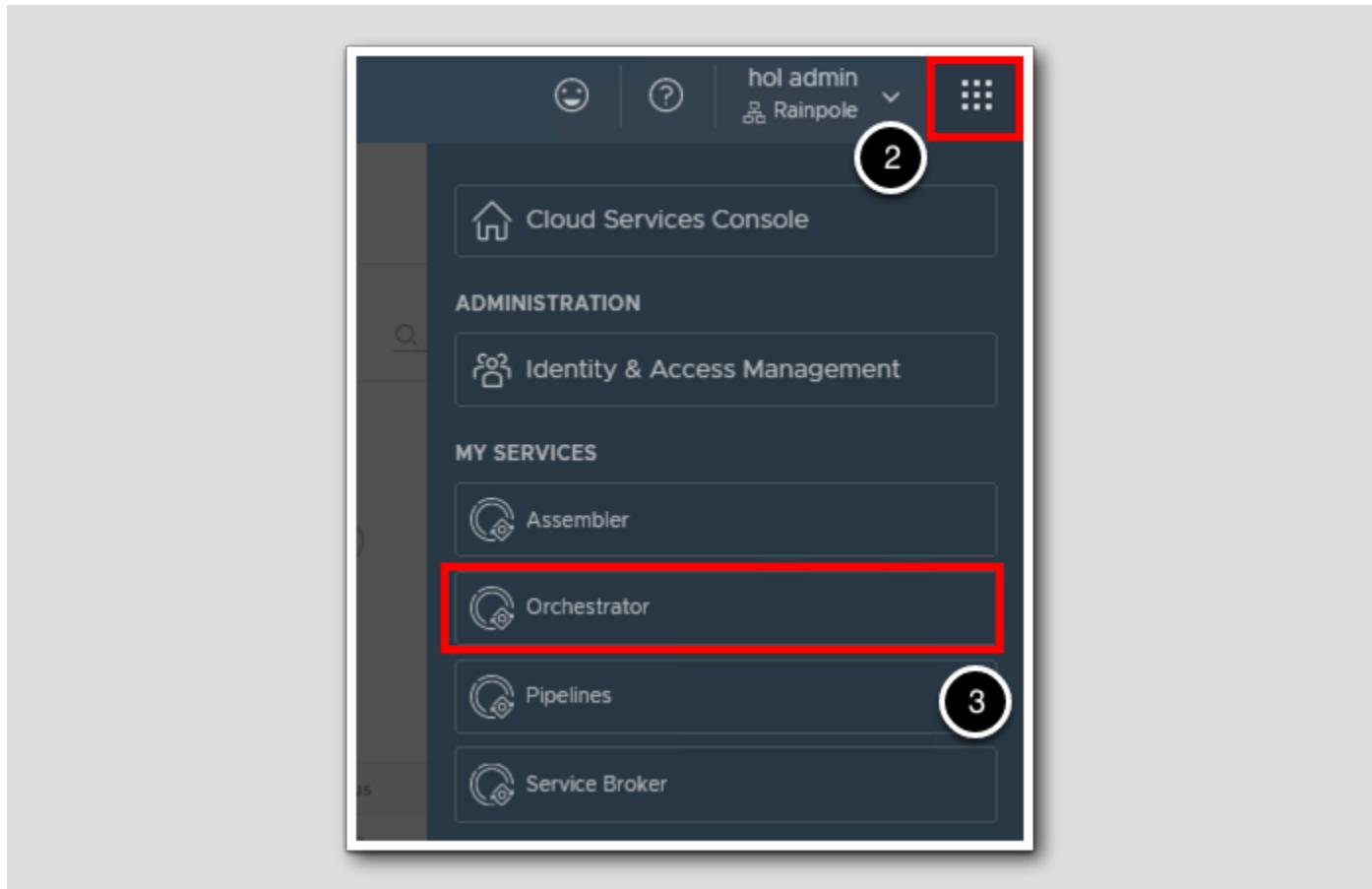
The screenshot shows the vSphere Client interface. At the top, there are two tabs: 'Deployments - Assembler' and 'vSphere - Development - Summary'. The 'vSphere - Development - Summary' tab is highlighted with a red box and has a circled '1' above it. Below the tabs is a navigation bar with icons for back, forward, search, and a URL field containing 'https://vcenter-mgmt.vcf.sddc.lab/ui/'. Underneath the navigation bar is a list of environment tabs: 'VCF Management', 'HOL Administration', 'VCF Workload', and 'VCF Cloud Manageme..'. A search bar follows, with the placeholder 'Search in all environments'. The main content area is titled 'Development' with an 'ACTIONS' button. It has four tabs: 'Summary' (which is selected and highlighted in blue), 'Monitor', 'Configure', and 'Permissions'. On the left, a tree view shows the hierarchy: 'vcenter-mgmt.vcf.sddc.lab' > 'mgmt-datacenter-01' > 'Development'. The 'Development' folder is also highlighted with a red box and has a circled '2' to its right. Inside the 'Development' folder, three virtual machines are listed: 'linux-dev-0010', 'linux-dev-0011', and 'ubuntu-000308'. Below the 'Development' folder, other items like 'Discovered virtual machine', 'mgmt-domain-fd-edge', etc., are listed. At the bottom of the content area is a 'Custom Attributes' section.

