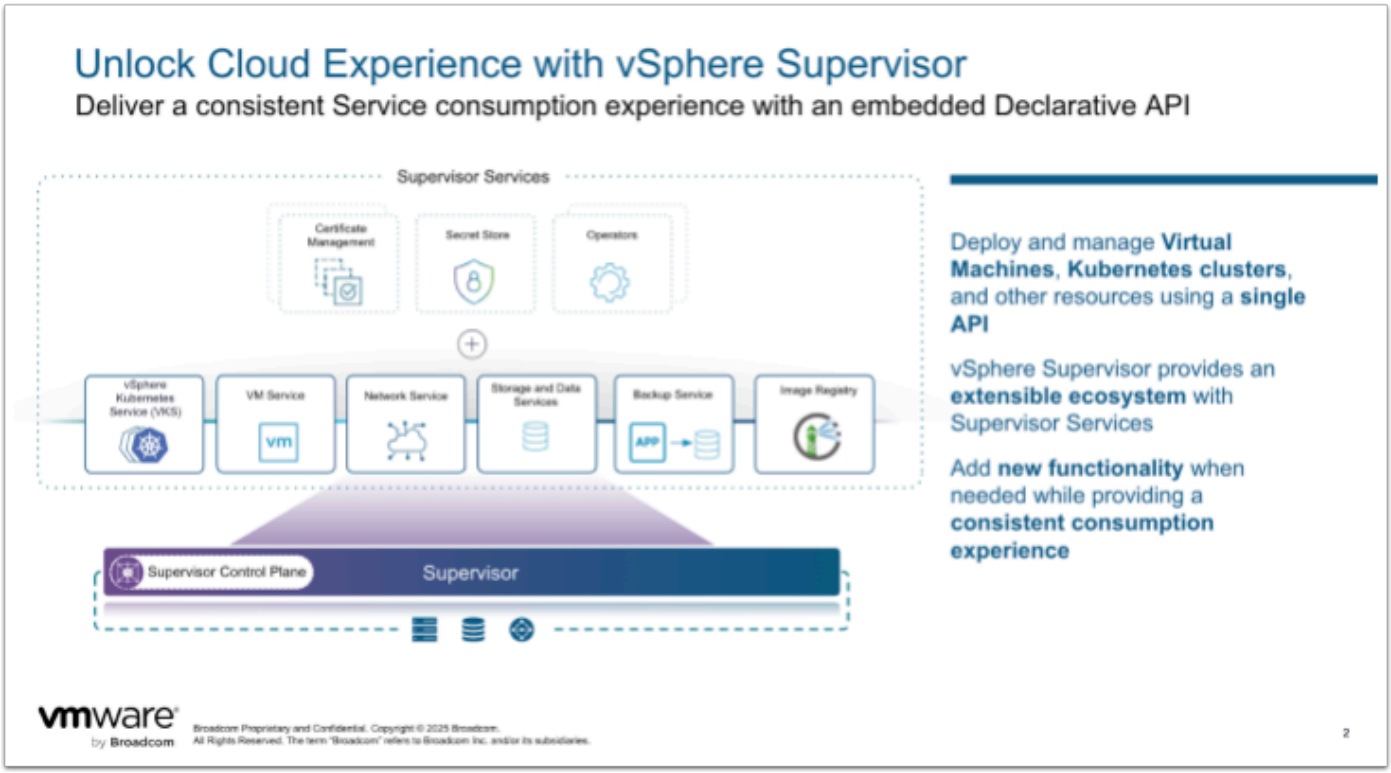
HOL-2636-01-VCF-L - ATE

Table of Contents

Lab Overview - HOL-2636-01-VLP-L: Managing Kubernetes and Cloud Services on VMware Cloud Foundation

Introduction



vSphere Supervisor delivers a consistent Service consumption experience with an embedded Declarative API. By using the integration between the Supervisor and VMware Cloud Foundation you can deploy and operate the compute, networking, and storage infrastructure for vSphere Supervisor (previously known as Workload Management). vSphere Supervisor transforms vSphere into a platform for running Kubernetes workloads natively on the hypervisor layer. When enabled on a vSphere cluster, the Supervisor provides the capability to deploy VMs using the VM Service, to run Kubernetes workloads directly on ESX hosts, and to create upstream Kubernetes clusters within dedicated resource pools.

Lab Guidance

Welcome! This lab is available for you to repeat as many times as you want.Use the Table of Contents in the upper right-hand corner of the Lab Manual to jump ahead to any module.

Module	Title	Length	Level
1	vSphere Supervisor concepts and components.	15 min	Beginner
2	Enable and configure the vSphere Supervisor	30 min	Beginner
3	Login to The Cluster and Deploy An Application	30 min	Basic
4	Updating The Supervisor And VKS Services	15 min	Basic
5	Unlocking Gitops with ArgoCD	20 min	Advanced

This lab has 2 Modules. The first module explains vSphere Supervisor concepts and how it interacts with vCenter and vSphere. In Module 2 you will interact with the Supervisor (previously known as Workload Management) by configuring the service and deploying both Kubernetes clusters and vSphere PODs with virtual machines.

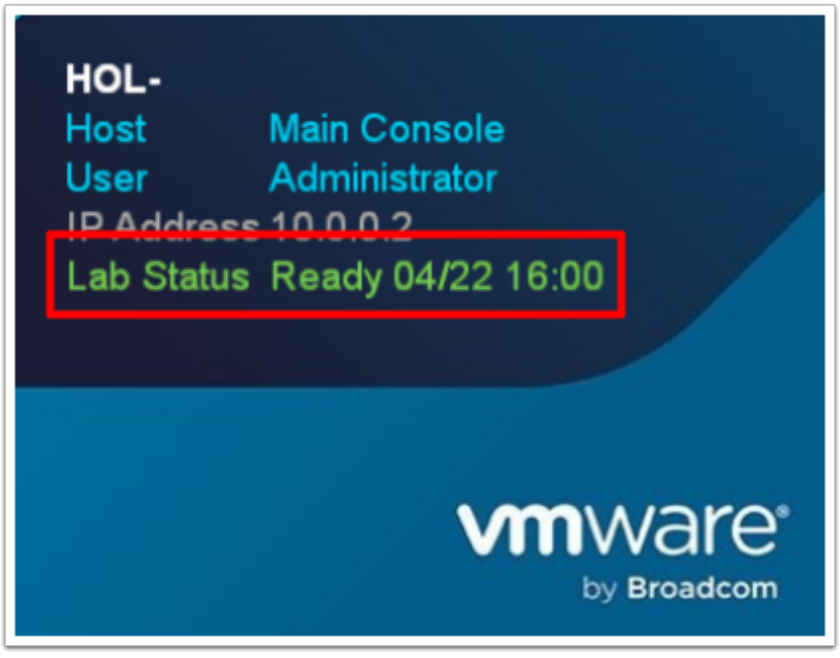
Lab Authors:

Principal: Kevin Brady

Captains: Katarina Brookfield, Randy Carson, Michael Fleisher, Pablo Iglesias

First Time Using Hands-on Labs?

If this is your first time taking a lab you can review the VMware Learning Platform interface before proceeding.



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.

Module 1 - vSphere Supervisor concepts and components (15 minutes) Basic

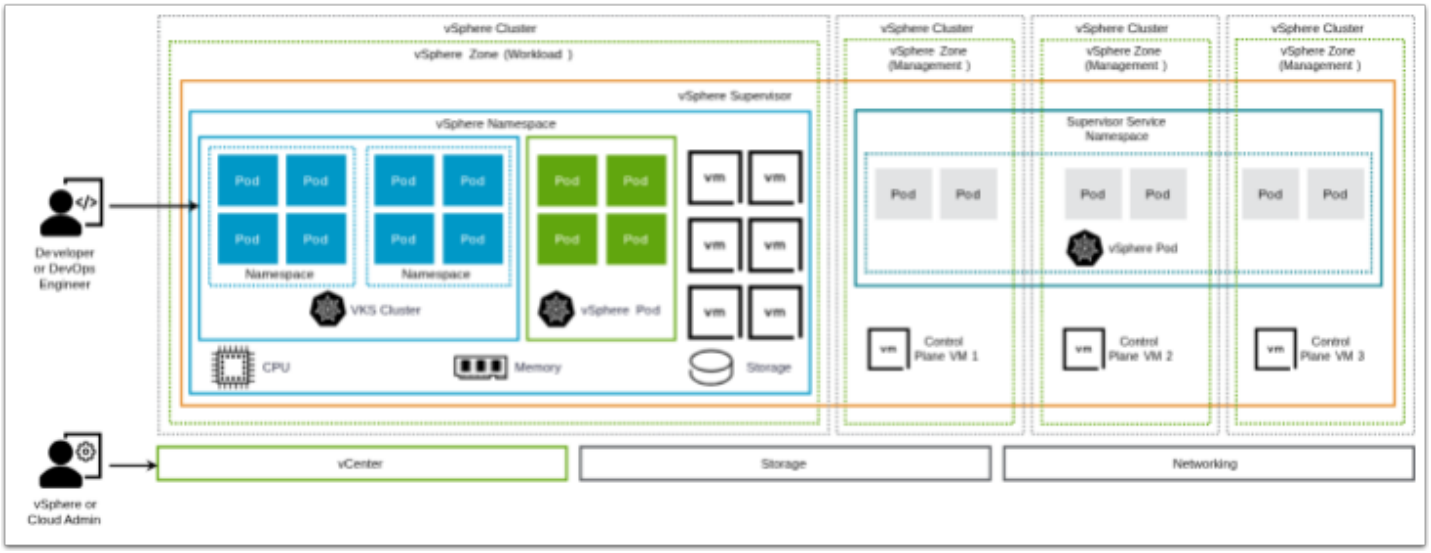
Intro

You can use vSphere Supervisor to enable a consistent declarative API in vSphere and create a platform for building a modern cloud experience with VCF Automation. When activated on vSphere clusters, the Supervisor provides the capability to run workloads in vSphere in declarative fashion, which includes creating and managing upstream Kubernetes clusters, virtual machines, and vSphere Pods . The resource boundaries where workloads run in the Supervisor are called vSphere Namespaces .

vSphere Supervisor supports Kubernetes 1.32, is bootstrapped with cluster API, and is decoupled from the vCenter lifecycle. That means you can upgrade Kubernetes based on your applications and not on vCenter upgrades.

vSphere Supervisor

vSphere Supervisor provides a declarative API and desired state by deploying a Kubernetes control plane directly on vSphere clusters. As a vSphere administrator, you activate existing vSphere clusters for the Supervisor , thus creating a Kubernetes layer within the clusters.



Let's define what some of the vSphere Supervisor constructs are. We start with a standard vSphere Cluster as a resource boundary for each Kubernetes environment needed for our business needs.

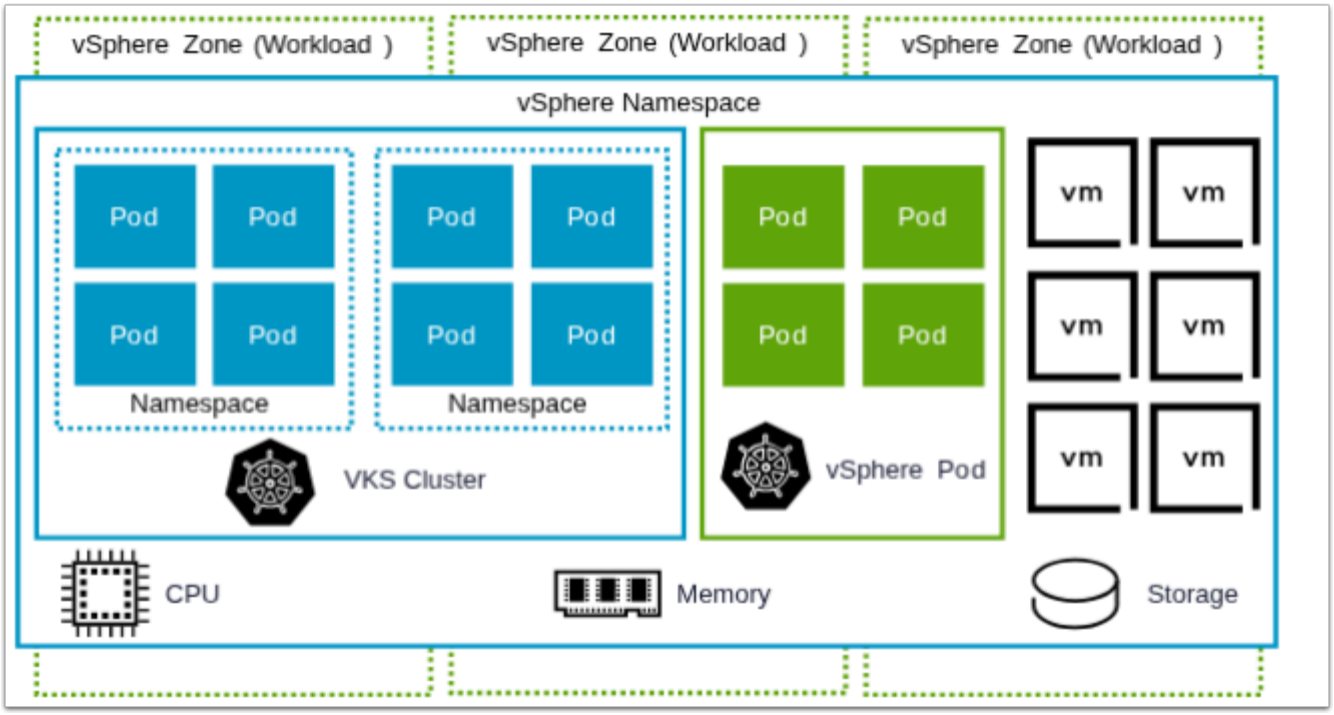
vSphere Zone (Workloads)

In the vSphere Supervisor, workloads are applications deployed in one of the following ways:

- Applications that consist of containers running inside vSphere Pods .
- Workloads provisioned through the VM service.
- Kubernetes clusters deployed by using vSphere Kubernetes Service (VKS)
- Applications that run inside VKS clusters.

vSphere Zones protect against cluster-level failure by providing high-availability to workloads deployed on the Supervisor . As a vSphere administrator, you create vSphere Zones in the vSphere Client and then map vSphere clusters to zones. There are 2 levels of high availability that you can provide with vSphere Zone. Creating a single vSphere Zone only gives you host level redundancy. To provide cluster level redundancy, you would create 3 vSphere Zones. This would spread the control plane VMs across these 3 zones giving you cluster level redundancy.

vSphere Namespace



Think of a vSphere Namespace as a Resource Pool that can contain vSphere Pods, VMs, and VKS clusters. A vSphere administrator can set limits for CPU, memory, storage, as well as the number of Kubernetes objects that can run within the vSphere Namespace.

When initially created, a vSphere Namespace has unlimited resources within the Supervisor. As a vSphere administrator, you can set limits for CPU, memory, storage, as well as the number of Kubernetes objects that can run within the vSphere Namespace. Storage limitations are represented as storage quotas in Kubernetes. A resource pool is created in vSphere per each vSphere Namespace on the Supervisor

You can map up to three vSphere Zones to a vSphere Namespace for running workloads on them. The vSphere Namespace spreads across all vSphere clusters that are part of the vSphere Zones. The resources utilized on a vSphere Namespace are taken from all the underlying vSphere clusters on equal parts. For example, if you dedicate 300 MHz of CPU, 100 MHz are taken from each vSphere cluster.

Kubernetes Namespace

In Kubernetes, namespaces provide a mechanism for isolating groups of resources within a single cluster. Names of resources need to be unique within a namespace, but not across namespaces.

Namespaces are a way to organize clusters into virtual sub-clusters — they can be helpful when different teams or projects share a Kubernetes cluster. Any number of namespaces are supported within a cluster, each logically separated from others but with the ability to communicate with each other. Namespaces cannot be nested within each other.

Any resource that exists within Kubernetes exists either in the default namespace or a namespace that is created by the cluster operator. Only nodes and persistent storage volumes exist outside of the namespace; these low-level resources are always visible to every namespace in the cluster.

Differences Between a vSphere Namespace and Kubernetes Namespace

Although in its core a vSphere Namespace serves the same function as a Kubernetes namespace, a vSphere Namespace is specific to the vSphere Supervisor. You should not confuse a vSphere Namespace with a Kubernetes namespace.

A vSphere Namespace is implemented as an extension to a vSphere resource pool and its function is to provide resources to workloads running in the Supervisor. A vSphere Namespace has a direct mapping to a Kubernetes namespace through which object and storage quotas are enforced upon workloads.

Another difference with a regular Kubernetes namespace is that the vSphere administrator manages the user access to vSphere Namespaces, as mentioned from above. The vSphere administrator can also associate VM classes and Content Libraries containing VM templates that DevOps engineers can use to self-service VMs.

Supervisor Networking

A Supervisor can either use a vSphere networking stack or NSX to provide connectivity to Supervisor control plane VMs, services, and workloads. A Supervisor that is configured with the vSphere networking stack, all hosts from the Supervisor are connected to a VDS that provides connectivity to workloads and Supervisor control plane VMs. A Supervisor that uses the vSphere networking stack requires a load balancer on the vCenter management network to provide connectivity to DevOps users and external services. A Supervisor that is configured with NSX, uses virtualized networks of the solution and NSX Load Balancer or the Avi Load Balancer to provide connectivity to external services and DevOps users.

Supervisor Storage

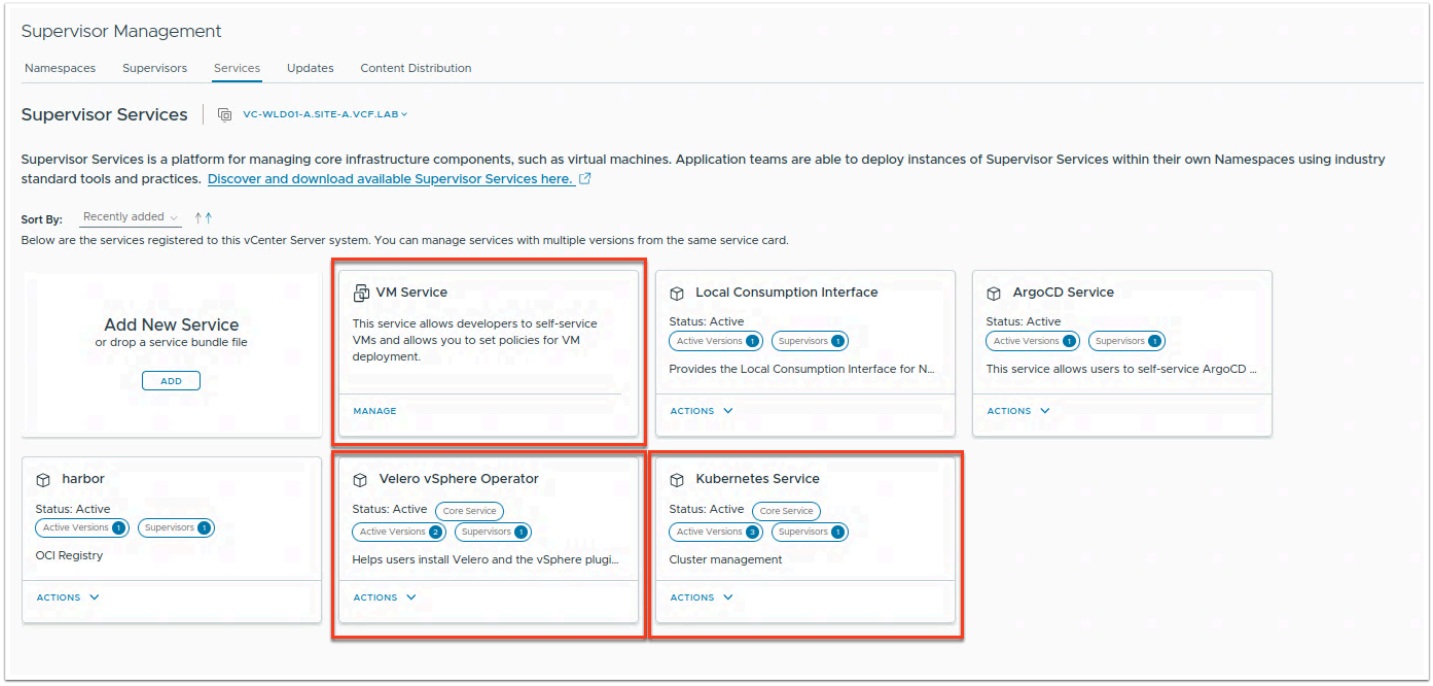
A vSphere Zone is a fault domain that can fail independently from other zones. Supervisor supports different storage topologies for zones. These topologies define how datastores are attached to ESX hosts and how they behave during zone failures. Zonal datastore, a zonal datastore is local to a single vSphere Zone. It is attached to all the ESX hosts that are part of the underlying cluster. Cross-zonal datastore, a cross-zonal datastore spans multiple vSphere Zones. It is attached to ESX hosts from all the zones it covers.

Supervisor uses storage policies to integrate with storage available in your vSphere environment. The policies represent datastores and manage storage placement of such components and objects as control plane VMs, vSphere Pod ephemeral disks, and container images. You might also need policies for storage placement of persistent volumes and VM content libraries. If you use VKS clusters, the storage policies also dictate how the VKS cluster nodes are deployed. Storage policies support any shared datastores in your environment, such as VMFS, NFS, vSAN, and vSAN ESA. Depending on your vSphere storage environment and the needs of DevOps, you can create several storage policies for different classes of storage. When you enable a Supervisor and set up vSphere Namespaces, you can assign different storage policies to be used by various objects, components, and workloads.

Certain Kubernetes workloads that DevOps run on a namespace require persistent storage to store data permanently.

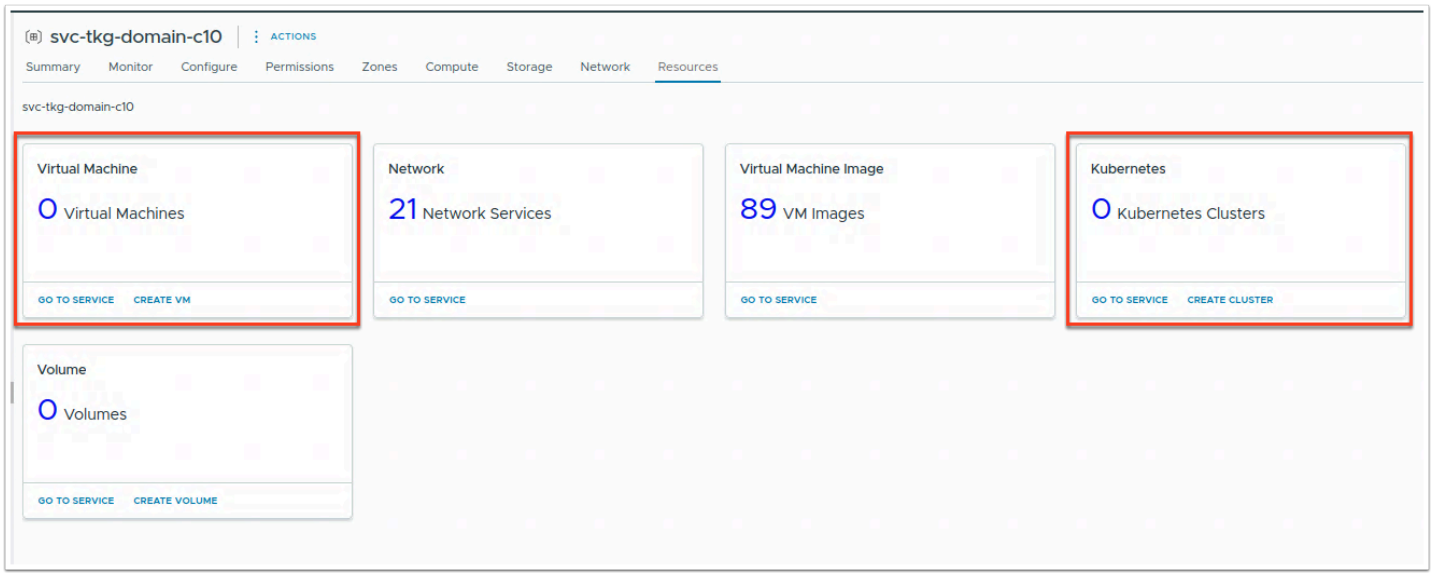
Persistent storage can be used by vSphere Pods , VKS clusters, VMs, and other workloads you run on the namespace. To make persistent storage available to the DevOps team, the vSphere administrator creates storage policies that describe different storage requirements and classes of services. The administrator then assigns storage policies and configures storage limits at a namespace level. To understand how the Supervisor works with persistent storage, be familiar with the such essential Kubernetes concepts as storage classes, persistent volumes, and persistent volume claims. For more information, see the Kubernetes documentation at https://kubernetes.io/docs/home/.