Let's now check that the workflow associated with the action successfully moved the virtual machine into the correct folder:

1. Return to the vSphere tab
2. Notice that our ubuntu-xxxxxx machine has moved to the Development folder

## Open Aria Automation Orchestrator

[352]



Before we complete this lesson, we are going to view the Aria Automation Orchestrator workflow run within the Aria Automation Orchestrator interface:

1. [not shown] Click the **Assembler** tab in Firefox
2. Select the **9 dots** in the top right hand corner
3. Select **Orchestrator**

## Confirm that the Workflow was run in Aria Automation Orchestrator

The screenshot shows the VMware Aria Automation Orchestrator interface. On the left, there's a sidebar with options like Dashboard, Library, Workflows, Actions, Policies, Activity, and Workflow Runs. The 'Activity' option is highlighted with a red box and has a circled '1' above it. In the center, there's a 'Usage' panel with a 'Completed 100%' status and a green circular progress bar. To the right, there's a 'Recent Workflow Runs' section with a table:

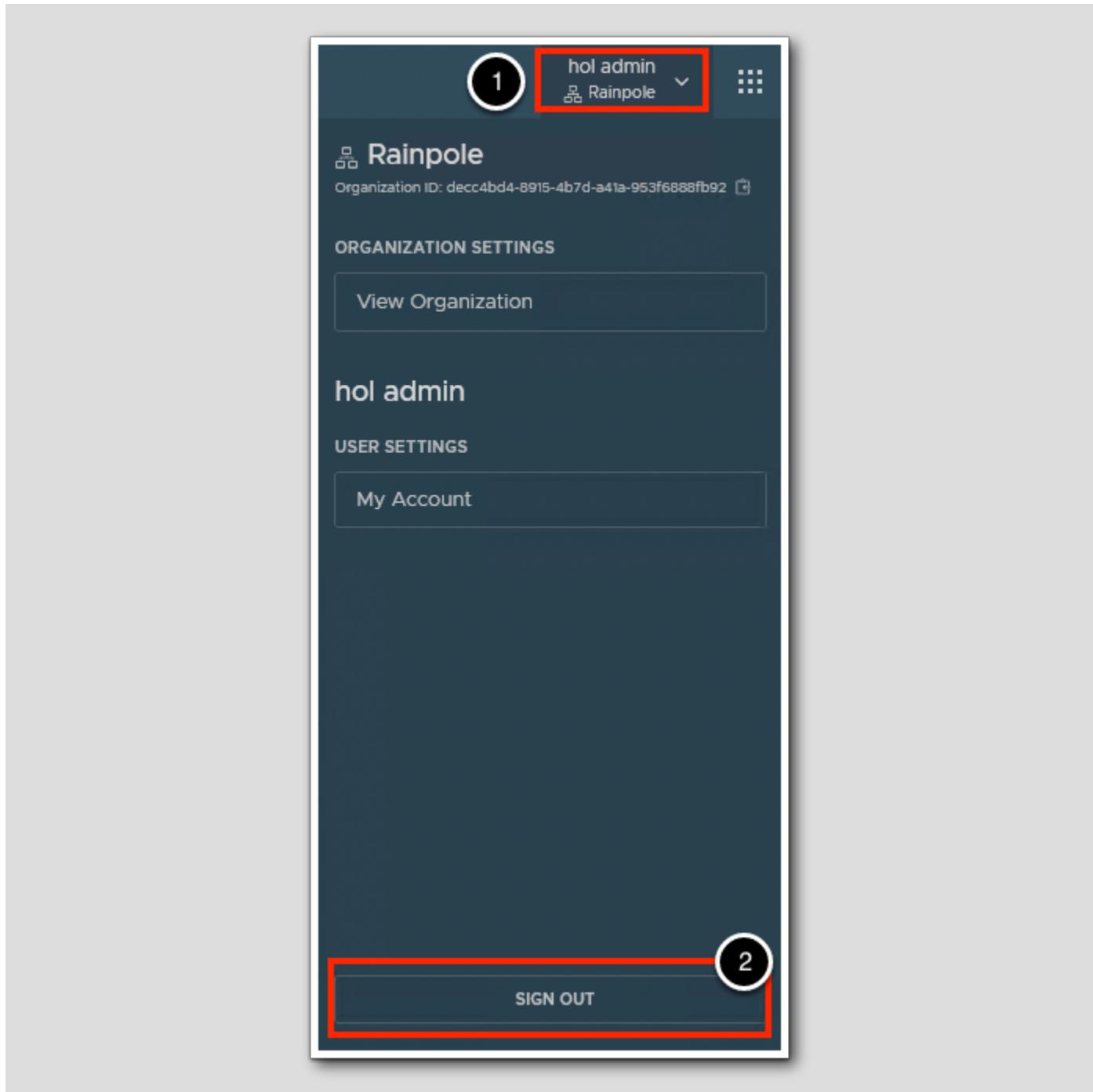
| Name   | Status    | Start Date              |
|--|-----------|-------------------------|
| <a href="#">Move virtual machine to folder</a>   | Completed | Jul 29, 2024 8:22:56 AM |
| <a href="#">Update a vCenter Server instance</a> | Completed | Jul 7, 2024 4:05:49 AM  |
| <a href="#">Add a vCenter Server instance</a>    | Completed | Jul 7, 2024 3:52:11 AM  |