Extensible through Supervisor Services



vSphere Supervisor has a catalog of Services that can be added once it is enabled. There are 3 services that are installed by default when you enable the Supervisor. Those services areVM Service, Kubernetes Service,andVelero vSphere Operator Service. These base services allow you to create vSphere PODs (for virtual machine deployments), Kubernetes clusters (VKS), and be able to backup your kubernetes environment with Velero.

Some of the other services that can be added to the Supervisor are Certificate Management, DNS, and vSAN Data Persistence Platform (vDPP) to mention a few. As you can see in the screen shot we have also added 2 services to the default install, Contour and Harbor. We added these 2 services so we could have an off-line image repository for use later in this lab. In Module 2 we will walk you through adding a Grafana Operator service to the vSphere Supervisor.



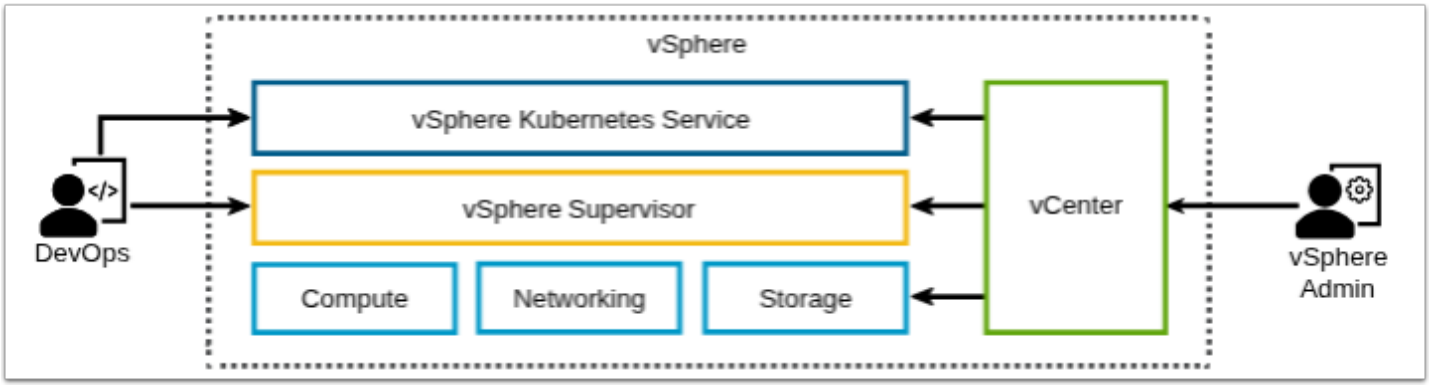
A very useful service to add to the Supervisor is the **Local Consumption Interface**. This service gives you a great picture into the resources being used in a namespace. You can also create **VMs**, **Kubernetes Clusters** (**VKS**), and storage volumes as well as add to some existing service for that particular namespace.

As you can see in the screen shot we have added 2 services to the default install, **Contour** and **Harbor**. We added these 2 services so we could have an off-line image repository. We will walk you through adding the **Grafana Operator** services to this **Supervisor**.

The control plane and declarative API

The control plane and declarative API

When you activate a Supervisor on vSphere Zones, a Kubernetes control plane is created. This control plane enables the capability to run workloads in a declarative fashion on vSphere clusters.



A Supervisor runs on top of an SDDC layer that consists of ESX for compute, NSX or vSphere Distributed Switch networking, and vSAN or another shared storage solution. Shared storage is used for persistent volumes for vSphere Pods and VMs running inside the Supervisor, and pods in a VKS cluster. After a Supervisor is created, as a vSphere administrator you can create vSphere Namespaces within the Supervisor. As a DevOps engineer, you can run workloads consisting of containers running inside vSphere Pods, deploy VMs through the VM service, and create VKS clusters.

Module 2 -Enable and Configure vSphere Supervisor(30minutes) Basic

Enabling vSphere Supervisor

We will not enable vSphere Supervisor in this section. We will walk through the steps but not actually create one. The creation process for the Supervisor takes longer than this lab allows

In this module, we will walk through the process of enabling the Supervisor on the Management vCenter. You can only enable 1 supervisor per cluster and in this lab the supervisor on the workload cluster `wld-01a` has been enabled for you to use in this lab.

A vSphere Zone Storage Policy are prerequisites for Enabling the Supervisor

vSphere Cluster

To activate a Supervisor, you must create at least one vSphere Zone that will act as the Management Zone of the Supervisor. To provide zone-level HA to the Supervisor control plane, you deploy the Supervisor on three Management Zones. In Supervisor with three Management Zones and HA for the control plane, meaning with three control plane VMs, each control plane VM is placed on a different Management Zone.

- Create one or three vSphere clusters with at least 3 hosts in each zone. For storage with vSAN, the cluster must have 4 hosts. For testing of POC environments, each cluster must have at least 1 host, and 2 hosts when using vSAN.
- Configure storage with vSAN or other shared storage solution for each cluster.
- Enable vSphere HA and vSphere DRS on Fully Automate or Partially Automate mode.
- All clusters must be connected to the same VDS and Management traffic network.

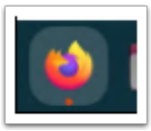
Storage Policy

Before you activate the Supervisor, create storage policies to be used in the Supervisor and vSphere Namespaces. The policies represent datastores and manage storage placement of such components and objects as Supervisor control plane VMs, vSphere Pod ephemeral disks, and container images. You might also need policies for storage placement of persistent volumes and VM content libraries. If you use VKS clusters, the storage policies also dictate how the VKS cluster nodes are deployed.

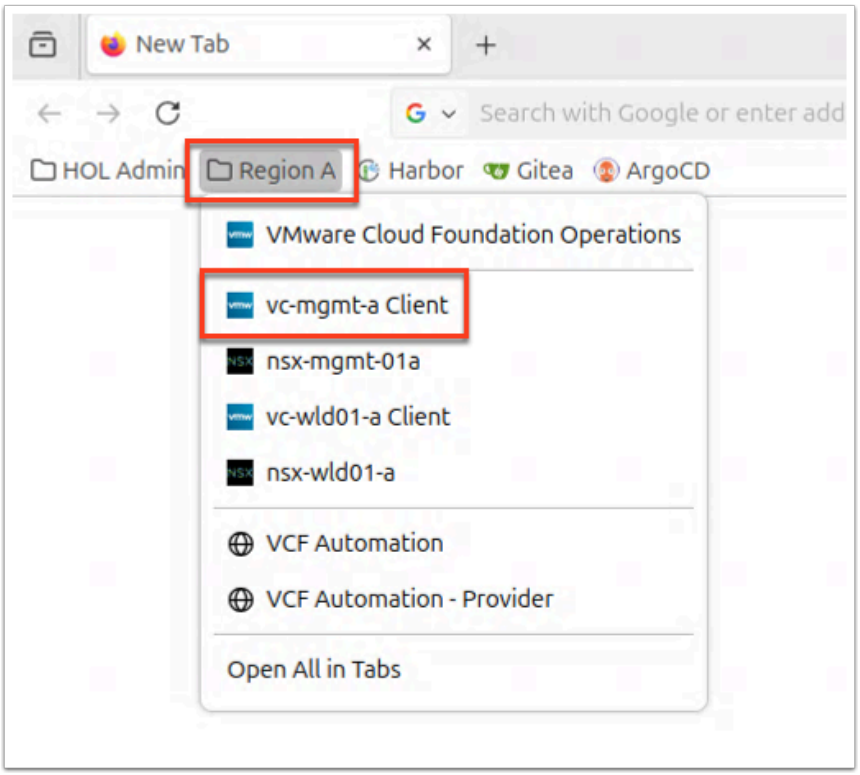
Depending on your vSphere storage environment and the needs of DevOps, you can create several storage policies for different classes of storage. For example, if your vSphere storage environment has three classes of datastores, Bronze, Silver, and Gold, you can create storage

policies for all datastore types. When you enable a Supervisor and set up namespaces, you can assign different storage policies to be used by various objects, components, and workloads.

Connect to the Management vCenter



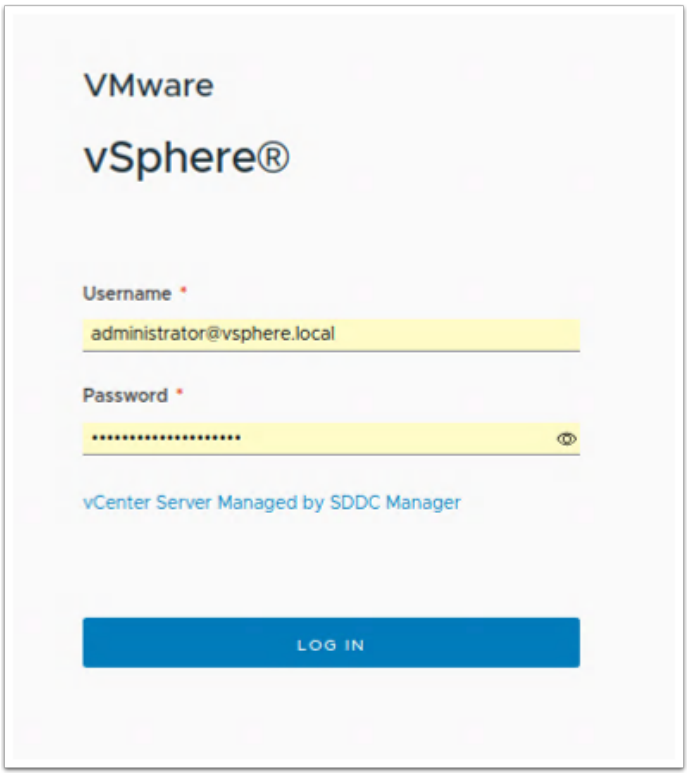
1. Open Firefox from the taskbar at the bottom of the screen



2. Select **Region A** tab

3. Select **vc-mgmt-a** Client from the **Region A** tab

Login to vSphere



Log in with:

Username:

```
Administrator@vsphere.local
```

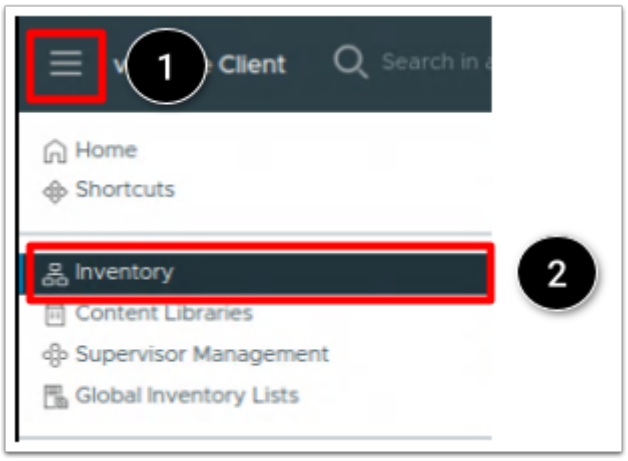
Click to copy

Password:

```
VMware123!VMware123!
```

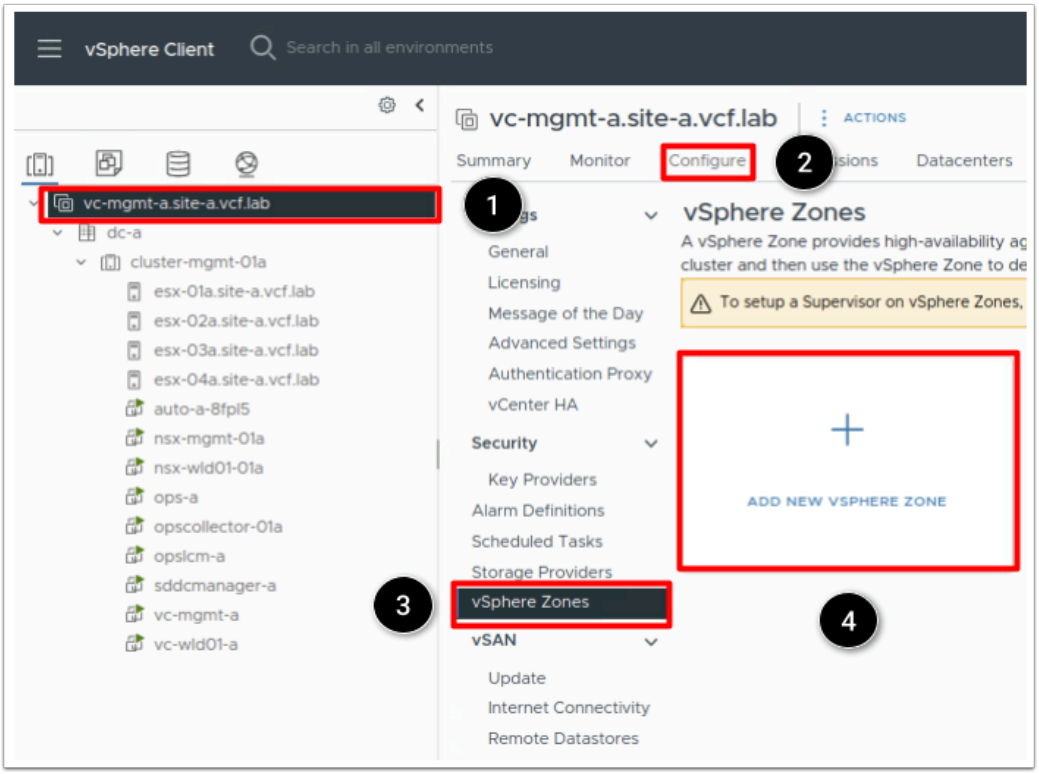
Click to copy

Create a vSphere Zone



1. Select the **Menu** in the upper left corner
2. Select **Inventory**

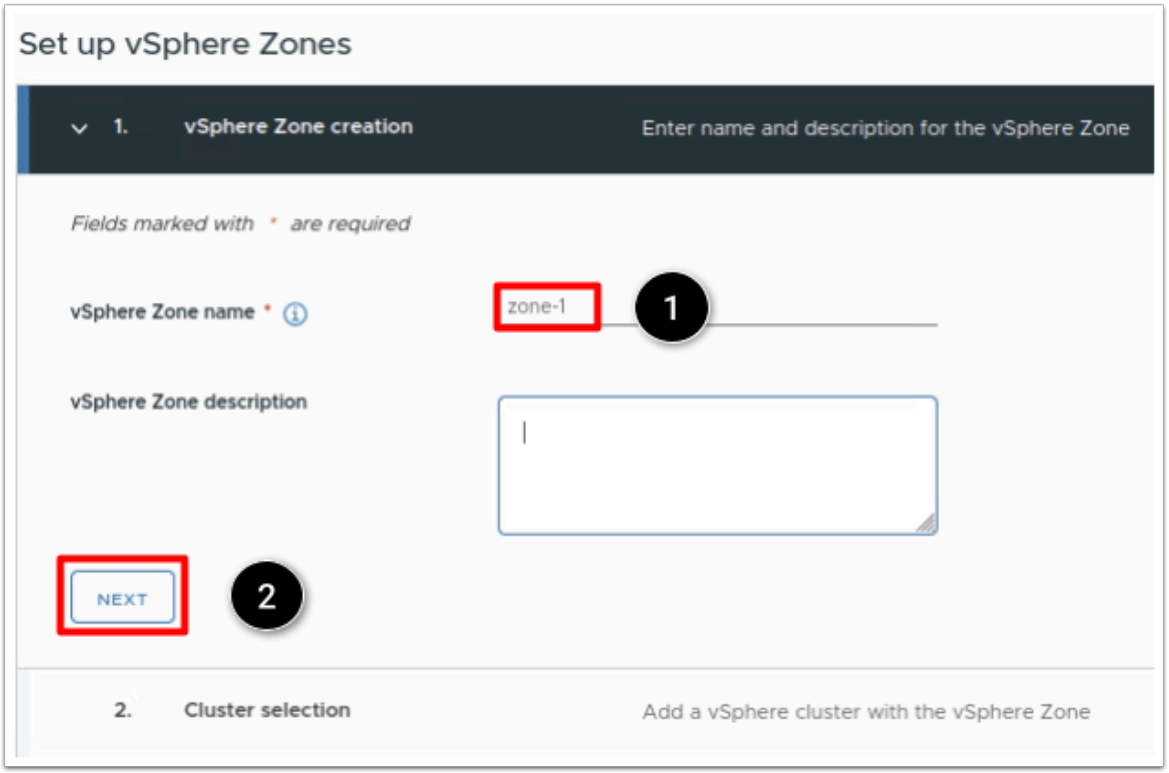
Create a New vSphere Zone



We need to set up at least 1 vSphere Zone to be able to activate the Supervisor.

1. Select the vCenter **vc-mgmt-a.site-a.vcf.lab**.
2. Select **Configure**
3. Select **vSphere Zones**
4. Select **ADD NEW VSPHERE ZONE**

Set Up the vSphere Zone

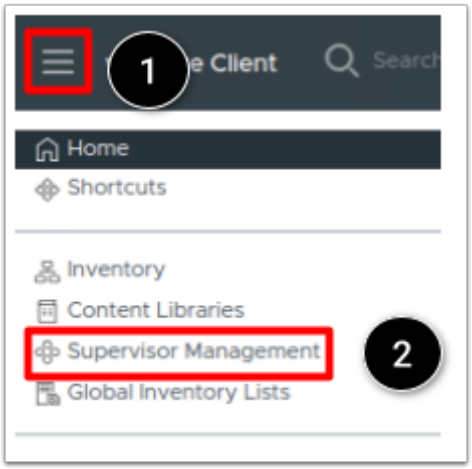


1. Type the name zone-1. (There are some rules to follow when naming your zone. To see those rules click the blue “i” next tovSphere Zone name.)
2. Select **Next**

Select the Clusters

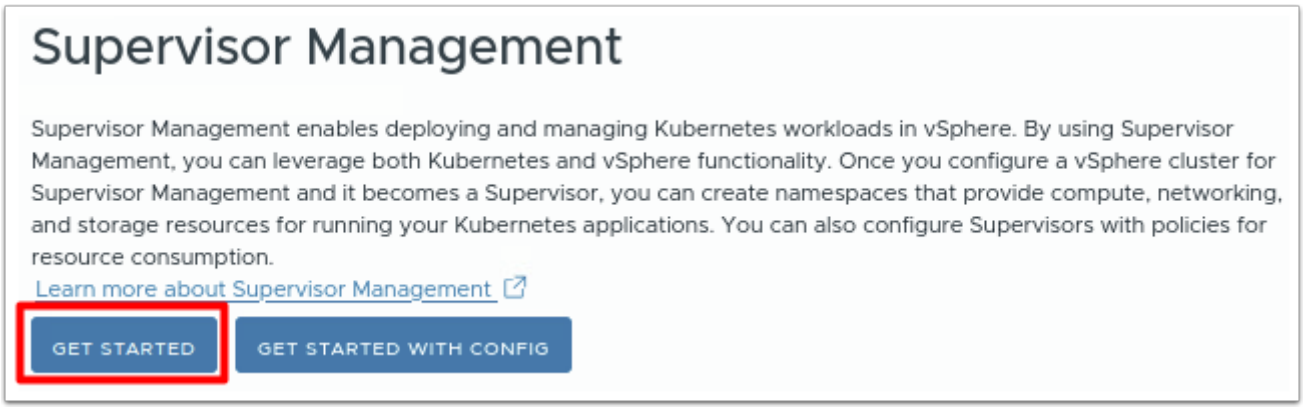
1. Select **cluster-mgmt-01a**. You will want to pick the cluster that you want to use for the *VKS Cluster vSphere Pods.*
2. Select **Finish**

Start the Supervisor create wizard



1. Select the **Menu** in the upper left corner
2. Select **Supervisor Management**

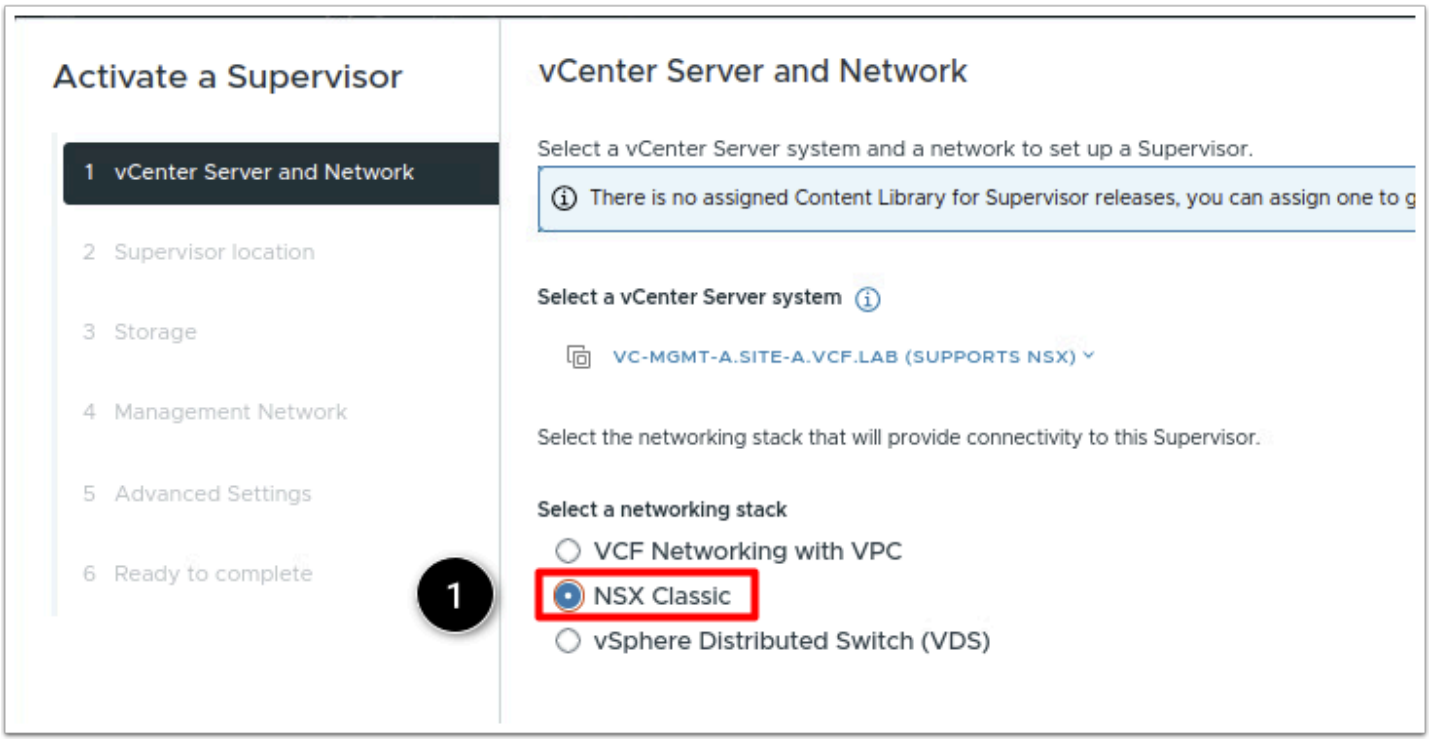
Get Started in the Wizard



We will be walking through the setup wizard. If you already had a JSON config file from a previous setup you could use that file to *declare* the state of the Supervisor deployment.