Three specific rows in the table are highlighted with red boxes and have circled numbers above them: 'Move virtual machine to folder' (circled '2'), 'Update a vCenter Server instance' (circled '3'), and 'Add a vCenter Server instance'.

1. Click Dashboard in the left pane
2. Ensure that the Usage panel is selected
3. Note in the Recent Workflow Runs section you will see that the workflow Move virtual machine to folder was completed

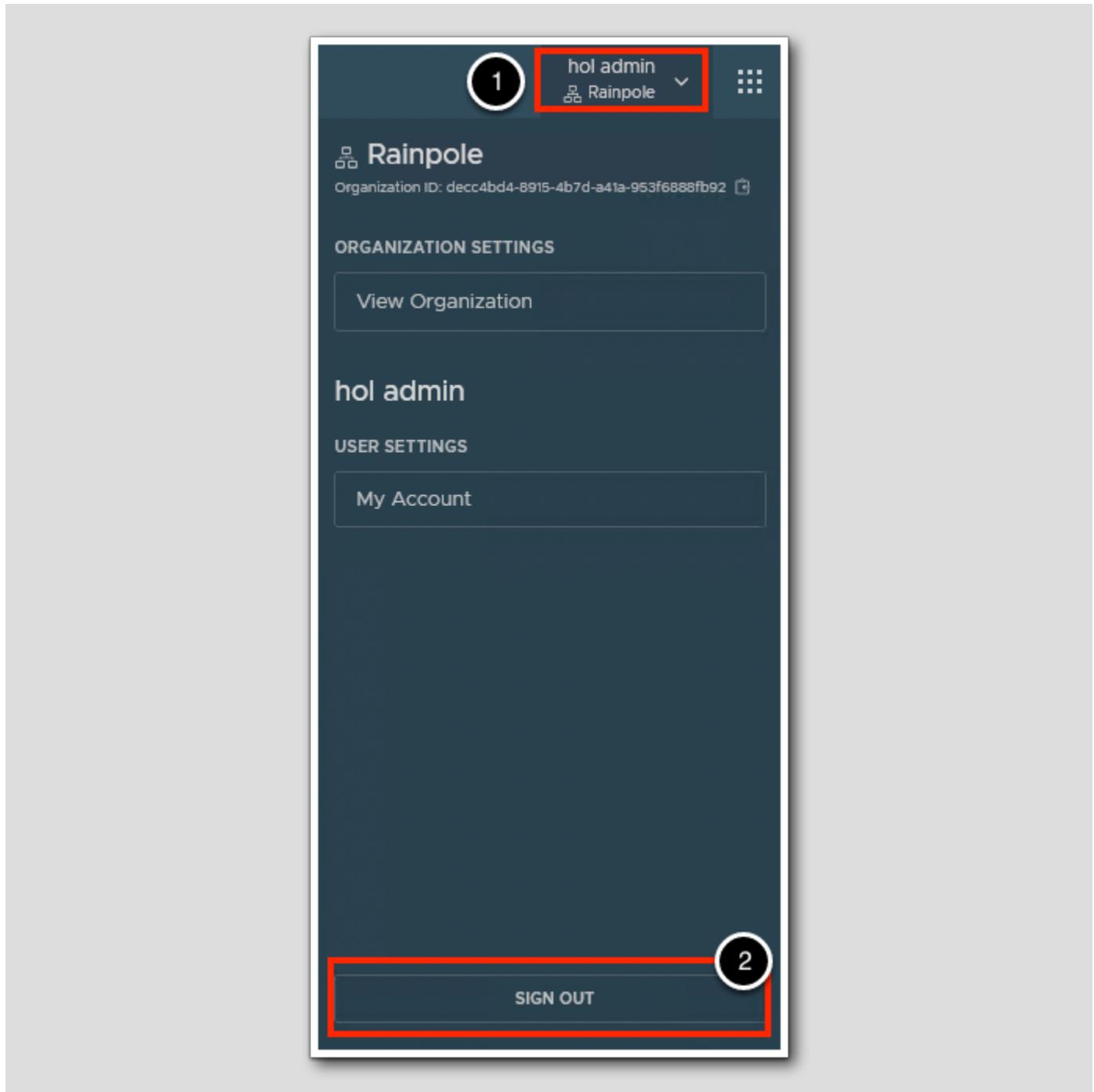
In this lesson we have successfully created a custom Resource Action which calls an Aria Automation Orchestrator Workflow to complete a Day 2 operational task on a virtual machine resource. In this example we moved a virtual machine to a new folder in VMware vCenter but the possibilities are only limited by our imagination.

## Log Out of Aria Automation Orchestrator



To Log out of Aria Automation Orchestrator:

1. Click **hol admin** to open the organization menu.
2. Click **SIGN OUT**.



## Conclusion

[355]

In this module, we walked through the basics of using Aria Automation Orchestrator and Aria Automation to perform extensibility tasks.

## You've finished the module

[356]

Congratulations on completing the lab module.

If you are looking for additional information on extensibility, have a look here;

<https://www.vmware.com/uk/products/aria-automation.html>

<https://www.vmware.com/uk/products/aria-automation-orchestrator.html>

From here you can:

1. Continue with the next lab module
2. Click [vlp:table-of-contents]Show Table of Contents] to jump to any module or lesson in this lab
3. End your lab and return in the future

## Appendix

### Hands-on Labs Interface (Windows Main Console)

[358]

Welcome to Hands-on Labs! This overview of the interface and features will help you to get started quickly. Click next in the manual to explore the Main Console or use the Table of Contents to return to the Lab Overview page or another module.

### Location of the Main Console

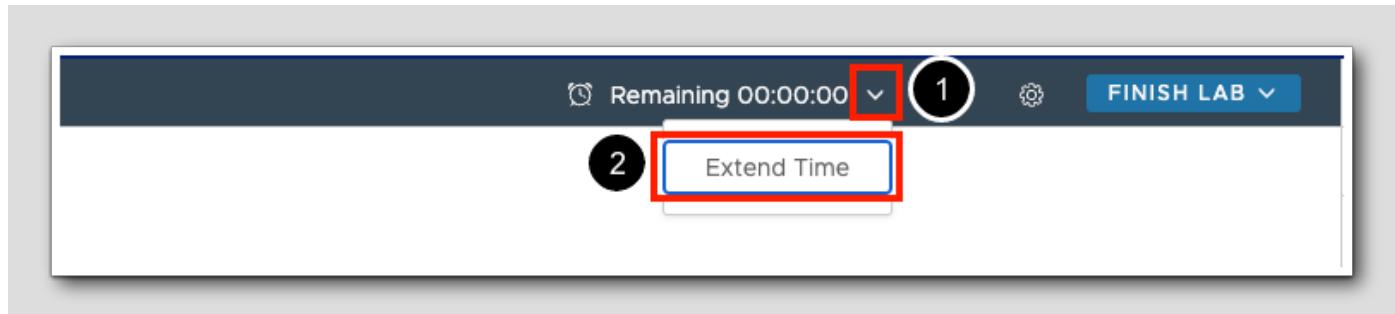
[359]



1. The area in the large RED box contains the Main Console. The Lab Manual is on the tab to the right of the Main Console.

## Extend Time

[360]



1. Your lab starts with a timer. The lab cannot be saved and will end when the timer expires. Click the drop down arrow next to the remaining time
2. Select Extend Timeto increase the time allowed. The amount of time you can extend will depend on the lab.

## Alternate Methods of Keyboard Data Entry

[361]

In this lab you will input text into the Main Console. Besides directly typing in the console, two alternate methods make it easier to enter complex data.