1. Click **GET STARTED**.

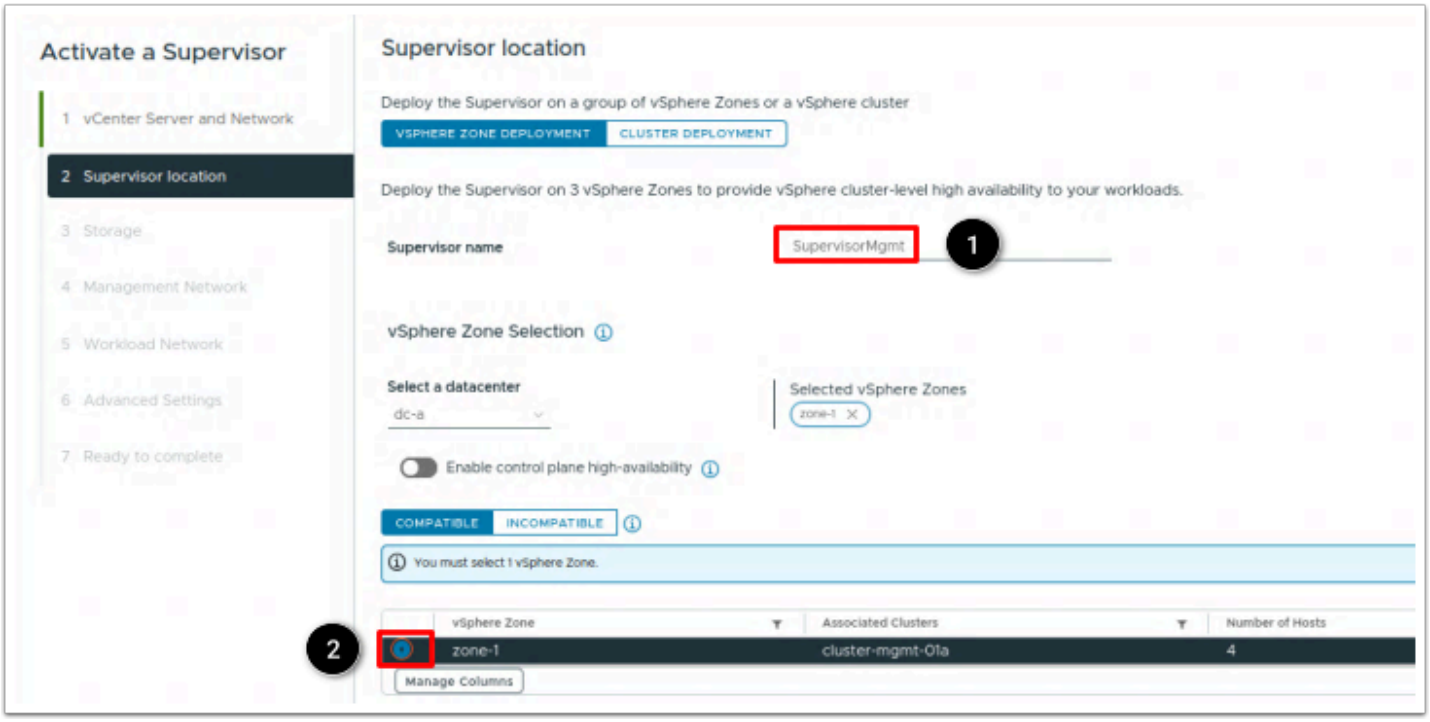
Select the Networking Stack



Select the best underlying networking for your supervisor to use when creating*VKS cluster* and *vSphere PODs*. For this module we will be using NSX networking

1. Select **NSX Classic**
2. Select **Next**(not shown)

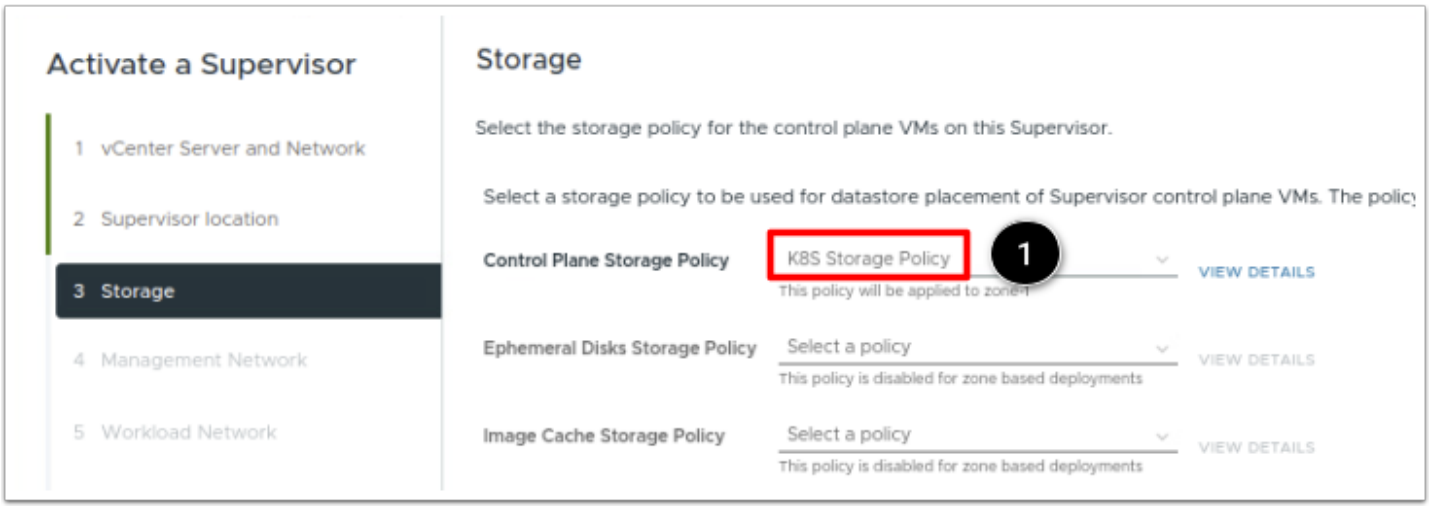
Configure Supervisor Location



In this step, we are going to use the zone that we created in the previous step.

1. Enter the **Supervisor Name:** ***SupervisorMgmt***
2. Select the available **vSphere Zone:** *zone-1*. This was previously created as one of the prerequisites.
3. Click **Next**(not shown).

Configure Storage



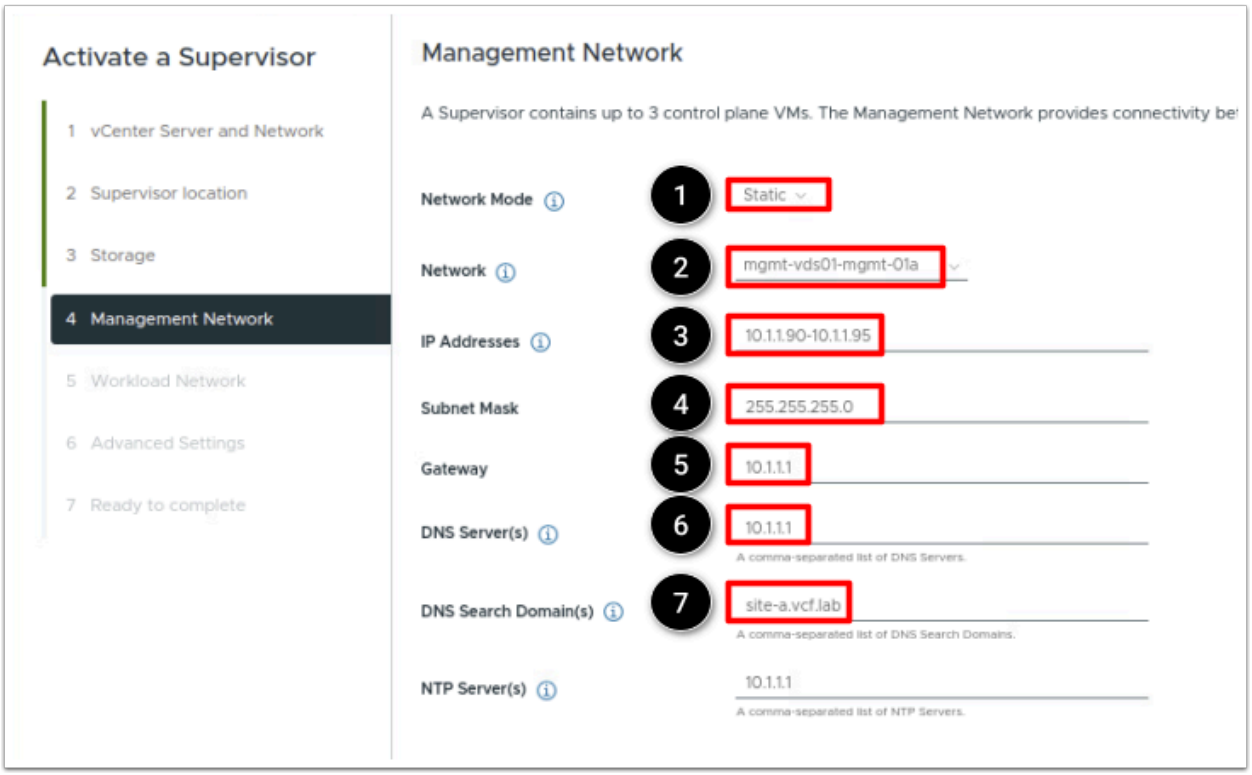
We will only need to select the Control Plane Storage Policy.

1. Select **K8S Storage Policy** from the drop down menu. This was previously created as one of the prerequisites.
2. Select **Next**(not shown)

Configure Management Network

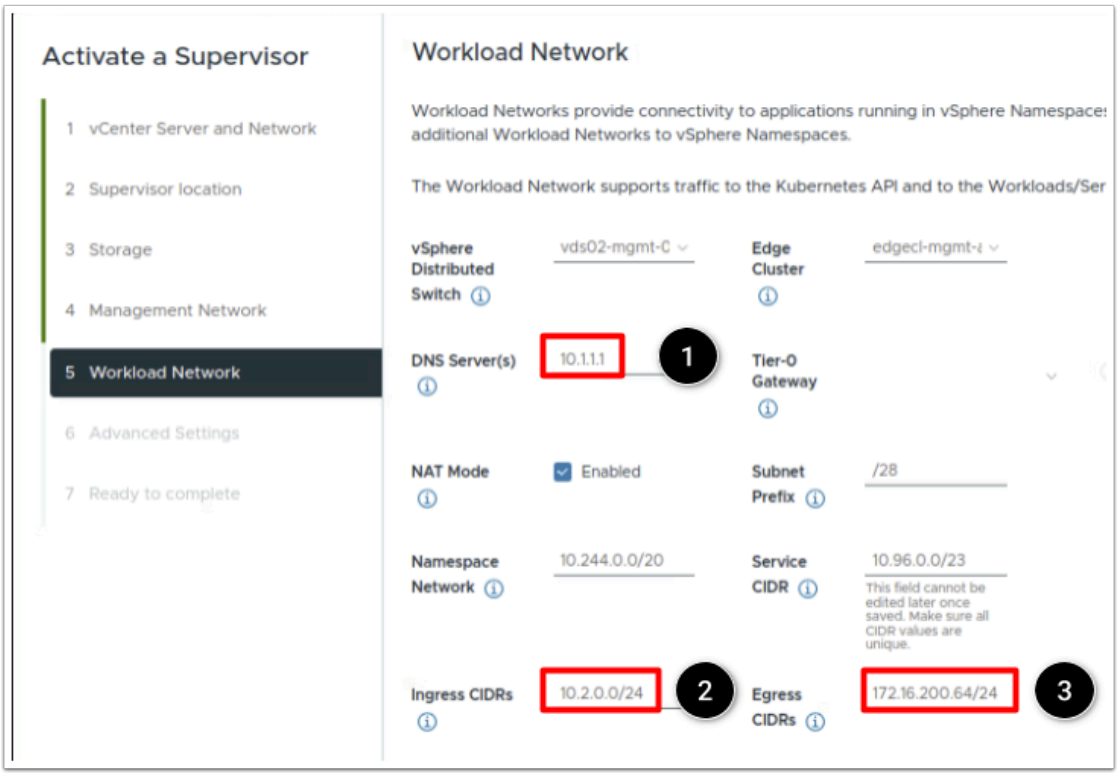
We need to enter the network values for the Management Network that contains the 3 control plan VMs.

These static IP addresses are specific to this HOL lab.



1. Change the Network Mode toStatic
2. Select the Network ***mgmt-vds01-mgmt-01a***
3. Input the IP address for the control plane VMs, type in ***10.1.1.90-10.1.1.95***
4. Enter the **Subnet Mask:** *255.255.255.0*
5. Enter the **Gateway:** 10.1.1.1
6. Enter the **DNS Server**: 10.1.1.1
7. Enter the **DNS Search Domain:** site-a.vcf.lab
8. Select Next

Configure Workload Network

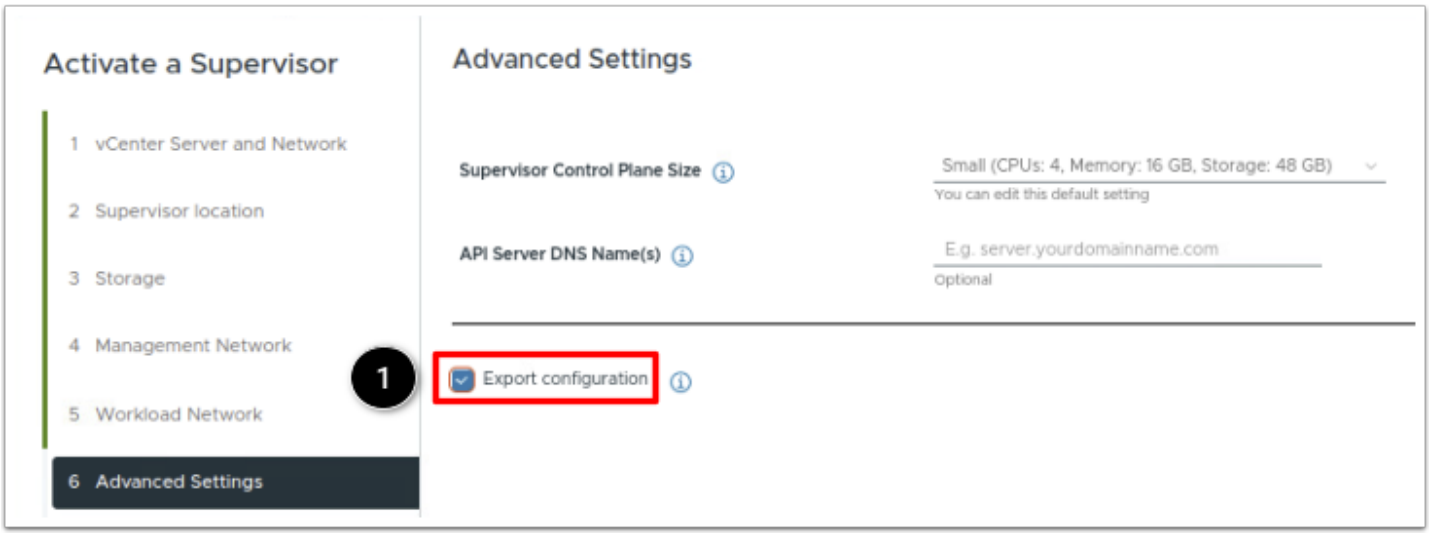


We need to enter the Network values for the Workload Network. These IP addresses will be used for the VKS clusters and vSphere PODs.

These static IP addresses are specific to this HOL lab.

1. Enter the **DNS Server:** 10.1.1.1
2. Enter the **Ingress CIDRs:** 10.2.0.0/24
3. Enter the **Egress CIDRs:** 172.16.200.64/24
4. Select **Next**(not shown)

Configure Advanced Settings

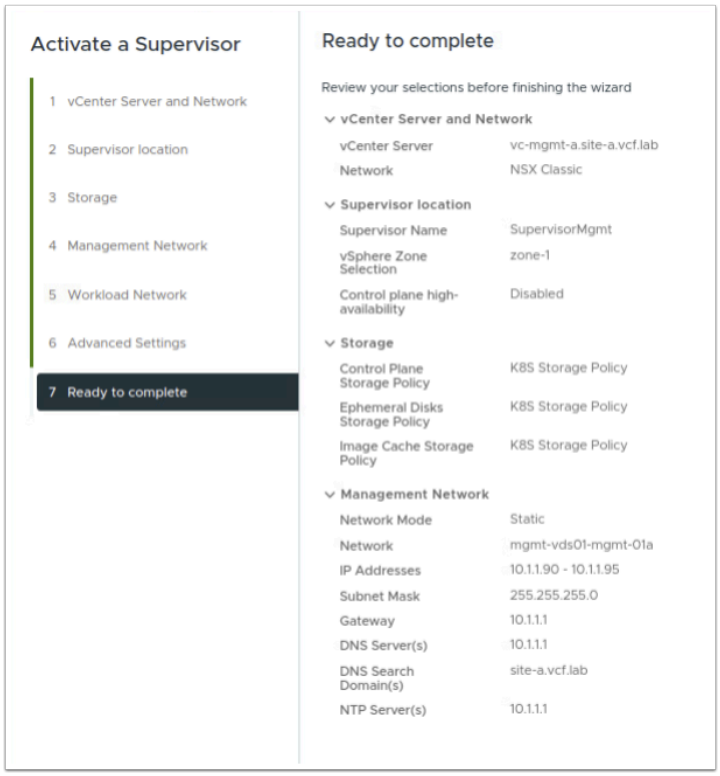


For this lab module we will select the small size. If you select the drop down arrow, you can see the different control plane sizes. Be sure to select the size that fits best for your applications.

1. Select the **Export configuration** check box.
1. Select **Next**(not shown)

A json file will be downloaded. This file can be modified and imported on a future supervisor deployment

Review Summary



On the Summary page

1. Select **Cancel**(not shown)

We are **not finishing this** install. We will be using the Supervisor already configured on the ***vc-wld01-a*** **vCenter**

Close the firefox browser by closing the vSphere Client tab for ***vc-mgmt-a.site-a.vcf.lab*** **vCenter.**

Deploy configure a namespace

We will walk through the process of creating a vSphere Namespace on the Workload vCenter.

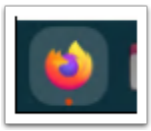
After creating a namespace, we will at least need to:

- Add VM Classes that will specify the size for the VMs we will be able to create (standalone or VKS VMs)
- Add Storage policies for volumes to be consumed by them and their internal components such as PODs

NOTE: We will not set specific permissions or create Resource and Object quotas or limits for our namespace in this section.

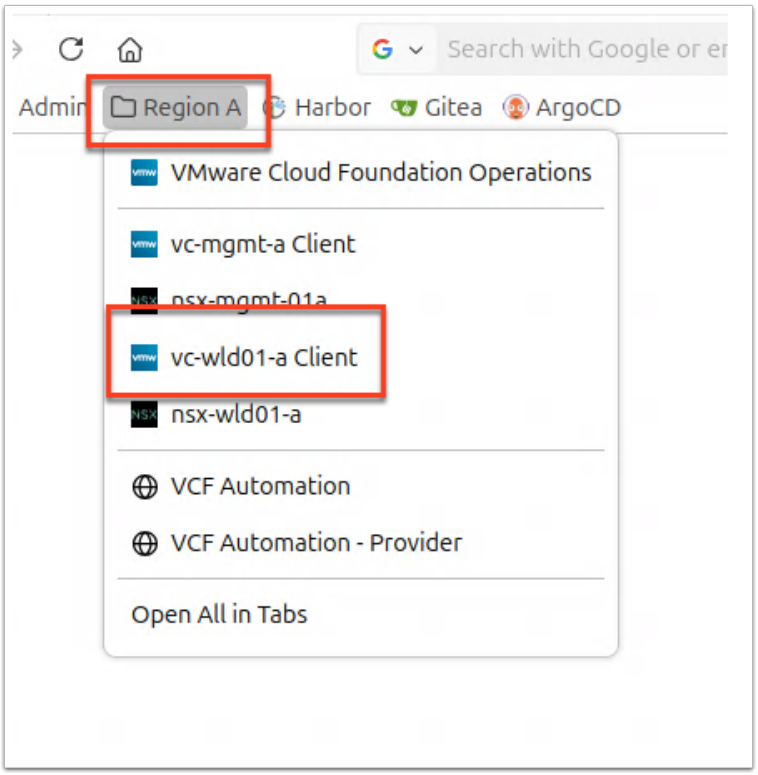
The next section will explain how to create and associate content libraries to our newly created namespace.

Open Firefox



1. Open Firefox from the taskbar at the bottom of the screen

Connect to the Workload vCenter (vc-wld01-a Client)



2. Select **Region A** tab
2. Select **vc-wld01-a Client** from the **Region A** tab

Login to vSphere

Log in with:

Username:

```
Administrator@wld.sso
```

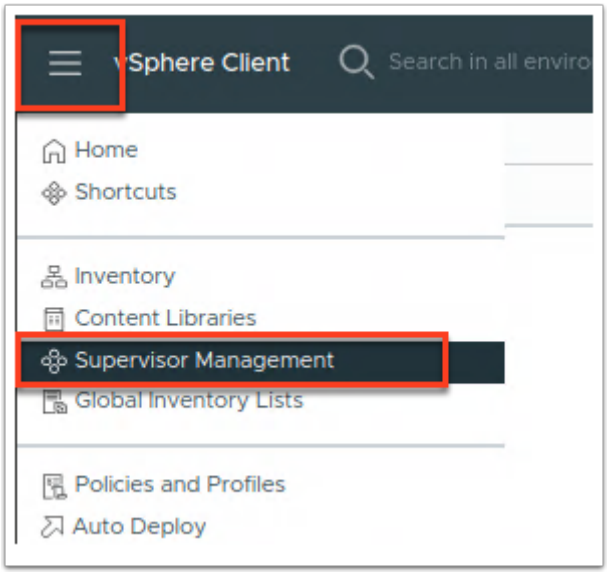
Click to copy

Password:

```
VMware123!VMware123!
```

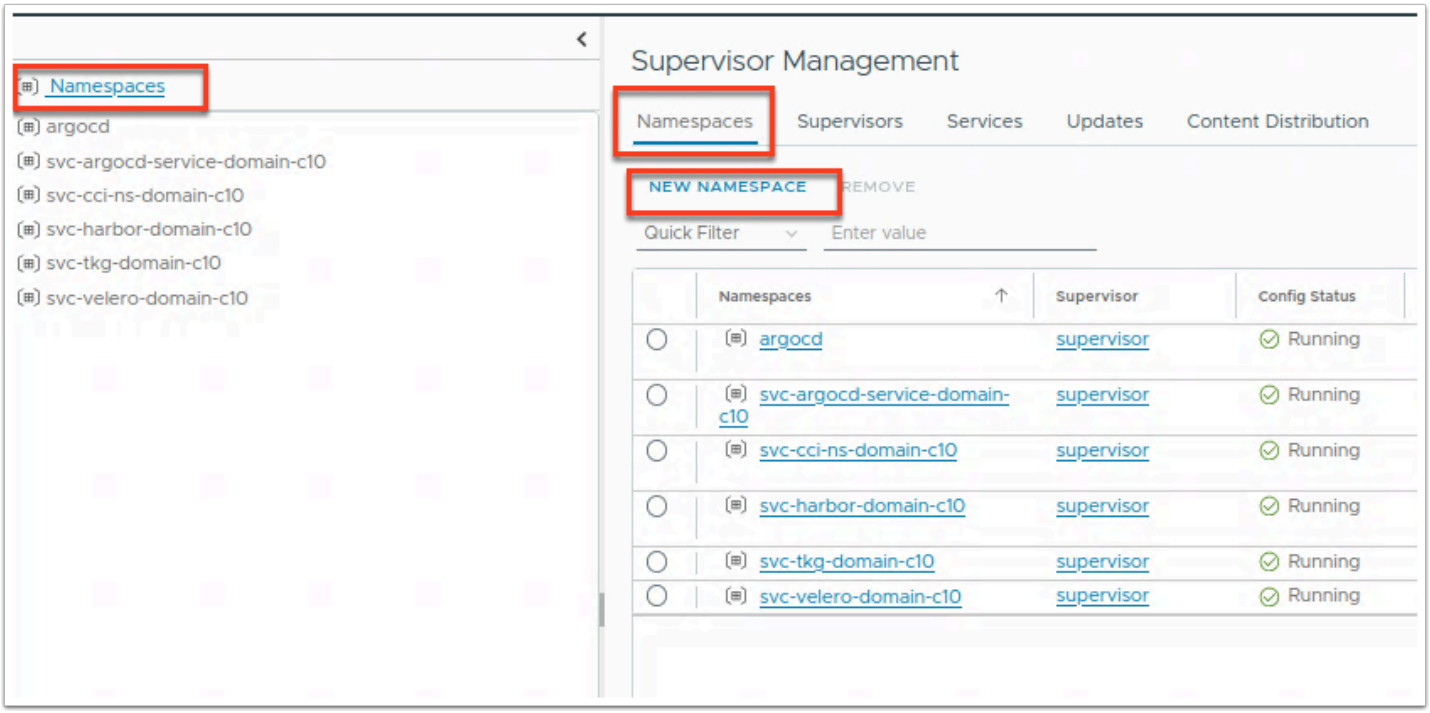
Click to copy

Create a vSphere Namespace



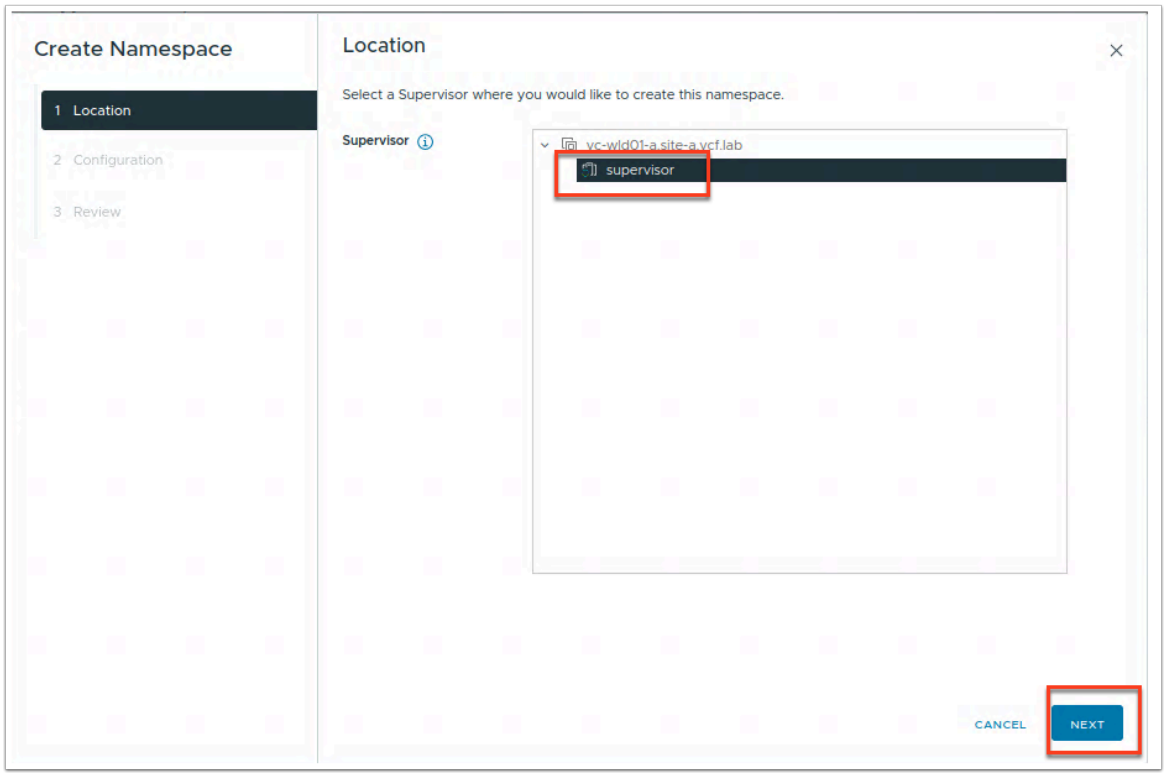
1. Select the **Menu** in the upper left corner
2. Select **Supervisor Management**

Select Namespace



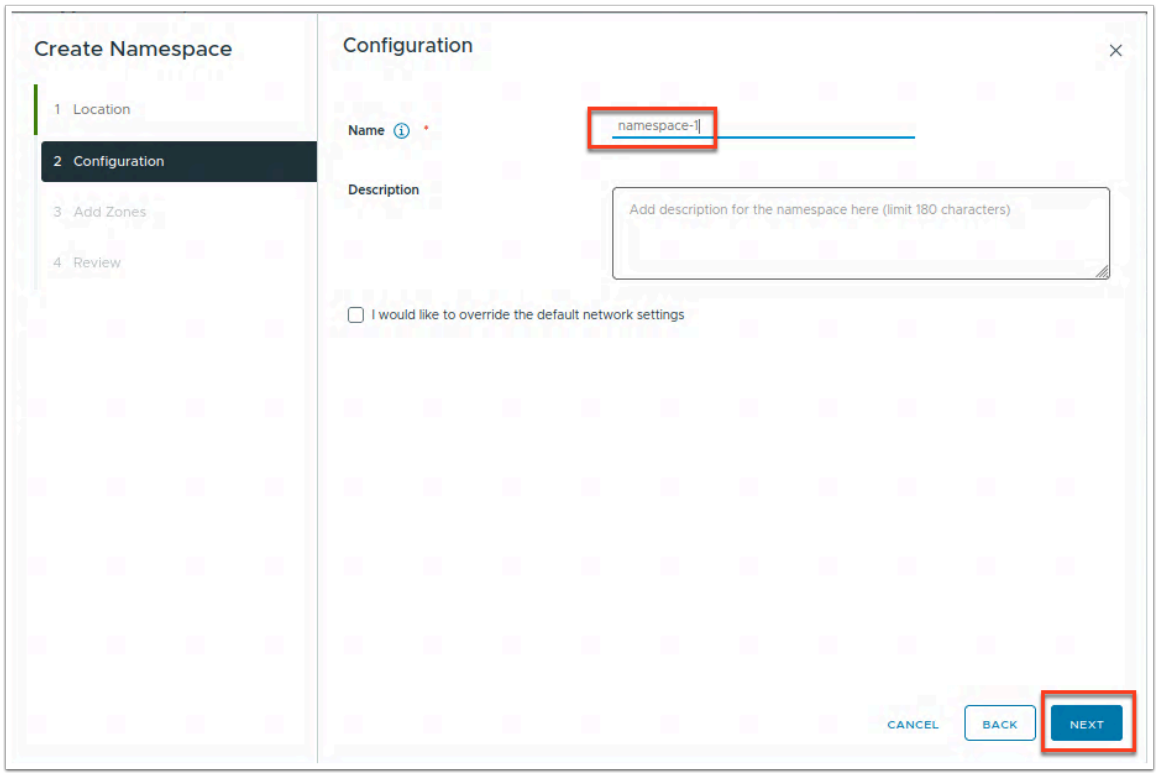
1. Select the main **Namespaces** element
2. Select the **Namespaces** tab
3. Click **NEW NAMESPACE**

Select the Supervisor For The Namespace



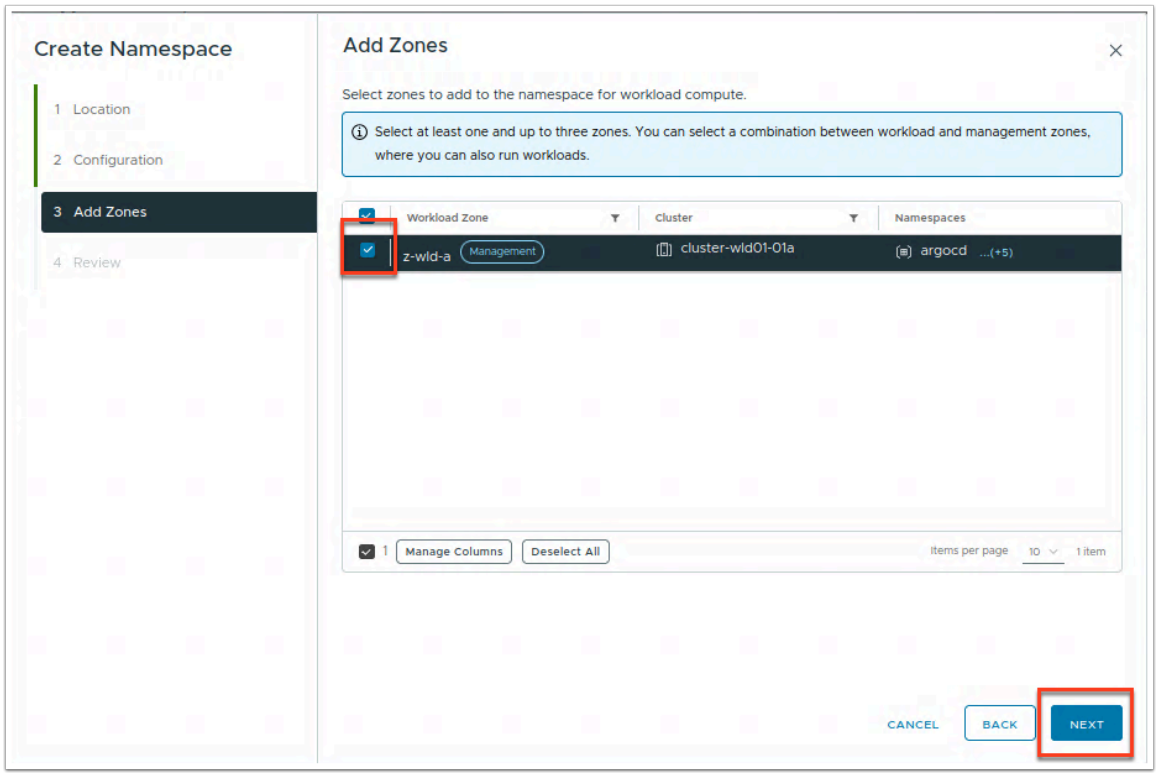
1. In the Create Namespace window, select ***Supervisor***
2. Click **NEXT**

Create a Name For The New Namespace



1. Type the name of your namespace. In this case, please enter **namespace-1**. There are some rules to follow when naming your namespace. To see those rules click the blue “i” next toName.
2. Click **NEXT**

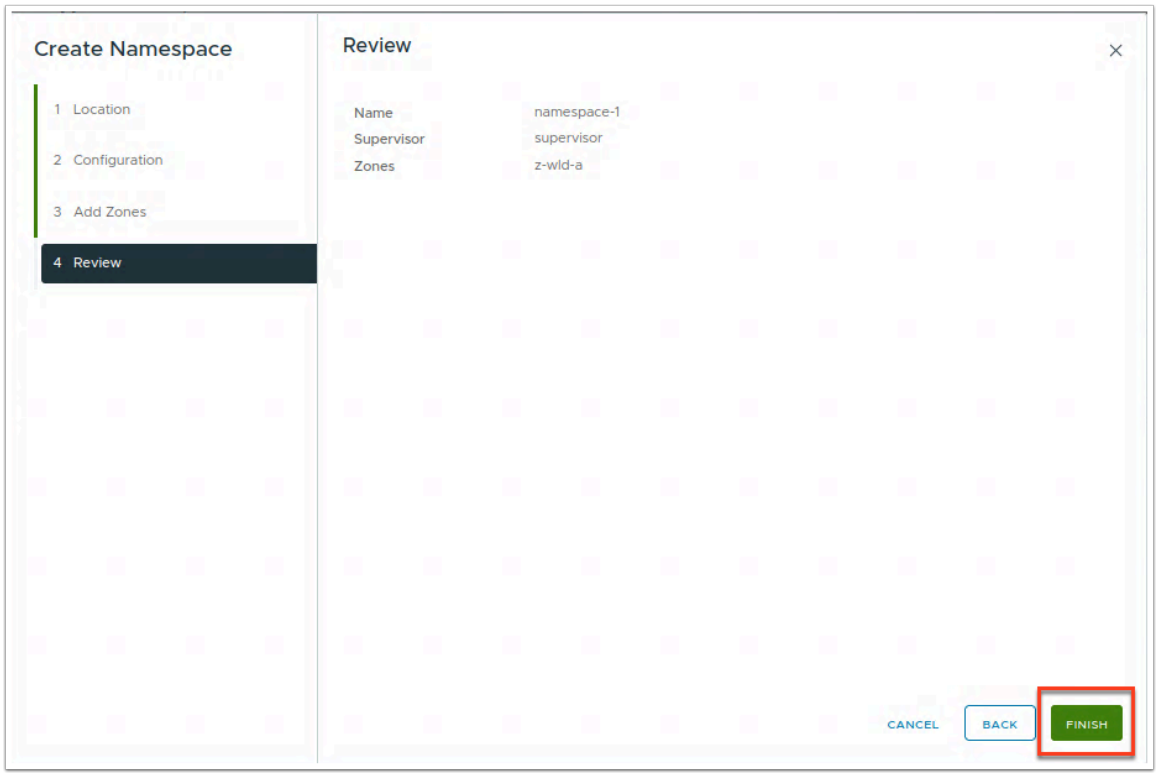
Add Zones



We need to set up at least 1 vSphere Zone to be able to create our Namespace.

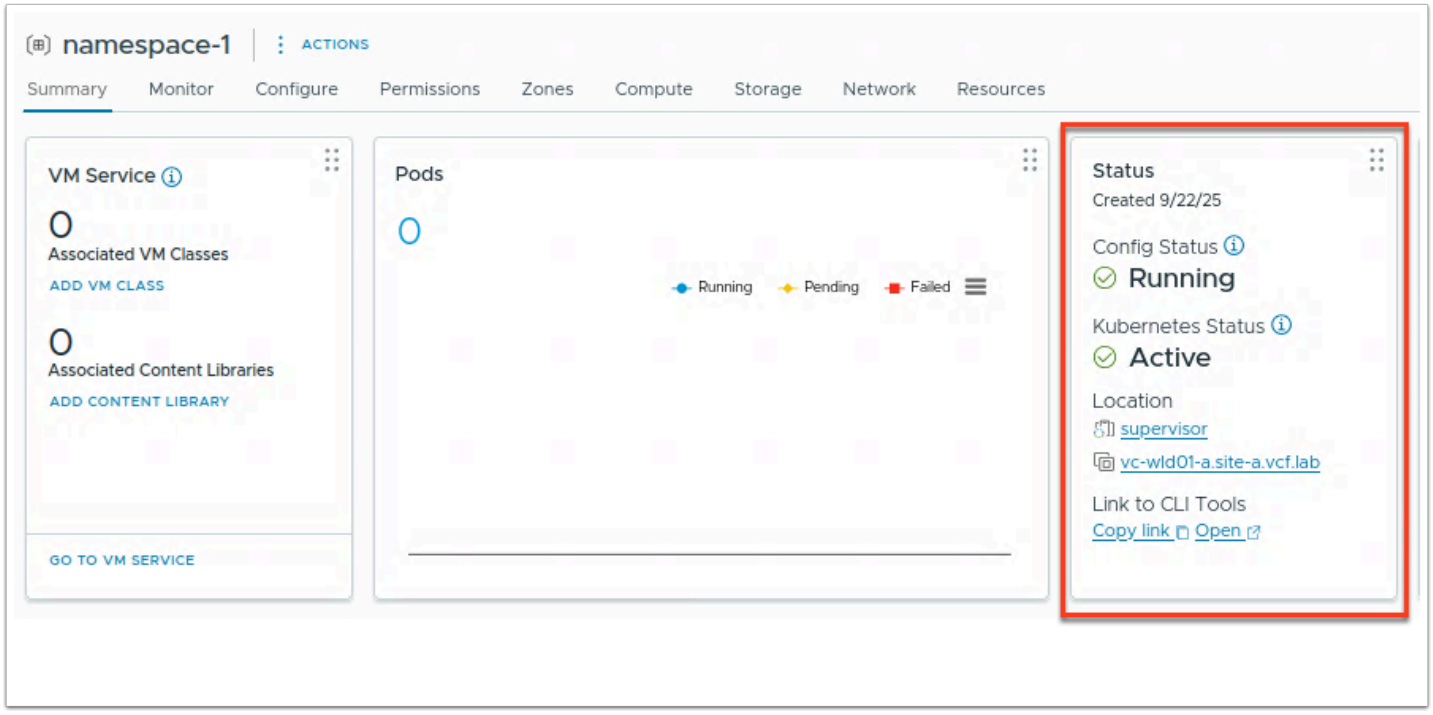
1. Select Radio check box for **Workload Zone: *z-wld-a***
2. Click **NEXT**

Review And Accept



1. Review the configuration and click **FINISH**

View The New Namespace

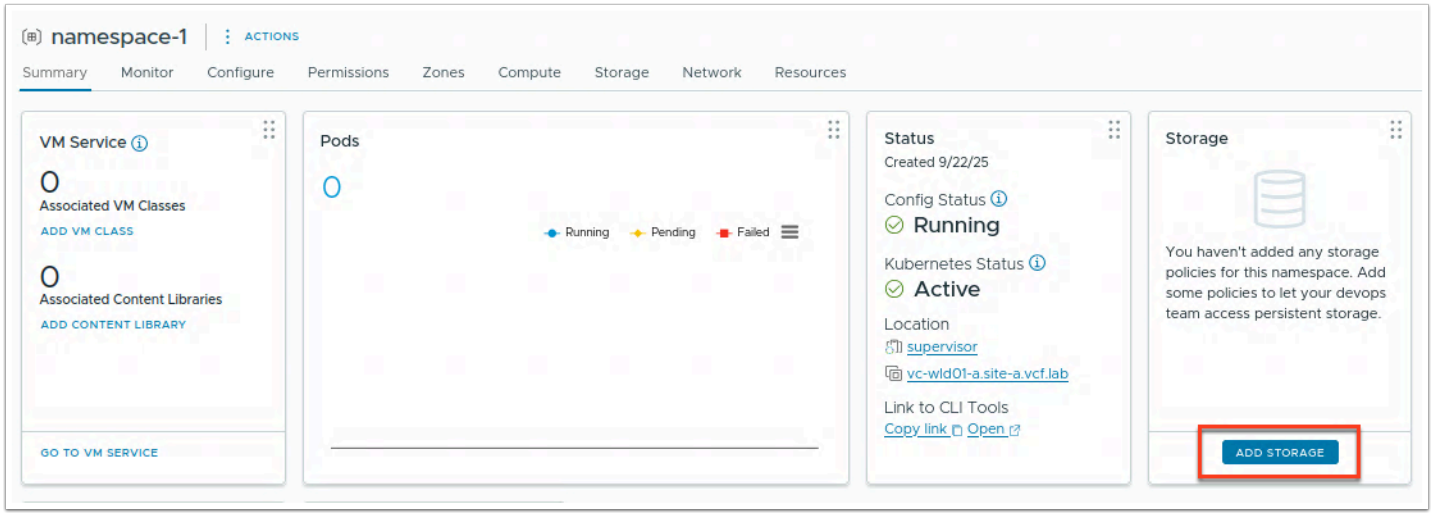


1. In the new **namespace-1** Summary page, check that the Status is **Running and Active**. The status may show Creating for a minute or two.

Configure a vSphere Namespace

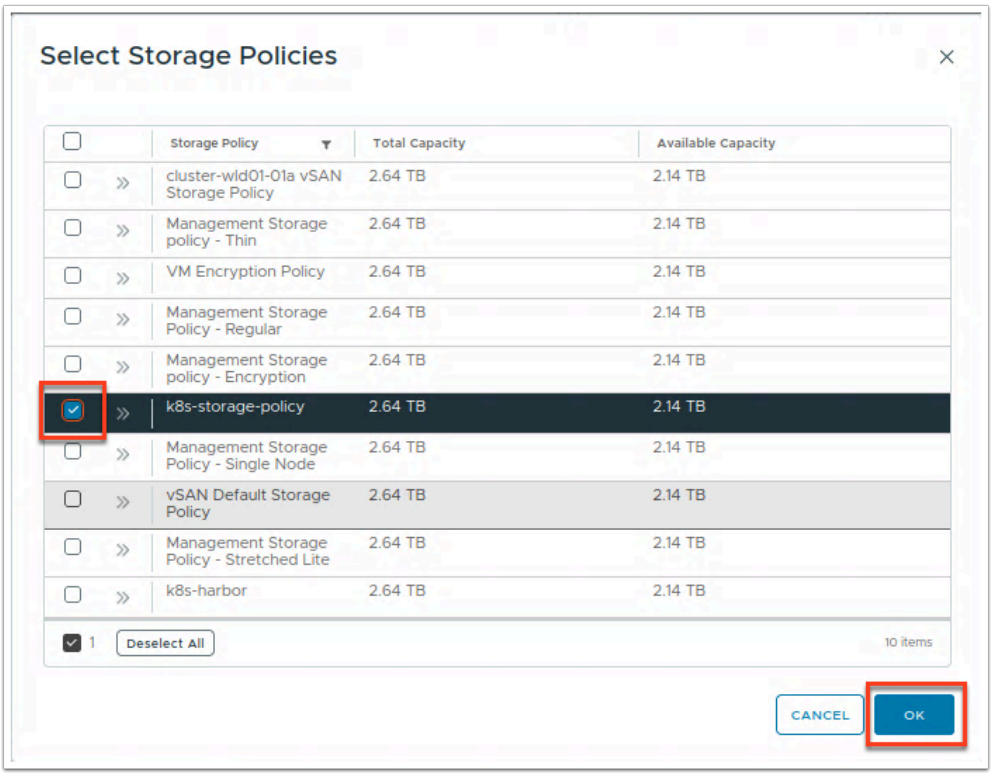
After having created our first namespace **namespace-1**, we will now associate VM Classes and Storage Policies.