## Click and Drag Lab Manual Content Into Console Active Window

[362]

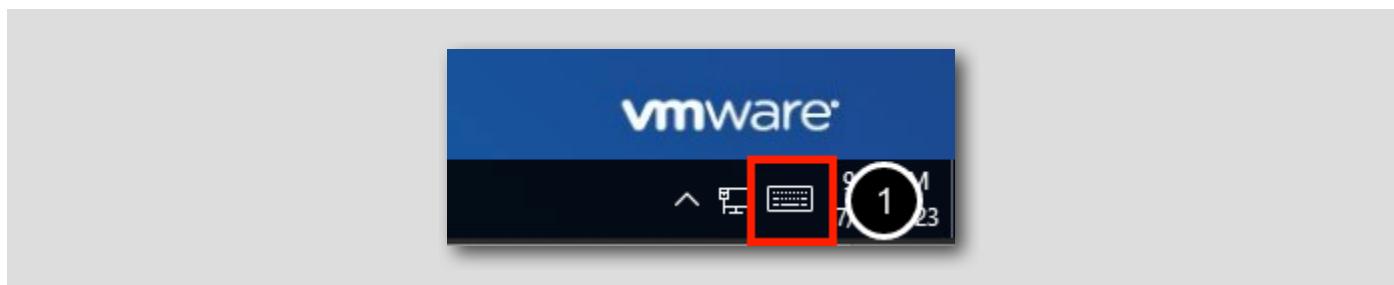
<https://www.youtube.com/watch?v=xS07n6GzGuo>



You can 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

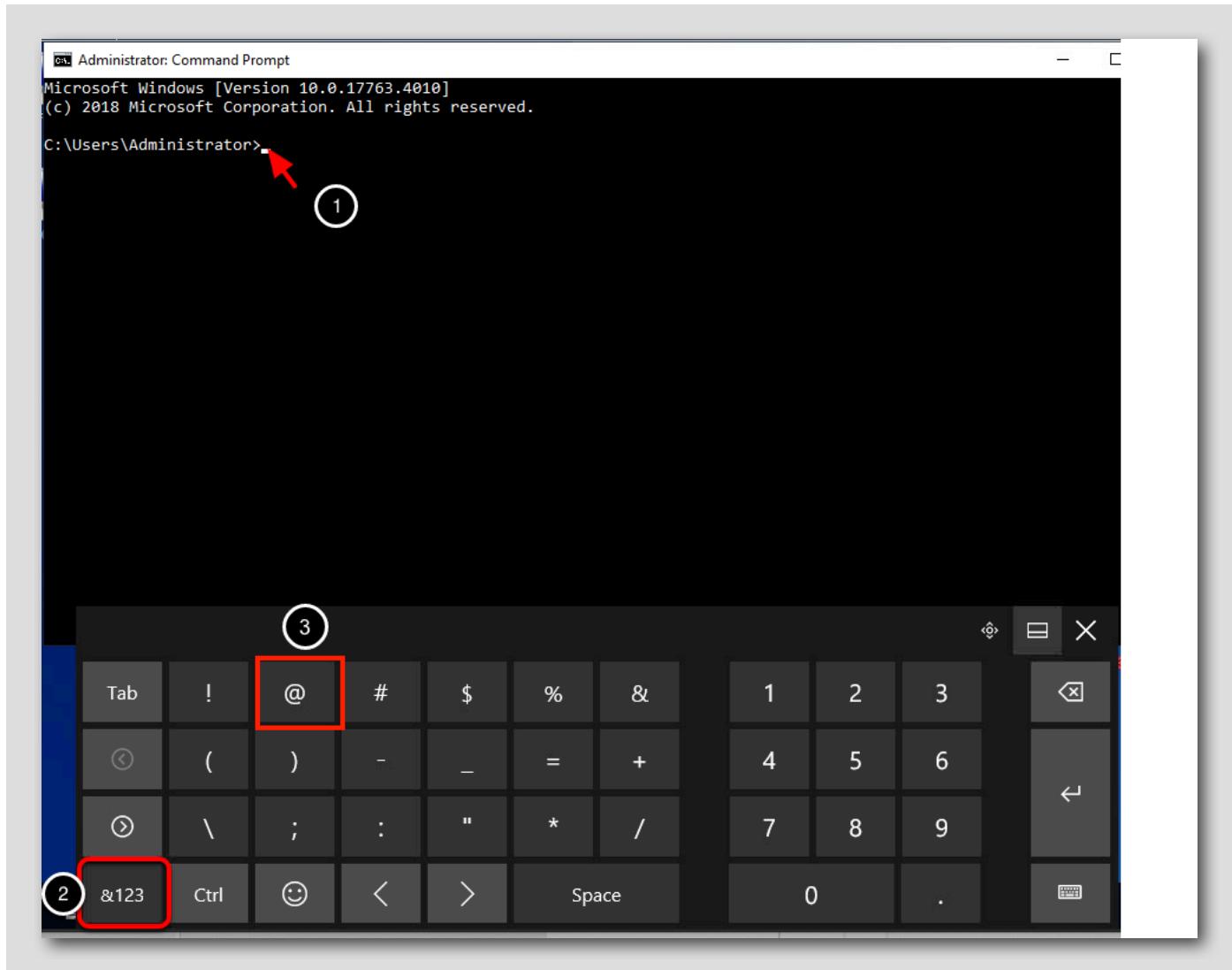
[363]



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



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

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

## Return to Lab Guidance

[365]

Use the Table of Contents to return to the Lab Overview page or another module.