Add a Storage Policy



1. In the Supervisor Management main Namespaces page, select **namespace-1** in the list
2. Click the **Summary** tab
3. Click the **ADD STORAGE** button

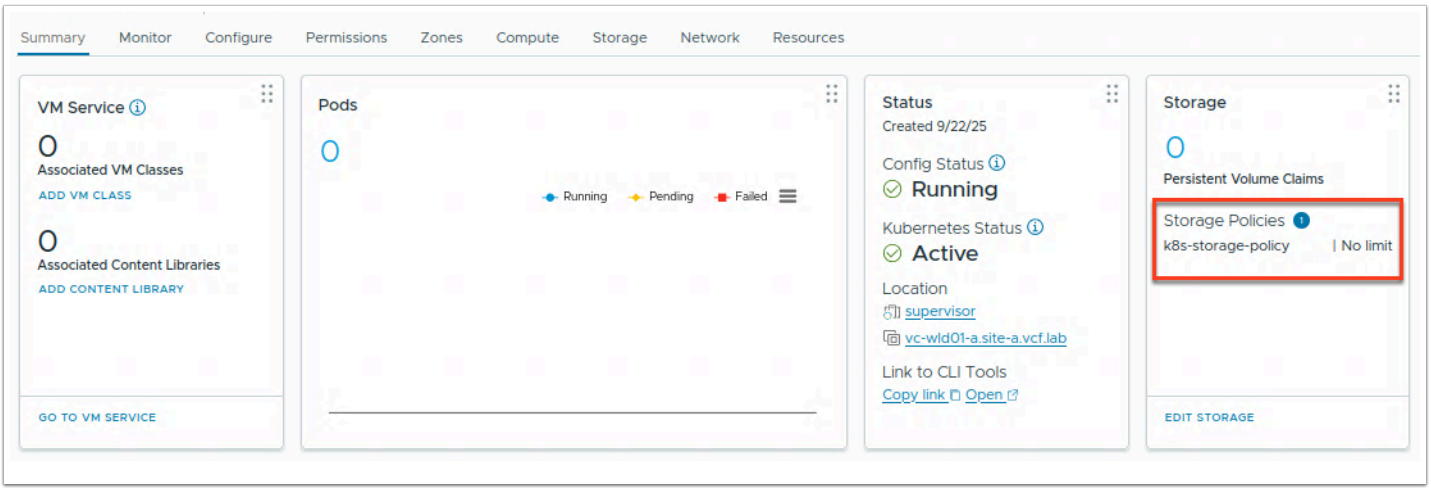
Select a Storage Policy



For the sake of simplicity, we will only select one storage policy, but you can select as many as you want to make them available for your workload volumes.

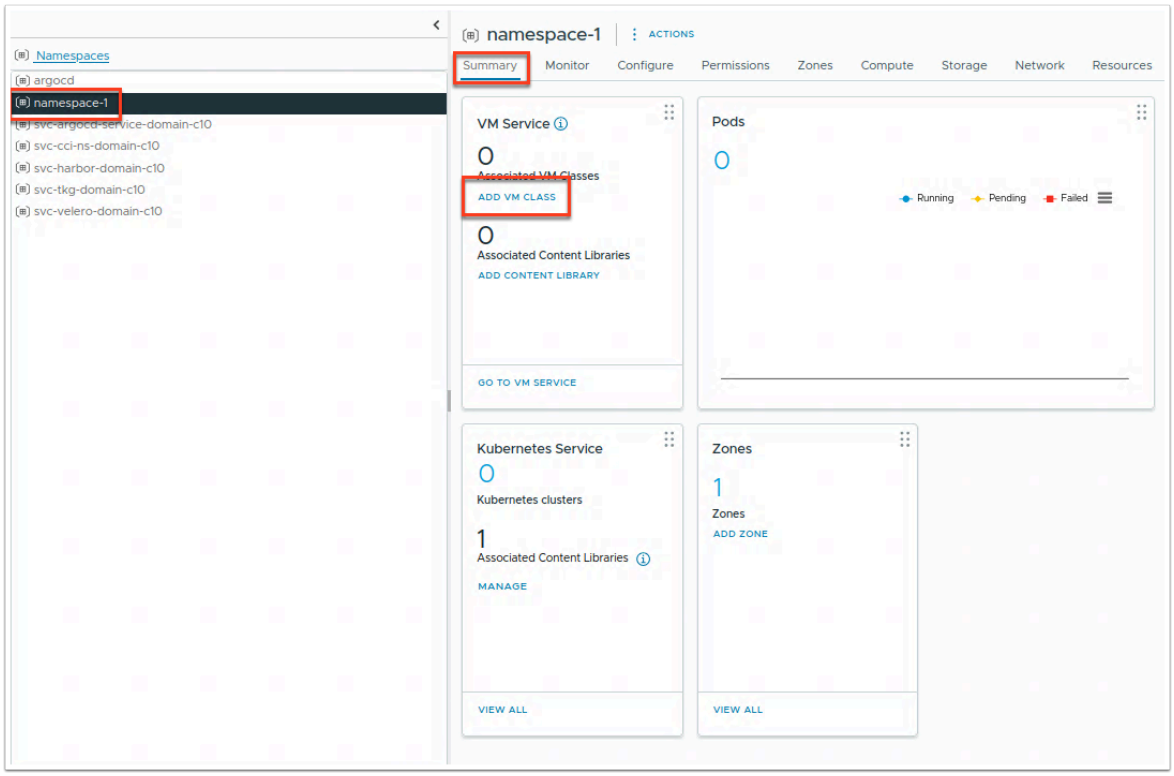
1. Select the ***k8s-storage-policy*** policy
2. Click **OK**

Review The Newly Created Storage Policy



1. Now, you will just need to check if the selected Storage Policy, ***k8s-storage-policy*** in our case, is listed in theStorage block on the **namespace-1** summary page Note it may take a moment for the added storage to display in the Storage window.

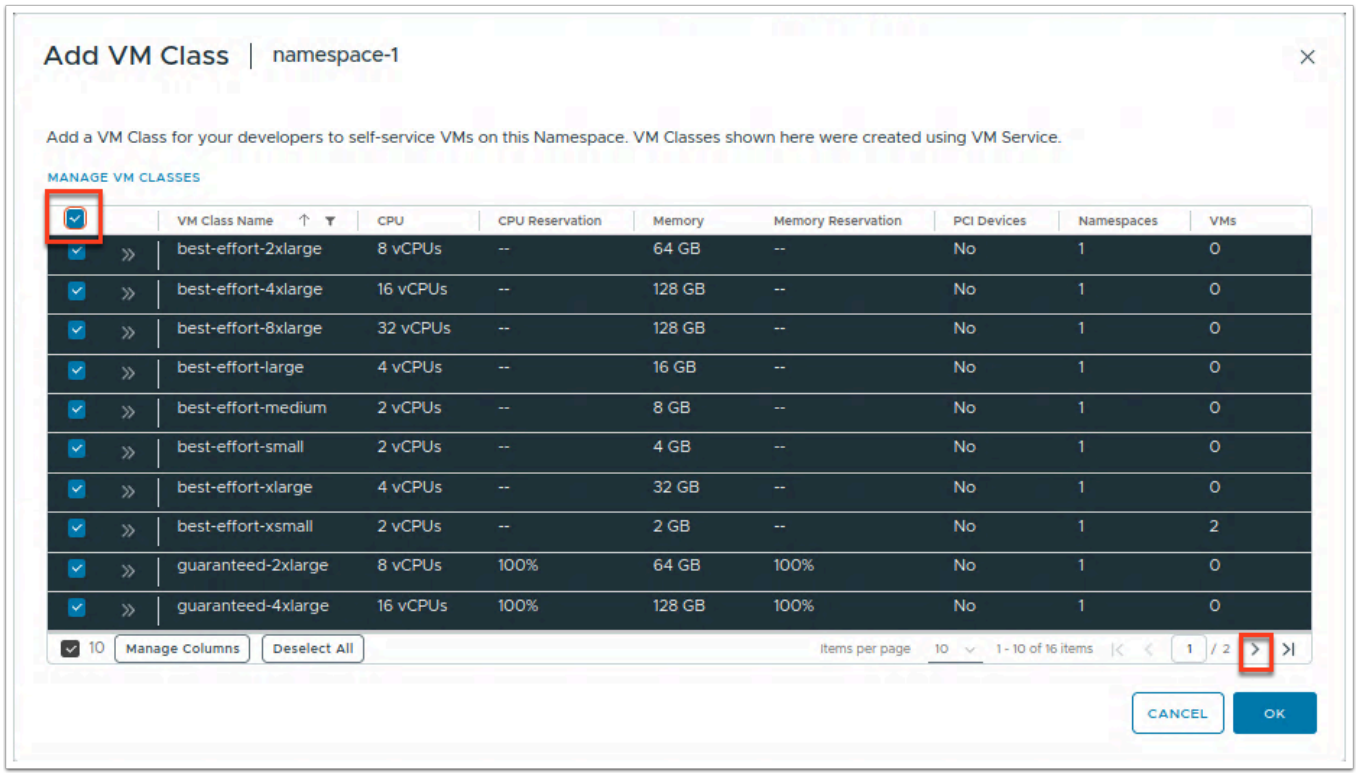
Add VM Classes to The Namespace



To add VM Classes that will specify the available sizes for our Namespace VMs, we will need to follow the next steps:

1. In the **Supervisor Management** main **Namespaces** page, select **namespace-1** in the list
2. Click the **Summary** tab
3. Click the **ADD VM CLASS** button. You may need to scroll the page down a bit to find it

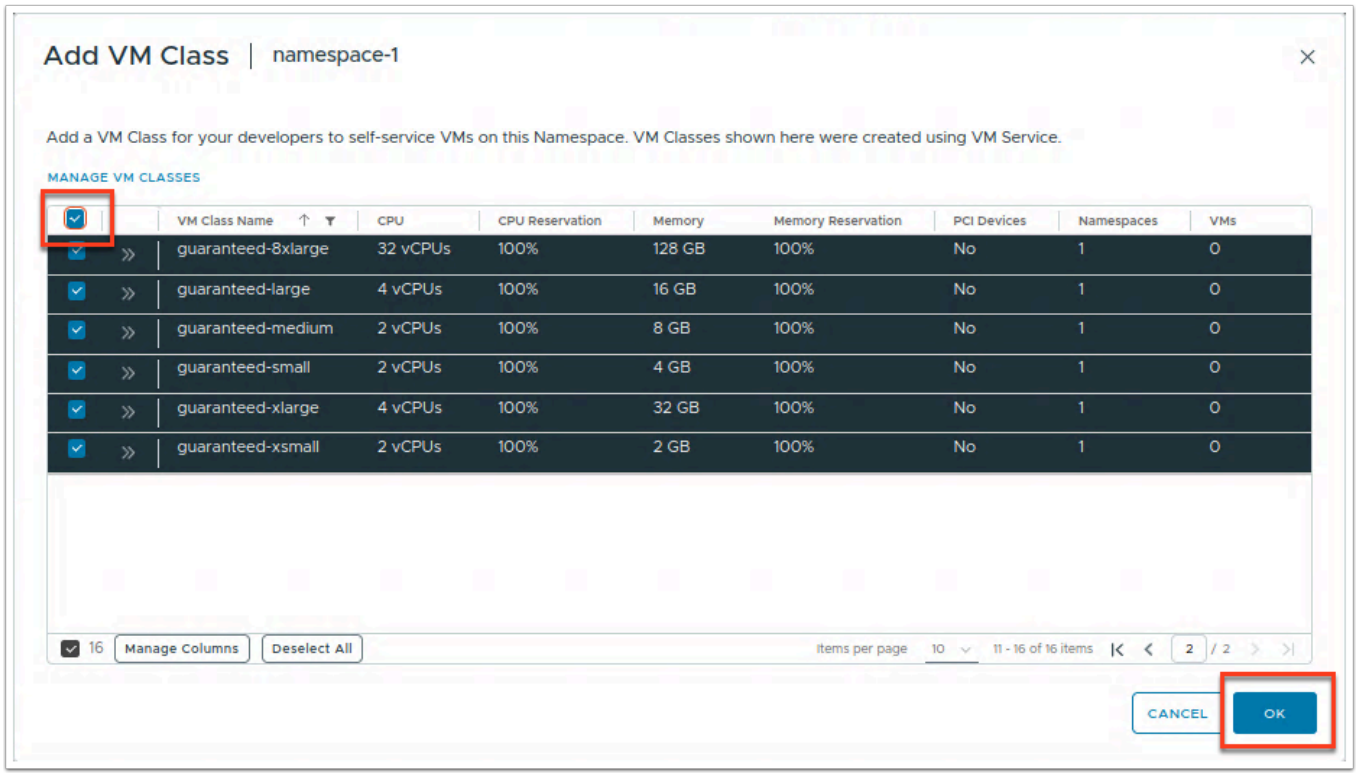
Select The VM Classes For This Namespace



You can select as many VM Classes as you want. In our case, we will select all the available VM Classes:

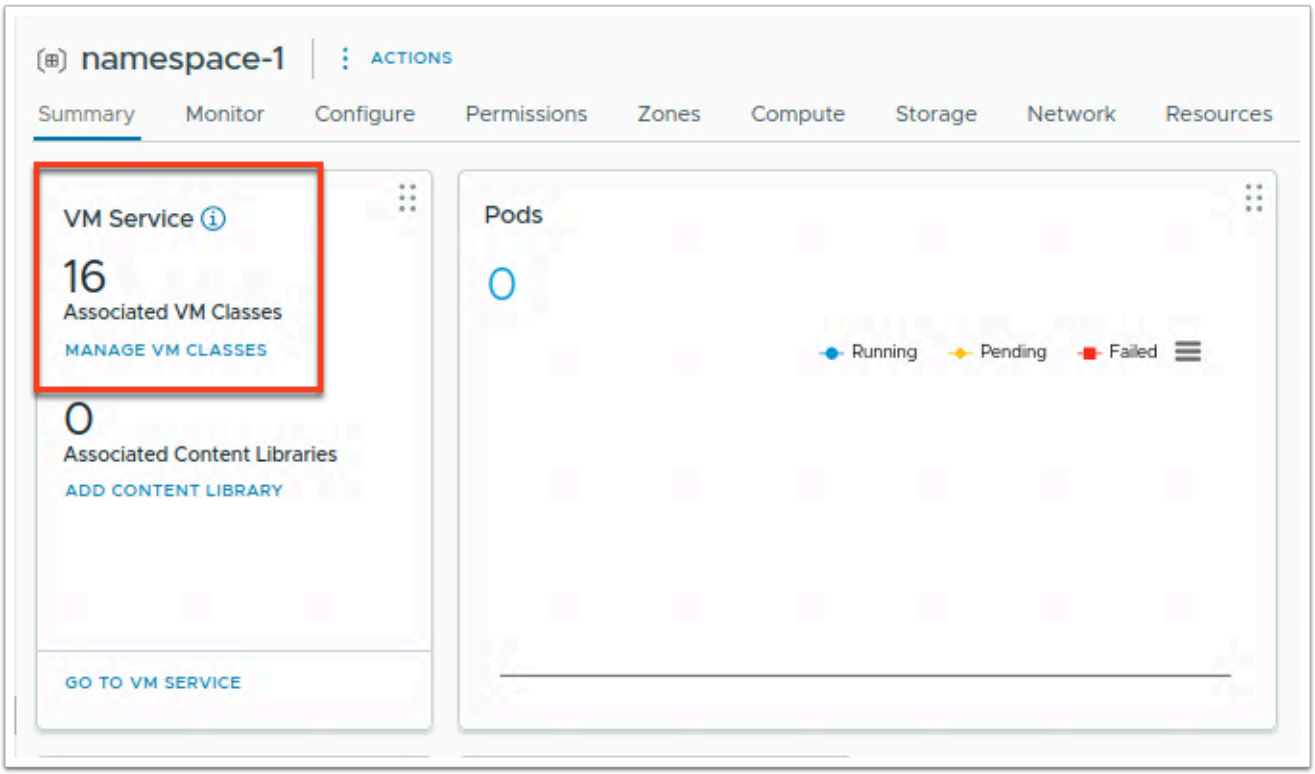
1. Click on the check box on the top left corner of the **VM Classes** table
2. Click the right arrow icon in the bottom right corner of the table to go to the second page

Select The VM Classes For This Namespace (continued)



1. Click on the check box on the top left corner of the second page of **VM Classes** table again, to finish selecting the rest of available VM Classes
2. Click **OK**

View The VM Classes Added to The Namespace

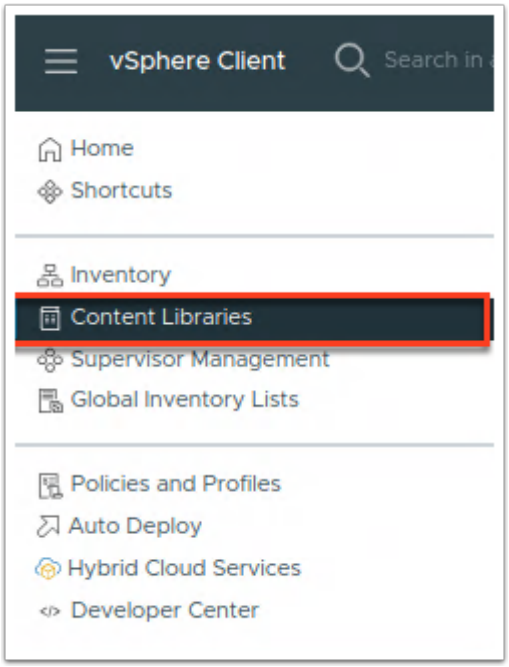


1. After finishing selecting the VM Classes, check in the **namespace-1 Summary** paget he number of associated VM Classes. Again it may take a moment for the newly added VM Classes to appear in the VM Service tile.

Configure a Content Library

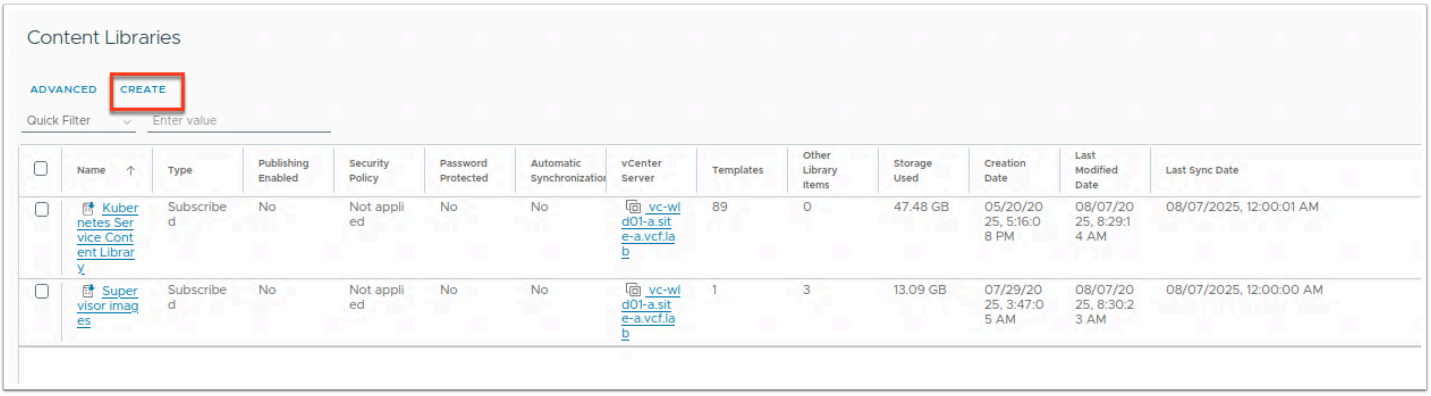
Next, we will create a new Content Library for deploying VMs through the VM Service. We will then associate that new Content Library with the Namespace created above.

Open Content Libraries



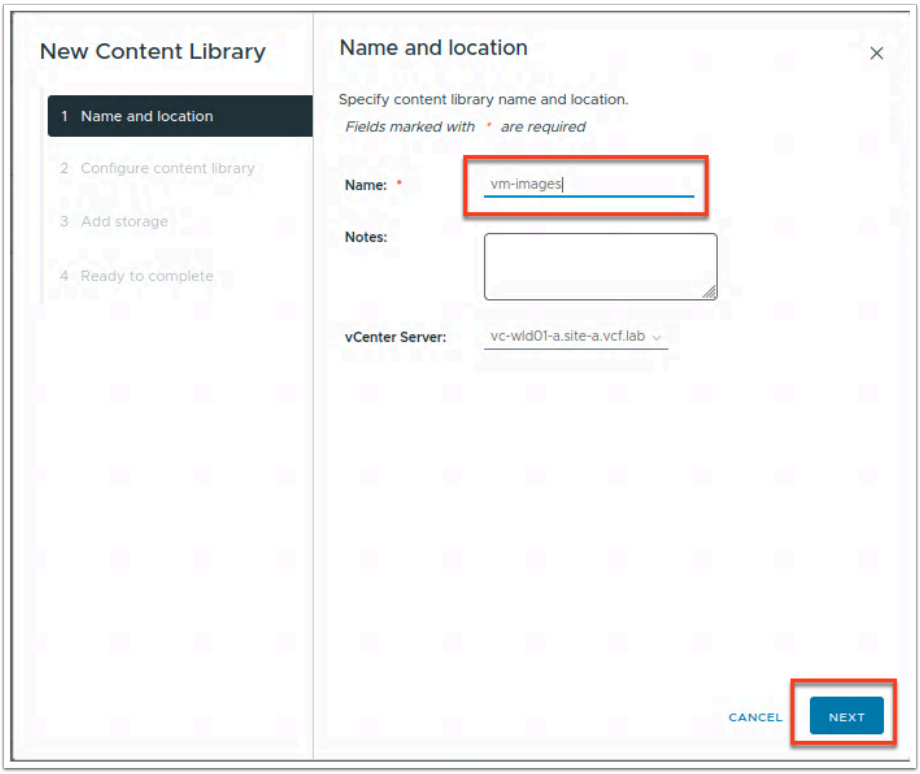
1. Select **Content Libraries** from the Menu drop down

Create a New Content Library



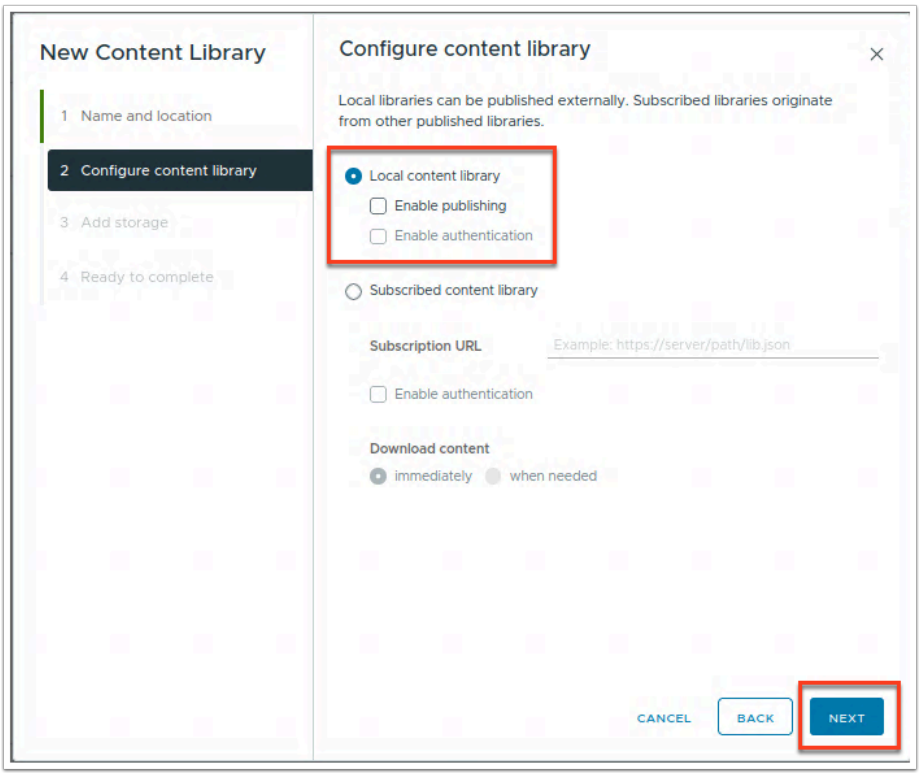
1. Click **CREATE**.