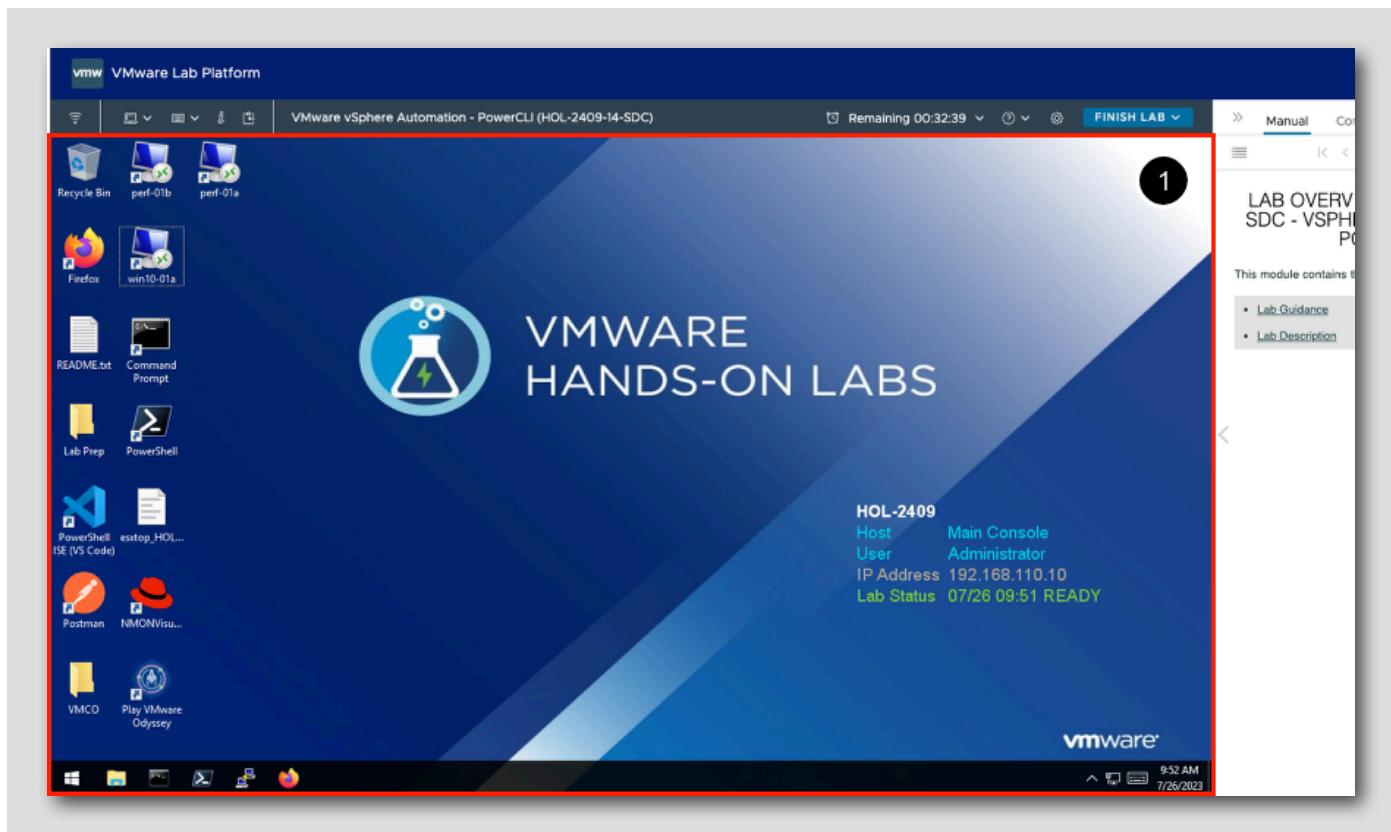
## Hands-on Labs Interface (Ubuntu Main Console)

[366]

Welcome to Hands-on Labs! This overview of the interface and features will help you to get started quickly. Click next in the manual to explore the Main Console or use the Table of Contents to return to the Lab Overview page or another module.

### Location of the Main Console

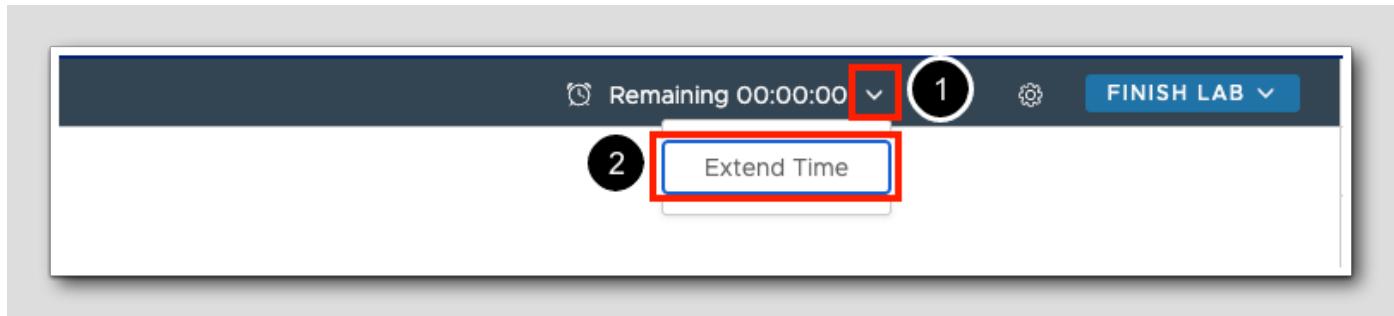
[367]



1. The area in the large RED box contains the Main Console. The Lab Manual is on the tab to the right of the Main Console.

## Extend Time

[368]



1. Your lab starts with a timer. The lab cannot be saved and will end when the timer expires. Click the drop down arrow next to the remaining time
2. Select Extend Timeto increase the time allowed. The amount of time you can extend will depend on the lab.

## Alternate Methods of Keyboard Data Entry

[369]

In this lab you will input text into the Main Console. Besides directly typing in the console, two alternate methods make it easier to enter complex data.

## Click and Drag Lab Manual Content Into Console Active Window

[370]

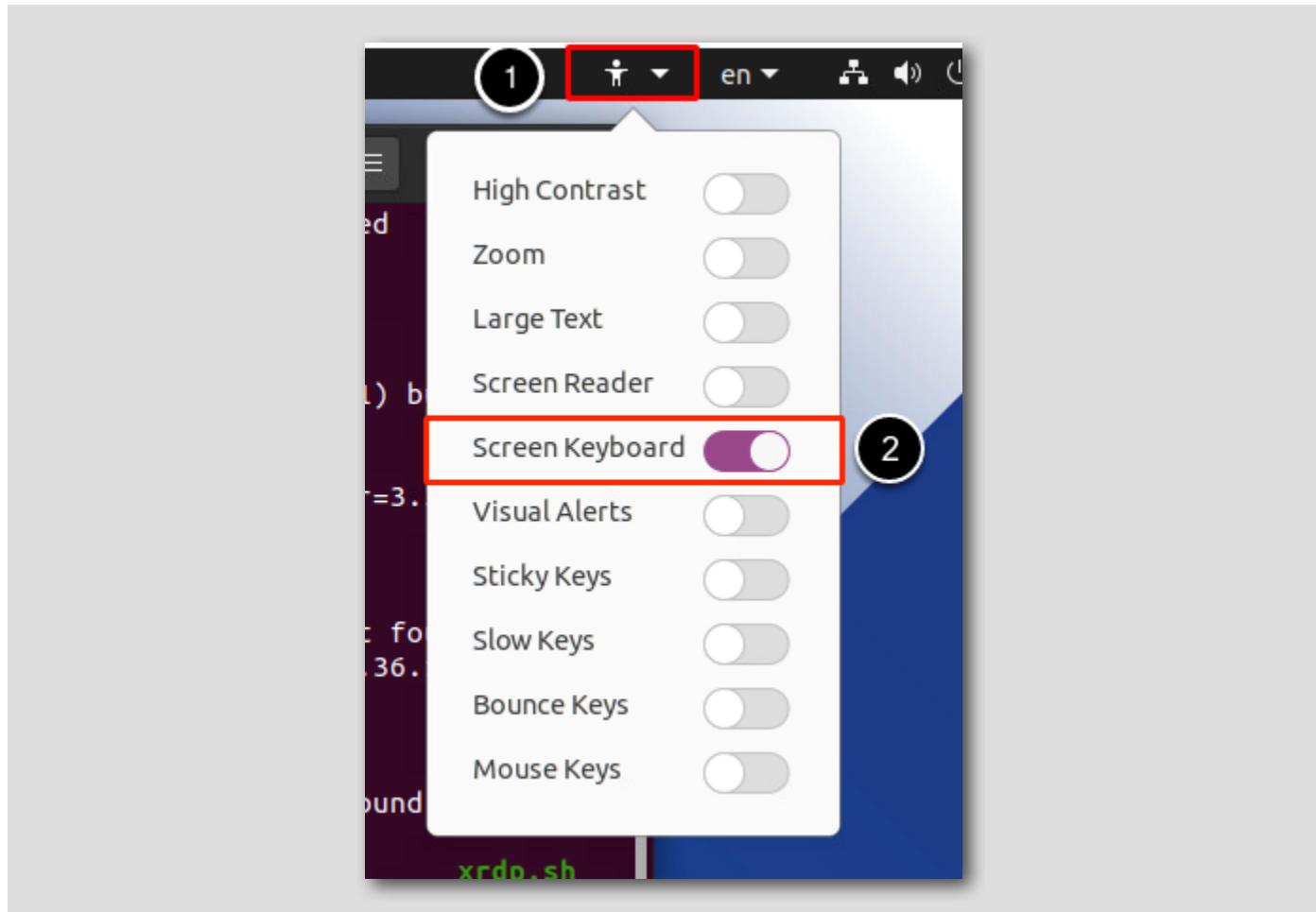
<https://www.youtube.com/watch?v=xS07n6GzGuo>