Specify the Name And Location



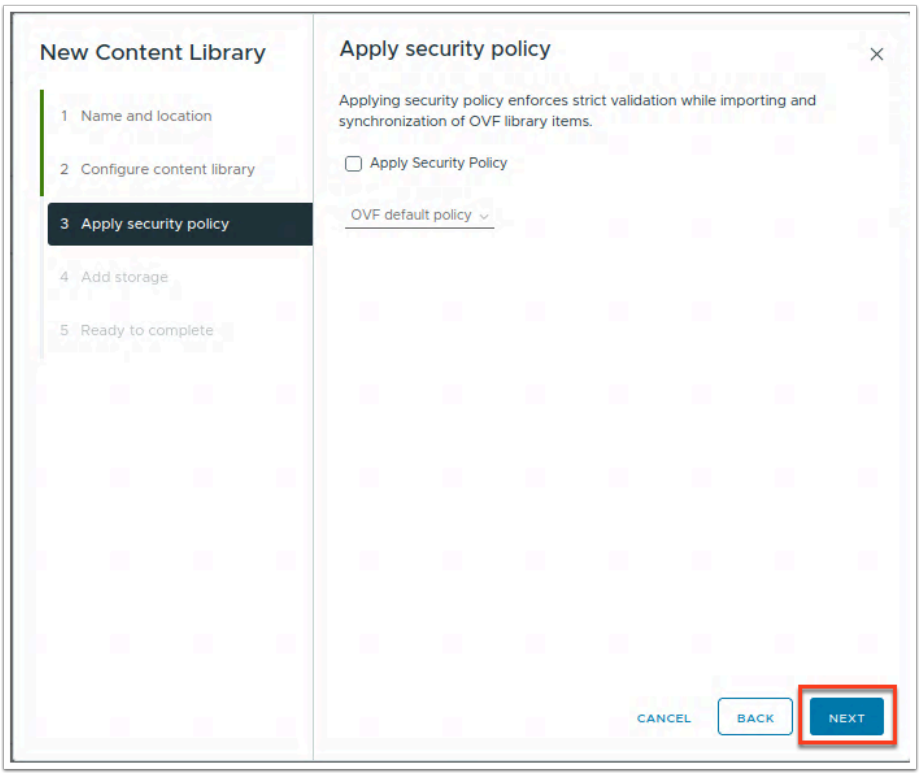
1. Name the content library, use ***vm-images*** for the name.

Specify as Local Content Library



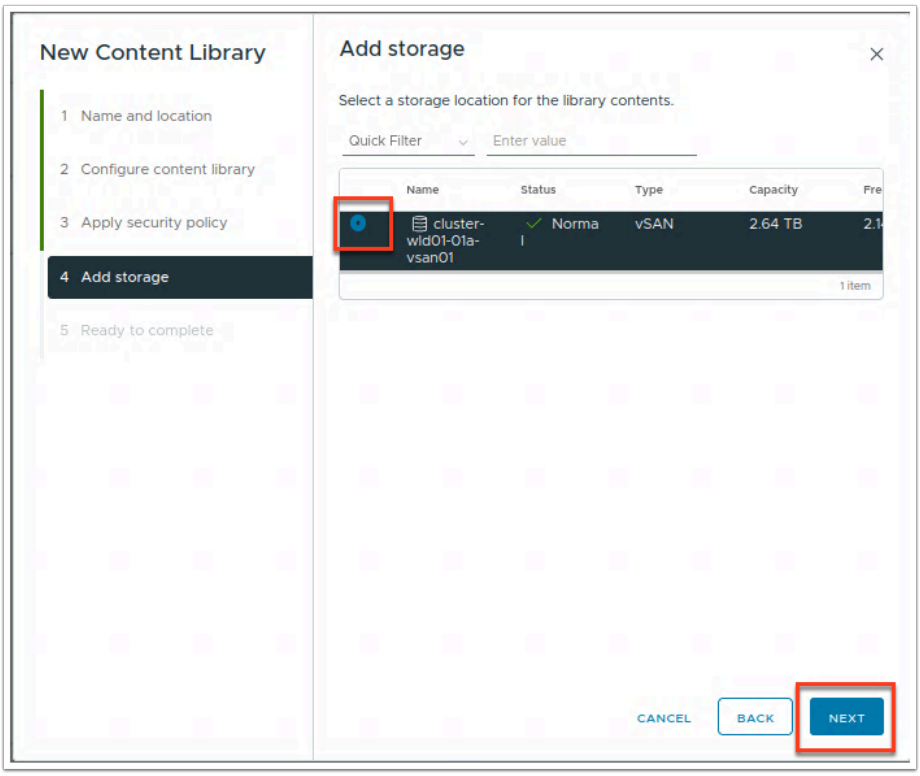
1. If not already selected, select use **Local content library** and hit **NEXT**.

Apply Security Policy



1. Accept the defaults and click NEXT.

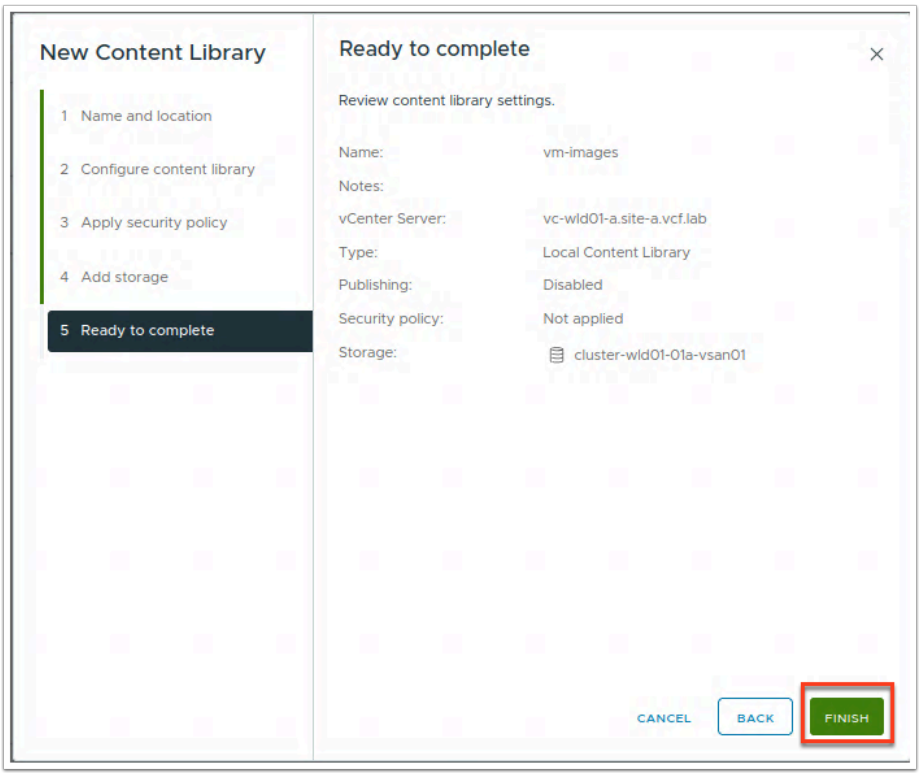
Add Storage For The Content Library



If it is not already selected, click the radio button to elect the ***cluster-wkld-01a-vsan01*** storage policy .

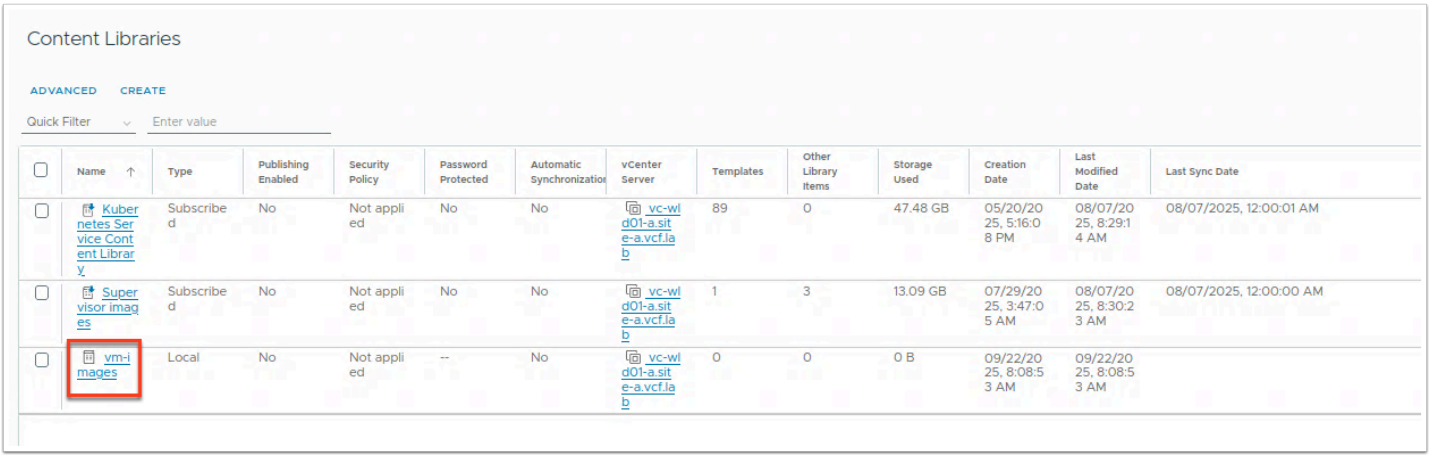
1. click **NEXT**.

Complete Adding The Storage Library



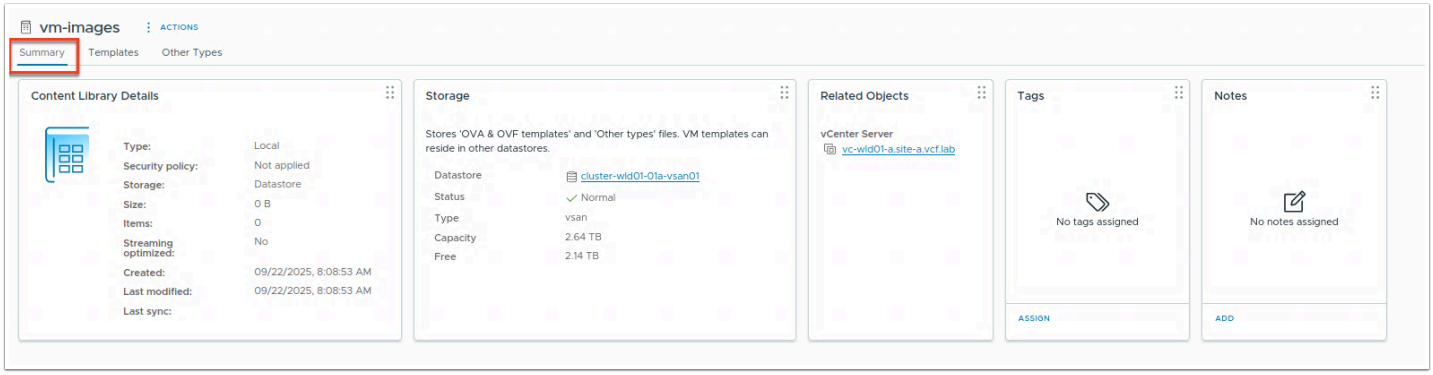
1. Click **FINISH** to add the storage library to the supervisor cluster.

Upload new .ova for VM Deployment to Content Library



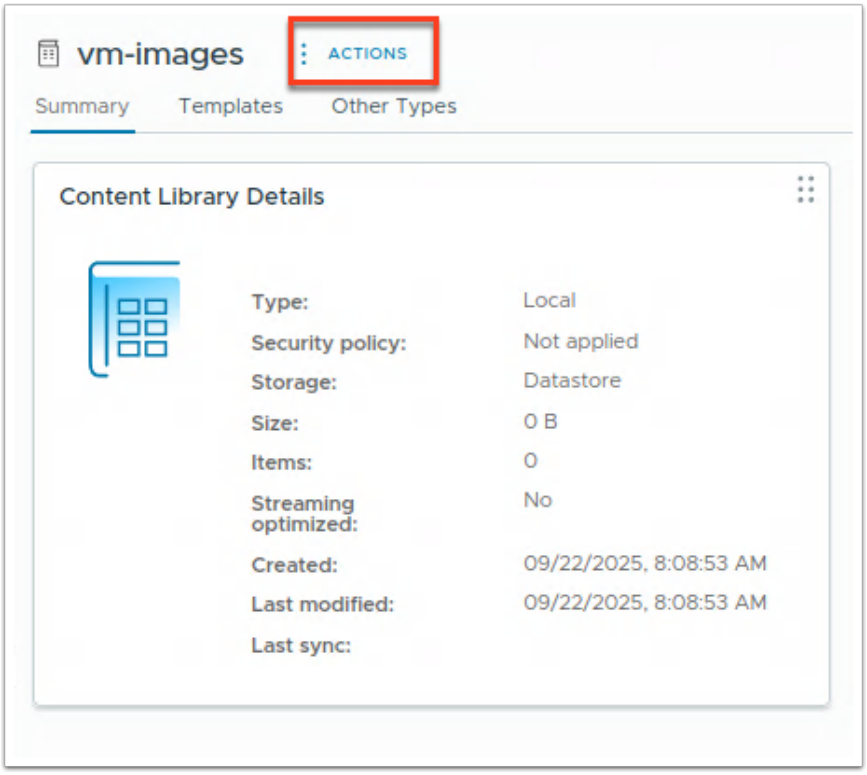
1. Click in the newly created content library, ***vm-images***.

Upload an Image to The vm-images Content Library



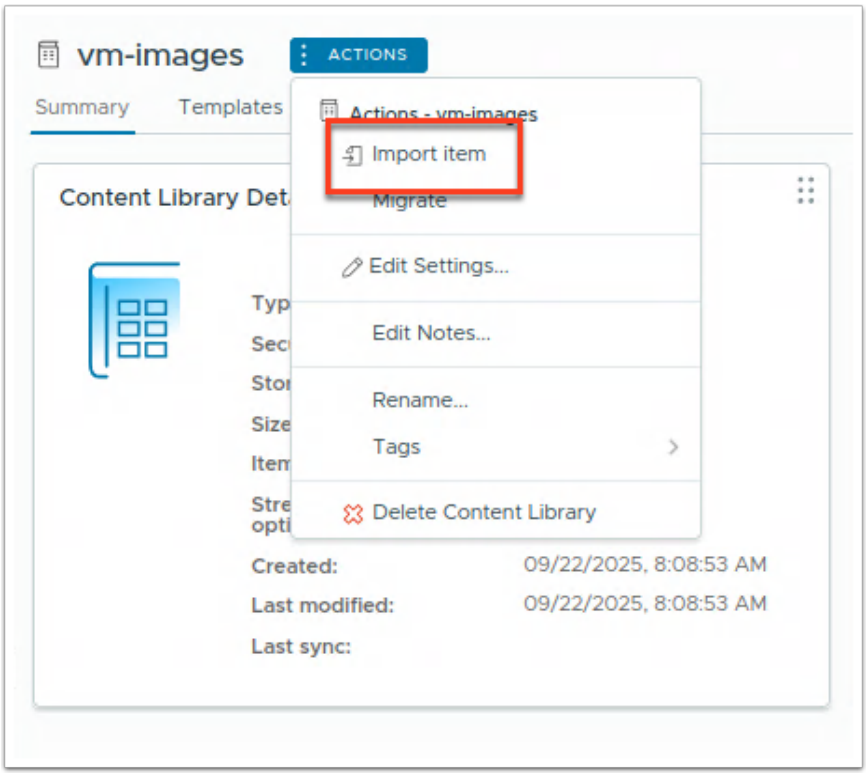
If the Summary tab is not selected, select the summary tab.

Select Actions



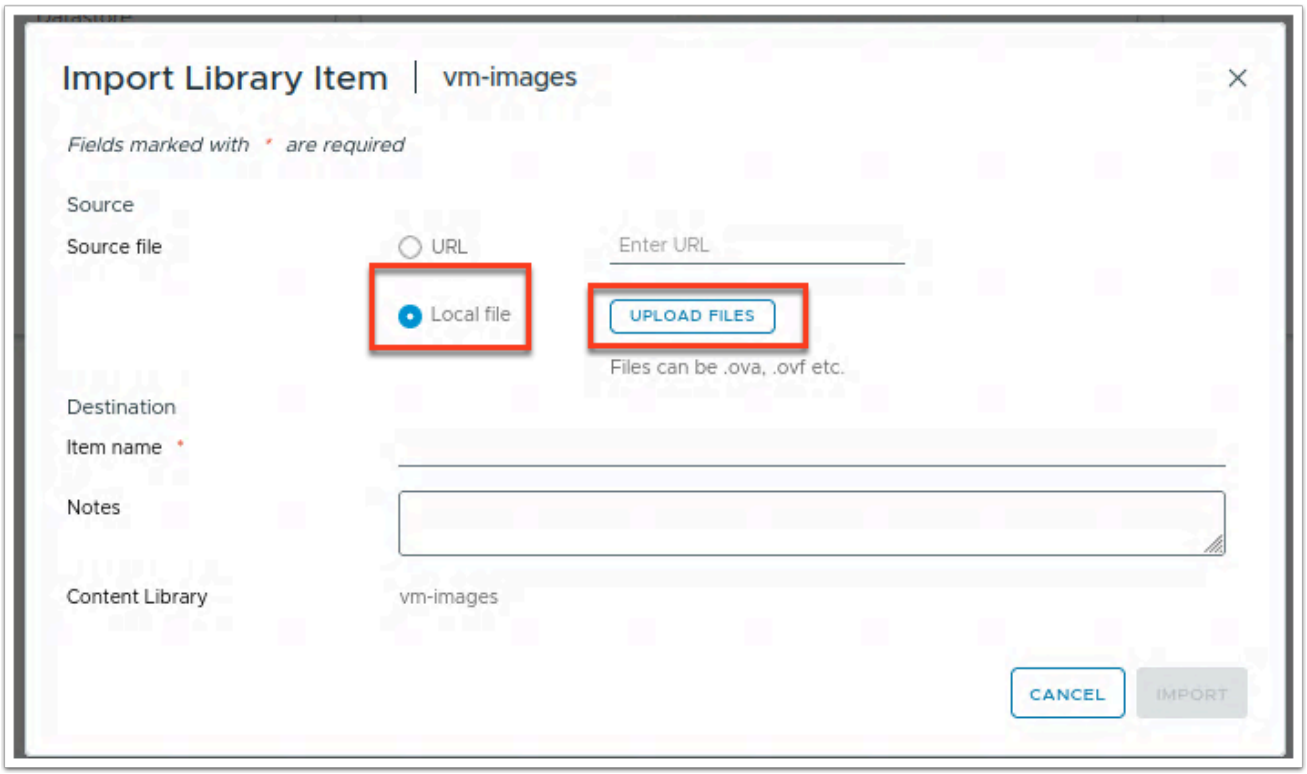
1. Click **Actions**.

Import a OVA FIle - Step 1



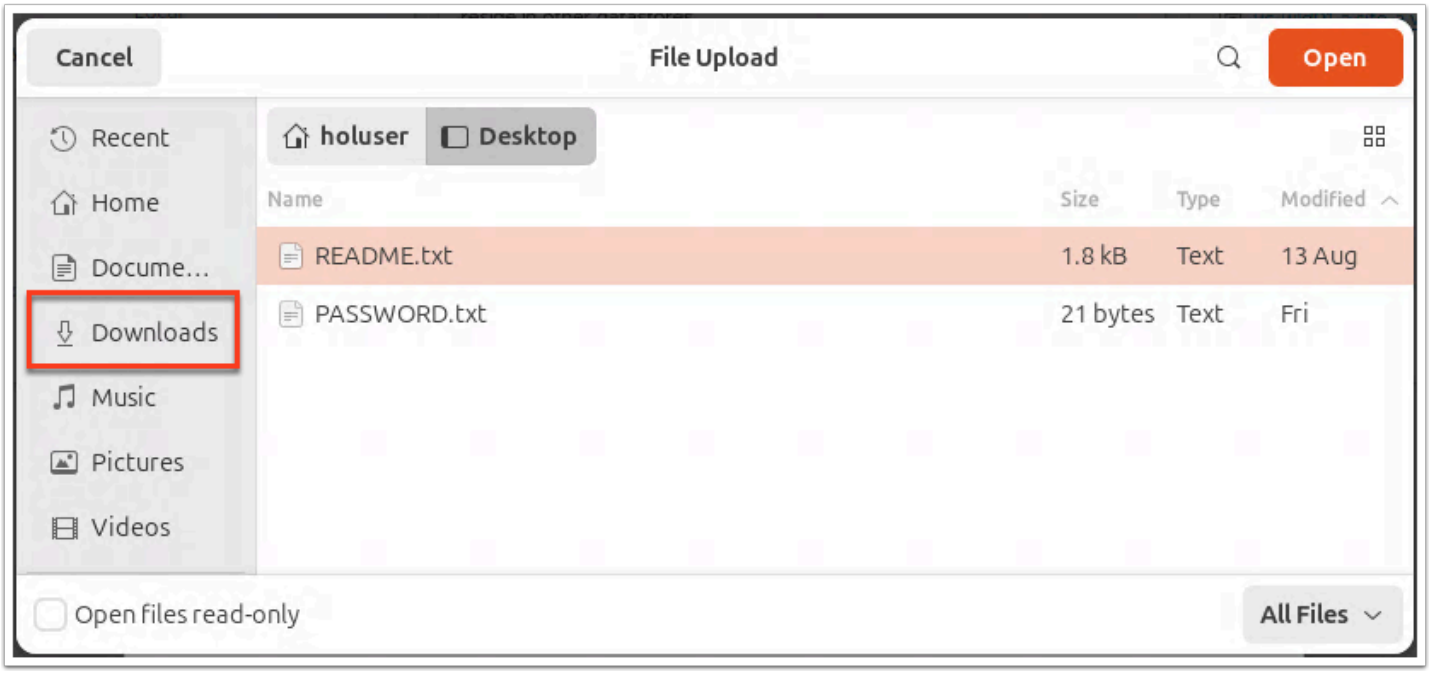
1. Select **Import Item**.

Select Local File



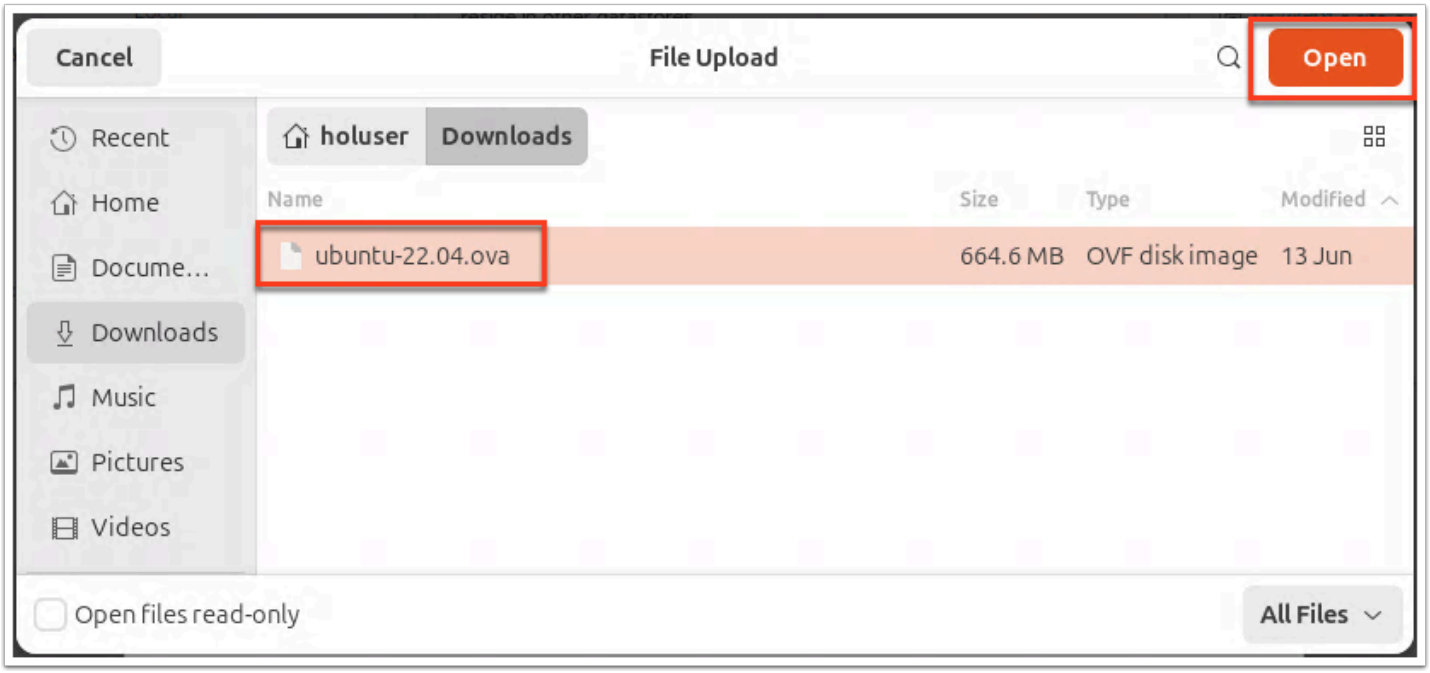
1. Click the radio button for Local File.
2. Click **UPLOAD FILES**.

Select FIle Image to Upload



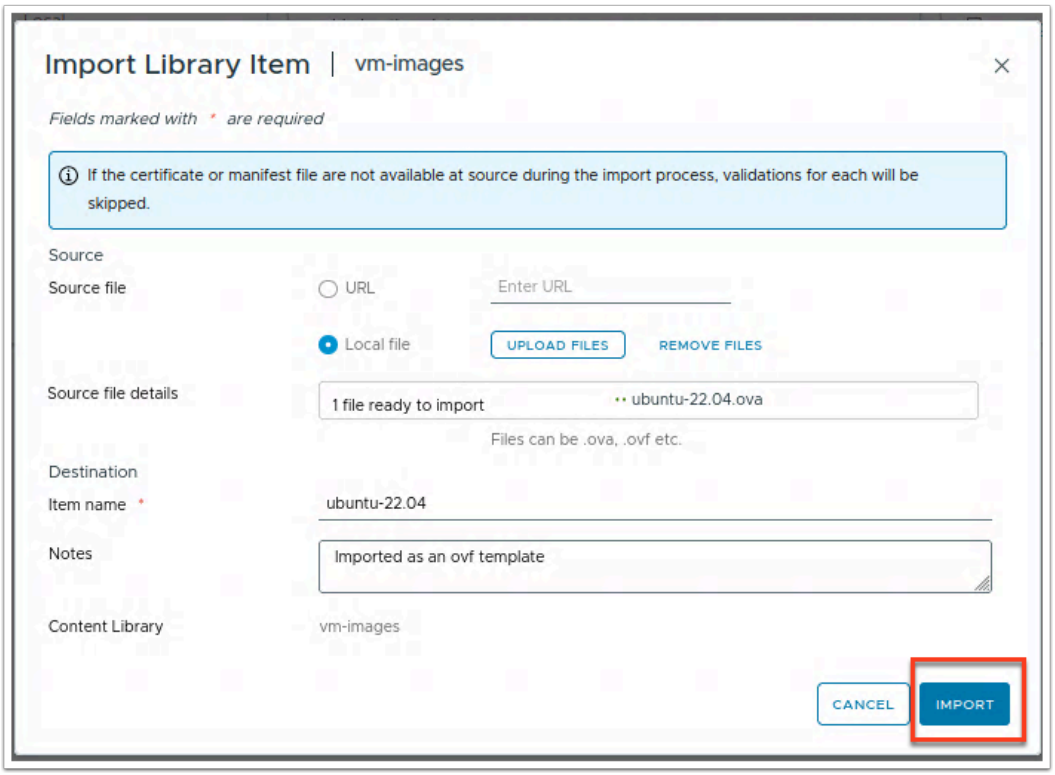
1. Select the **Downloads** folder.

Select Ubuntu Image



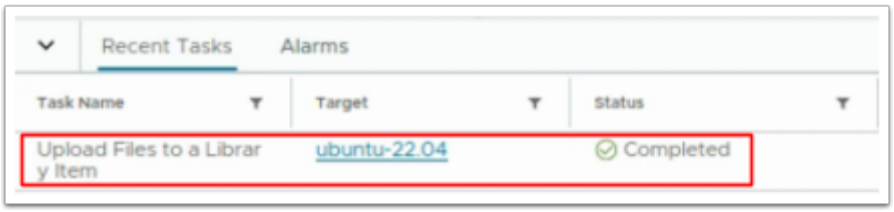
1. Select the ***ubuntu.22.04.ova*** image.
1. Click **OPEN**.

Import The Ubuntu Image to The Content Library



1. Click **IMPORT**.

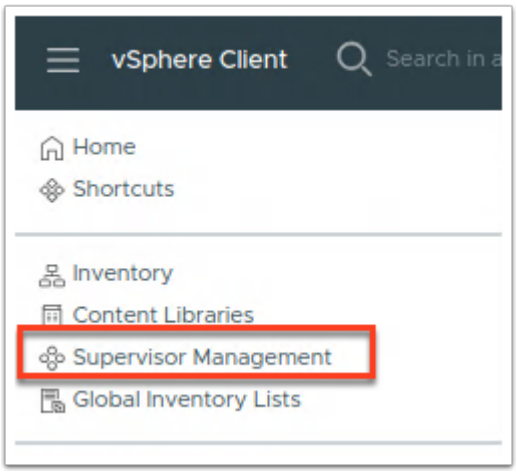
Monitor the Import



Looking at the **Recent Tasks** section, after some time you will see that the image has been imported successfully.

> You can move forward in the lab as the ova uploads

Navigate to Supervisor Management



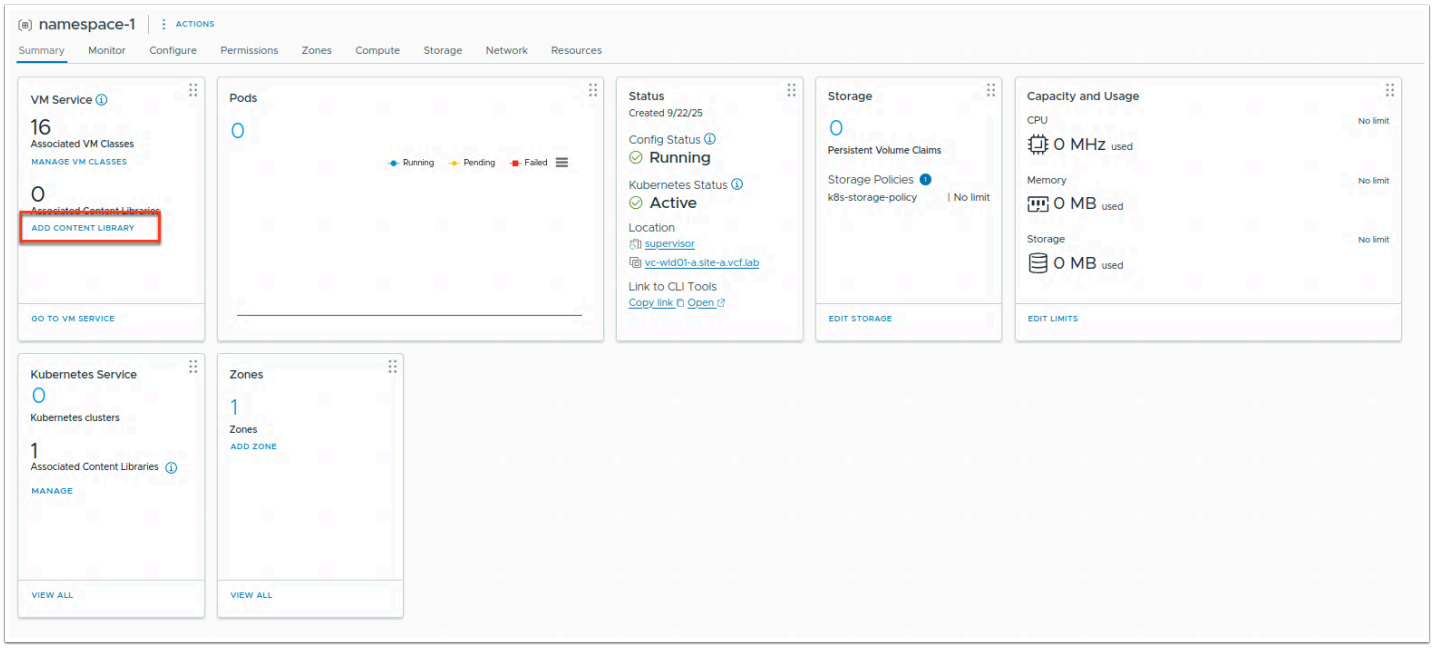
1. From the **vSphere Client** menu, select **Supervisor Management**.

Associate The Newly Created COntent Library With A Namespace

Ensure that the Namespaces tab is selected.

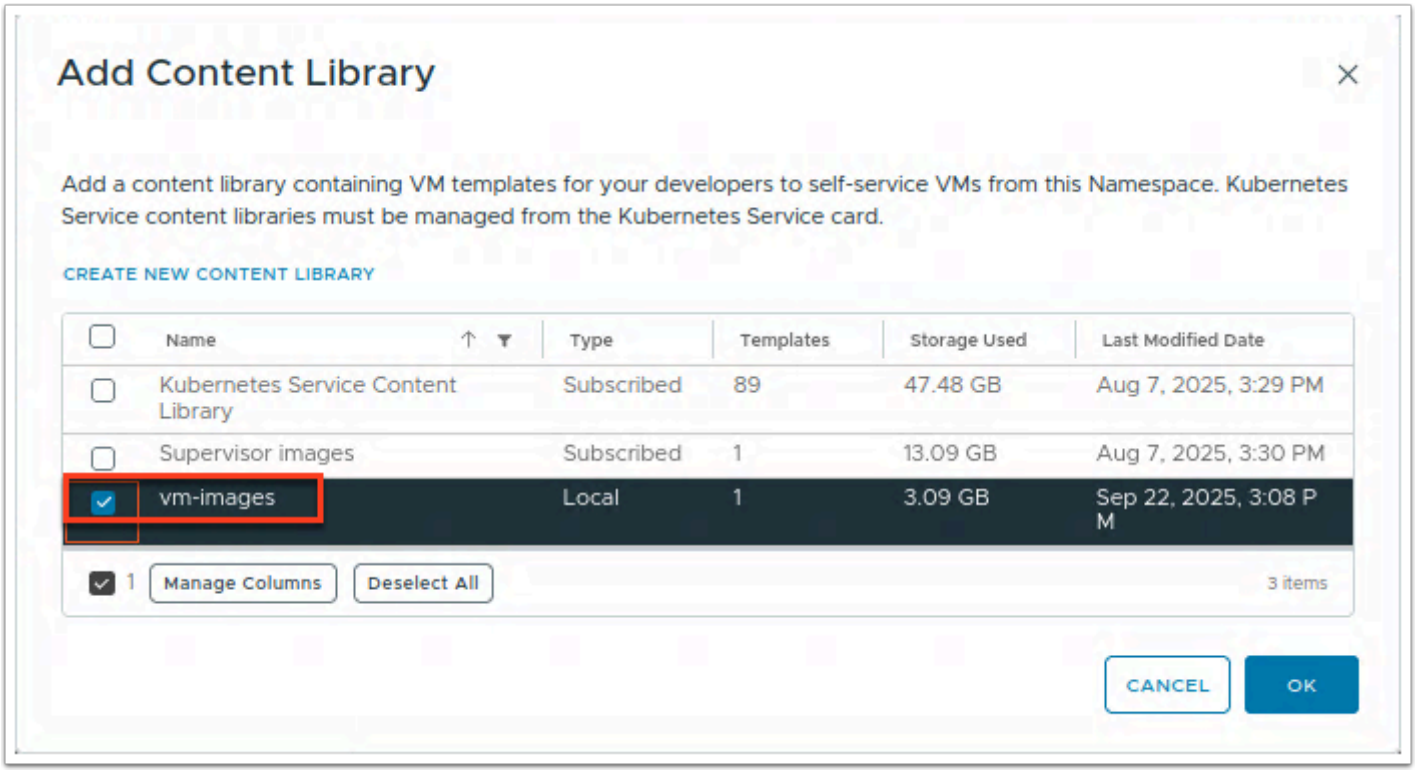
1. Click on **namespace-1**.

Manage the Content Library



1. Click **ADD CONTENT LIBRARY** on the **VM Service** tile.

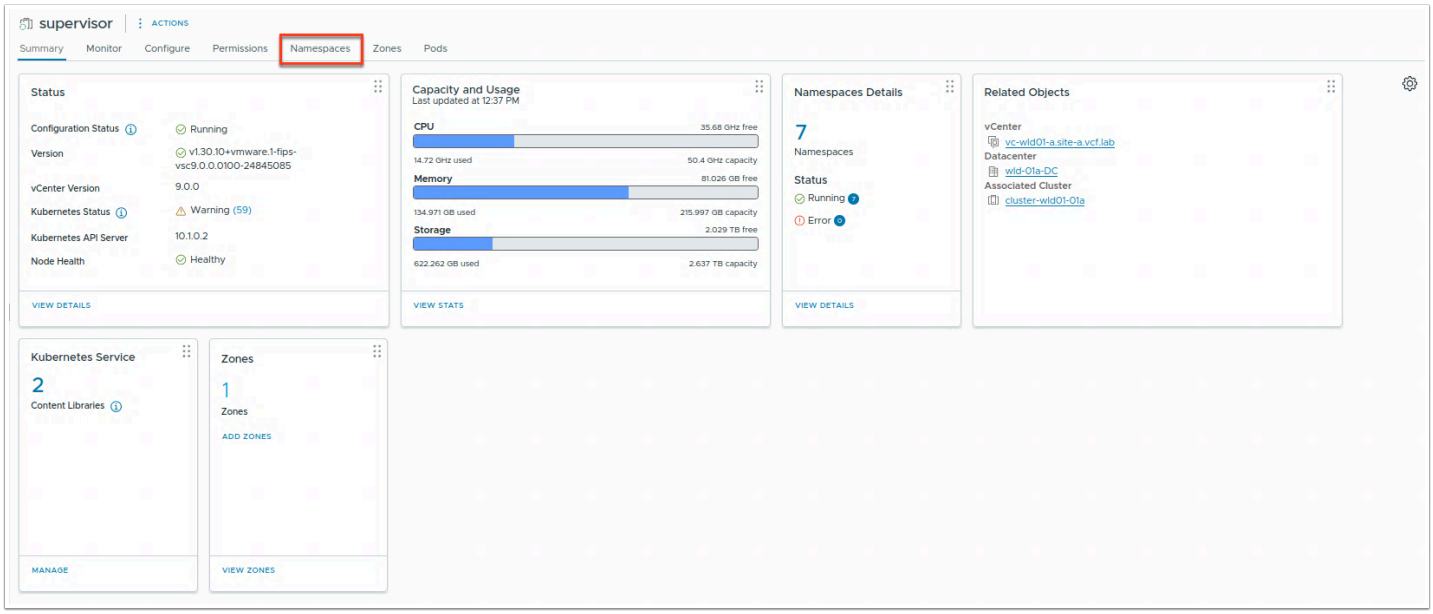
Select the Content Library



1. Click the Radio Checkbox for **vm-images**.
2. Click **OK**.

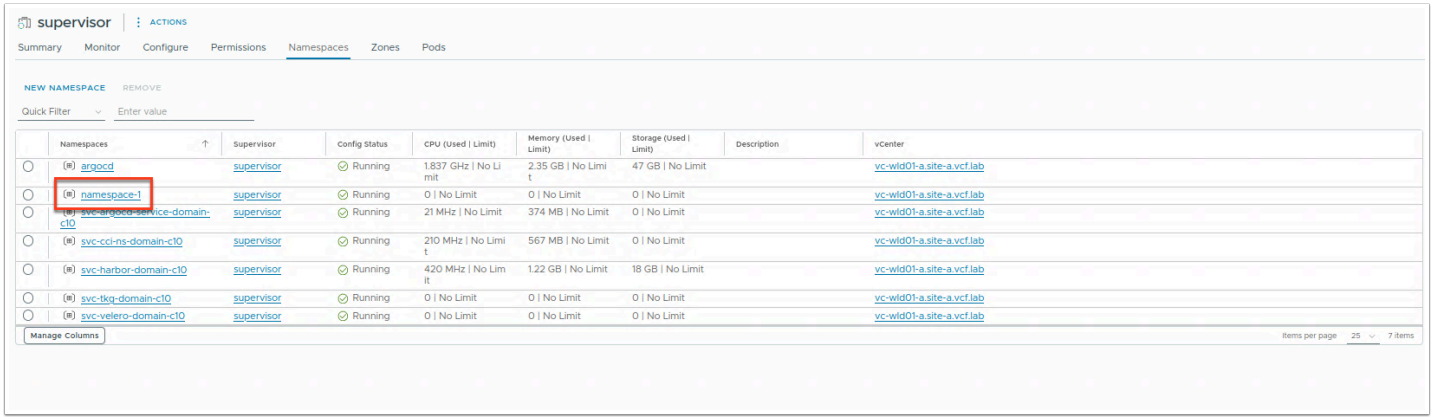
It may take a few seconds but you will see the ***vm-images*** content library added to the namespace.

Return to Namespaces



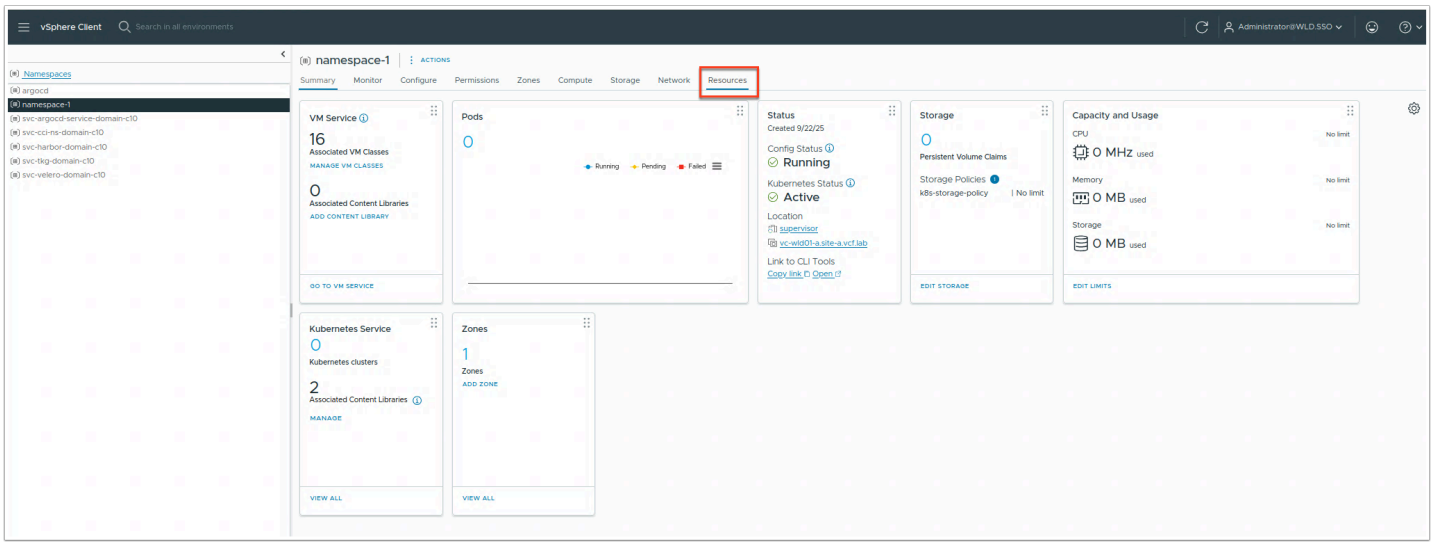
1. Click on the **Namespaces** Tab.

Select namespace-1



1. Click on **namespace-1**

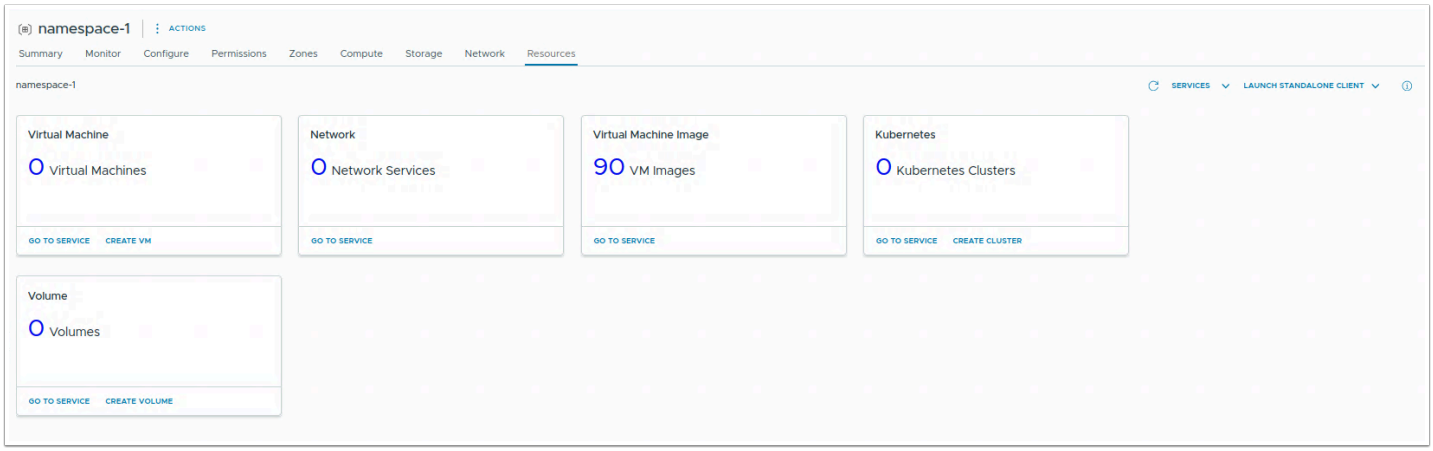
Create a VM



Next, let's deploy a VM via the VM Service through the Consumption Interface.