You can 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

[371]

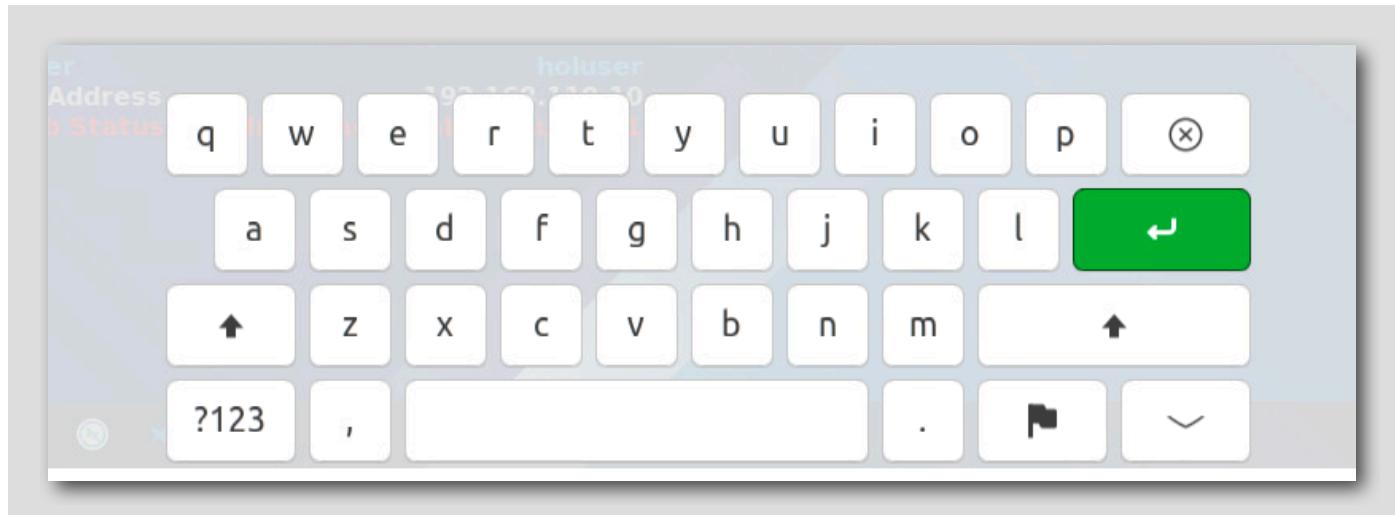


You can also use the Online International Keyboard found in the Main Console.

1. Click on the Human icon (Universal Access) on the top taskbar
2. Enable Screen Keyboard

## The Keyboard Is Now Enabled

[372]



The keyboard will now be enabled and will autohide and appear when needed; e.g., when you click in a text field or terminal.

## Return to Lab Guidance

[373]

Use the Table of Contents to return to the Lab Overview page or another module.



VMware, Inc. 3401 Hillview Avenue Palo Alto CA 94304 USA Tel 877-486-9273 Fax 650-427-5001 [vmware.com](http://vmware.com).  
Copyright © 2024 VMware, Inc. All rights reserved. This product is protected by U.S. and international copyright and intellectual property laws. VMware products are covered by one or more patents listed at [vmware.com/go/patents](http://vmware.com/go/patents). VMware is a registered trademark or trademark of VMware, Inc. and its subsidiaries in the United States and other jurisdictions. All other marks and names mentioned herein may be trademarks of their respective companies. Lab SKU: HOL-2501-11-VCF-L Version: 20241111-143154