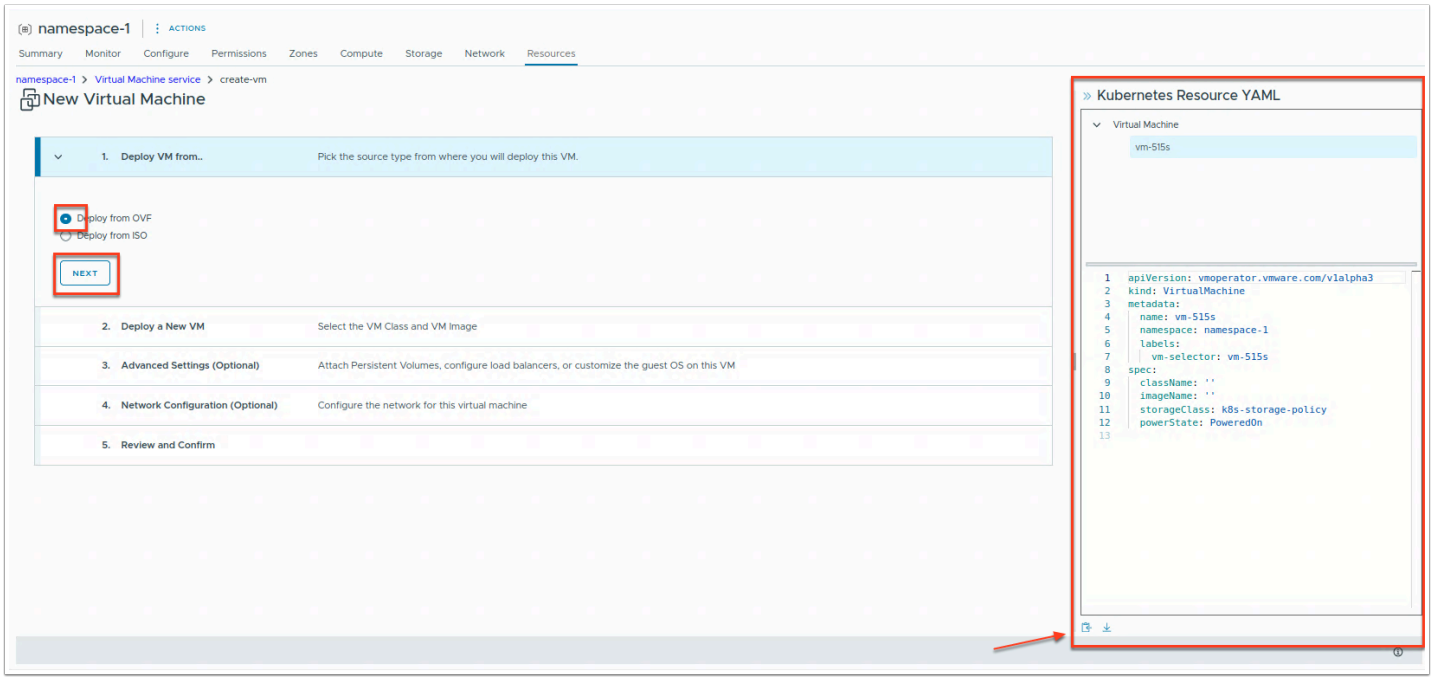
1. Click the **Resources** tab.

Select The Create Link



1. In the **Virtual Machine** tile, click **CREATE VM**

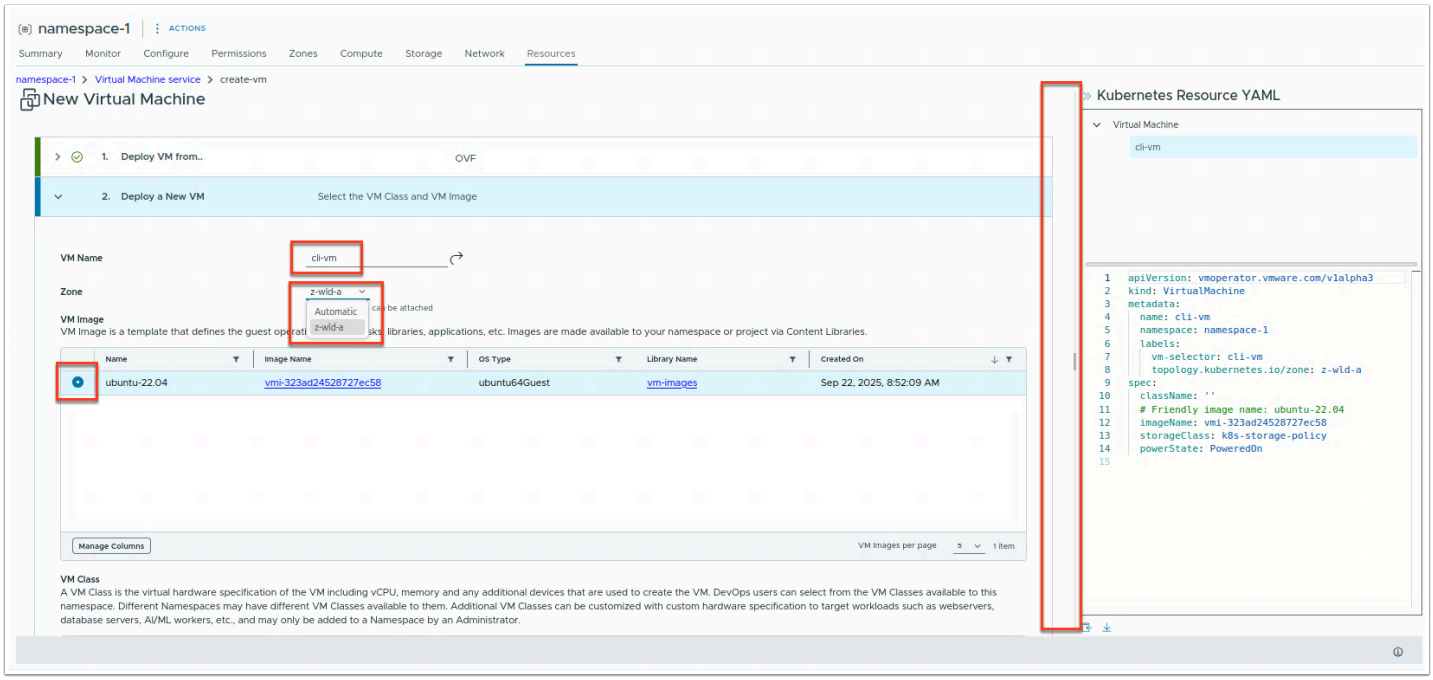
Select the OVF



1. Leave **Deploy from OVF** checked.
2. Click **NEXT.**

> Keep an eye on the Kubernetes Resource YAML part of the screen. Take note of the Copy to **Clipboard** and **Download** buttons

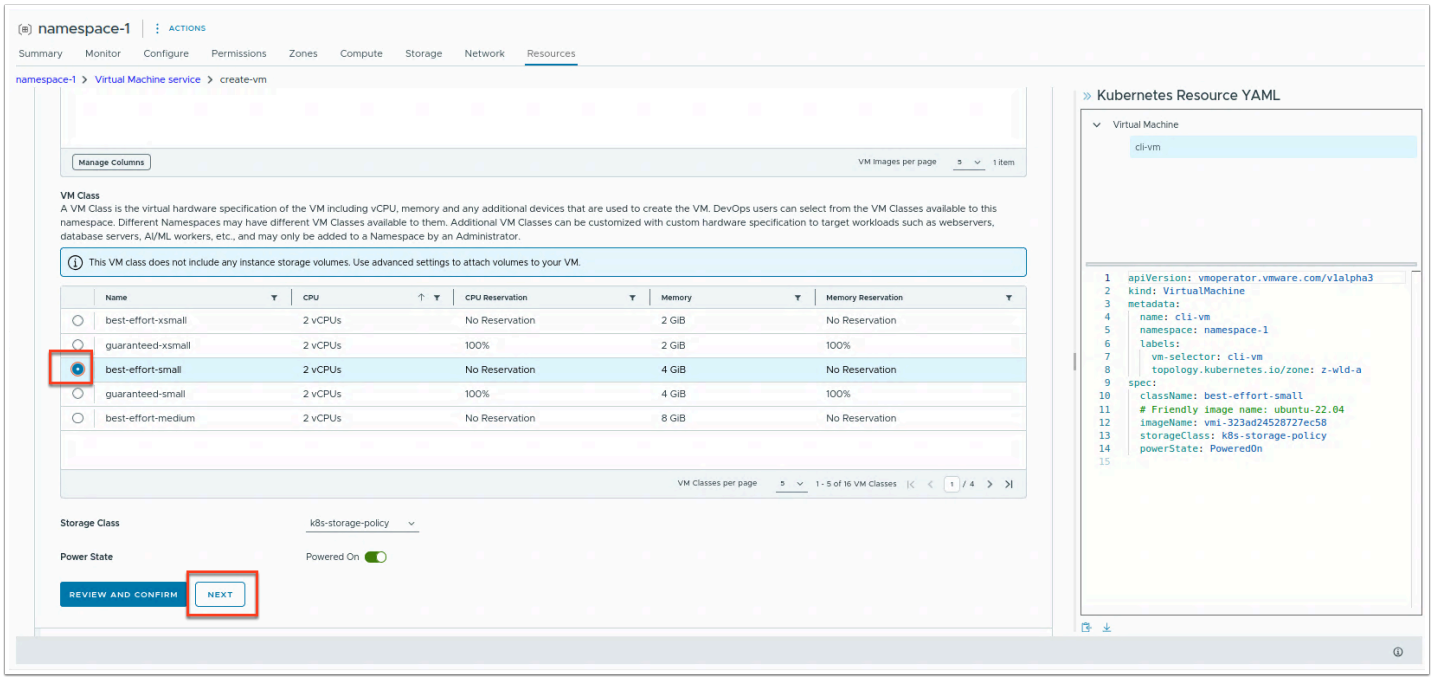
Configure the VM



1. For **VM NAME:** ***cli-vm***.
2. For **Zone**: ***z-wld-a***.
3. For **VM Image Name**: ***ubuntu-22.04***.

> This center window scrolls. You cannot see it when you are not scrolling up or down, but it is there.
>
> Scroll down

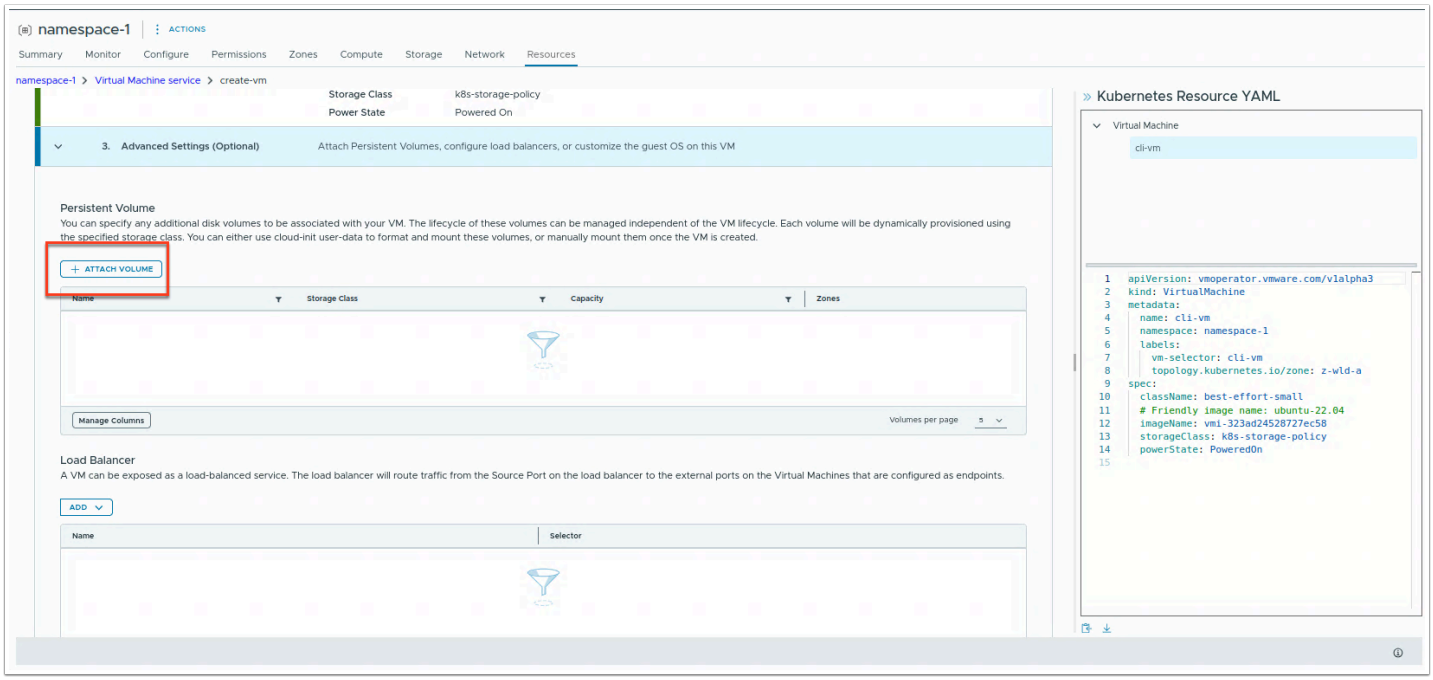
Select the VM Class



Now we will select the VM class, the VM class specifies the configuration for the VM that will be created. In this case we are requesting a best-effort-small, this will tell the service to TRY and create the VM with this specification. If a VM class is not available it will select the next available VM class. You may have to scroll down to see the VM classes.

1. Click the radio button for ***best-effort-small***.
2. Click **NEXT**

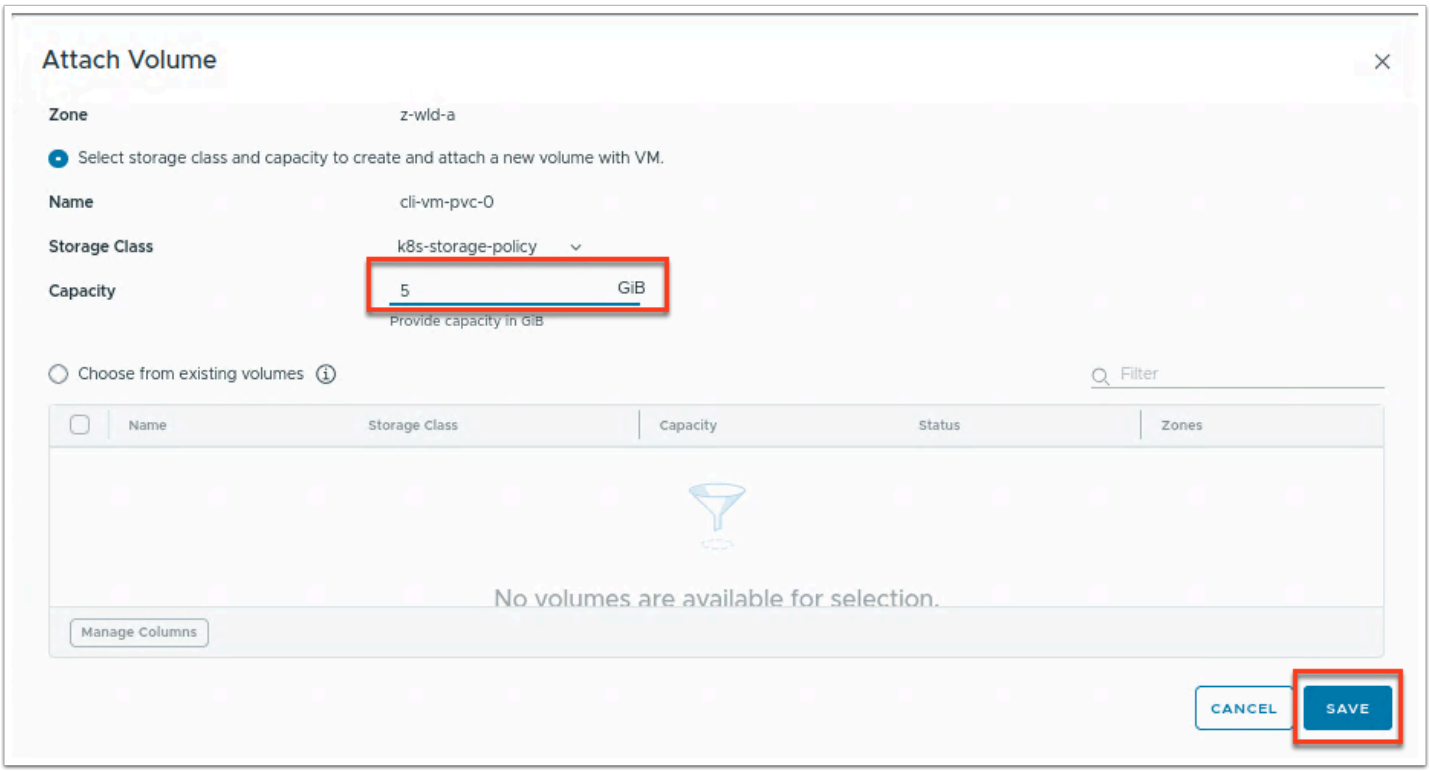
Select the Persistent Volume



Here we will assign a Persistent Volume to the VM we are creating. A Persistent Volume is storage assigned to the VM which will remain even if the VM is deleted.

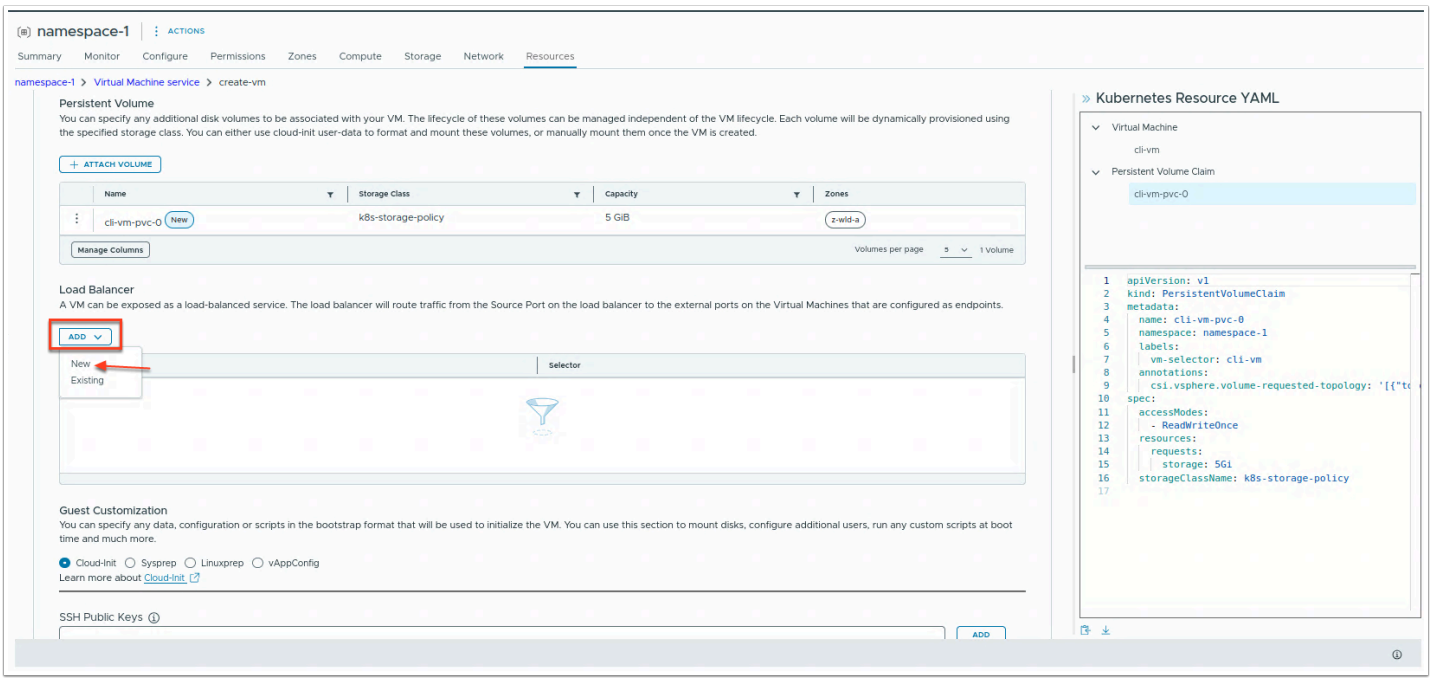
1. Click **ATTACH VOLUME**.

Attach a Storage Volume



1. Enter **5GB** for **Capacity**.
2. Click **SAVE**.

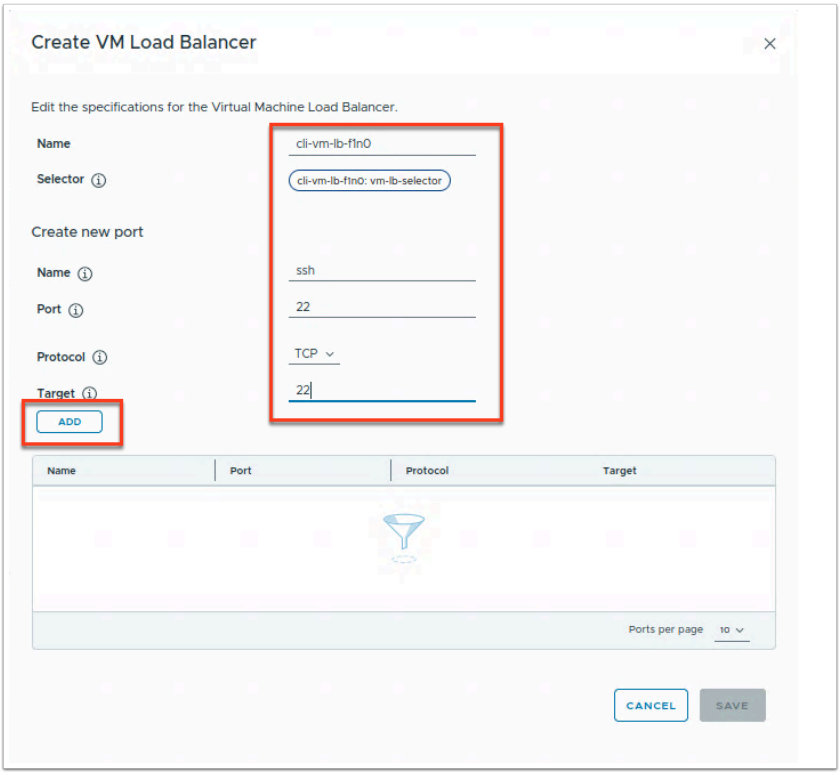
Attach a Load Balancer



Here we are going to attach a Load Balancer to our VM. A load balancer is a device or software that distributes network traffic across multiple servers, ensuring no single server is overwhelmed and maximizing resource utilization. It enhances application performance, reliability, and availability by intelligently routing client requests to healthy servers.

1. Scroll down to the Load Balancer section and click **ADD**.
2. Select **NEW**

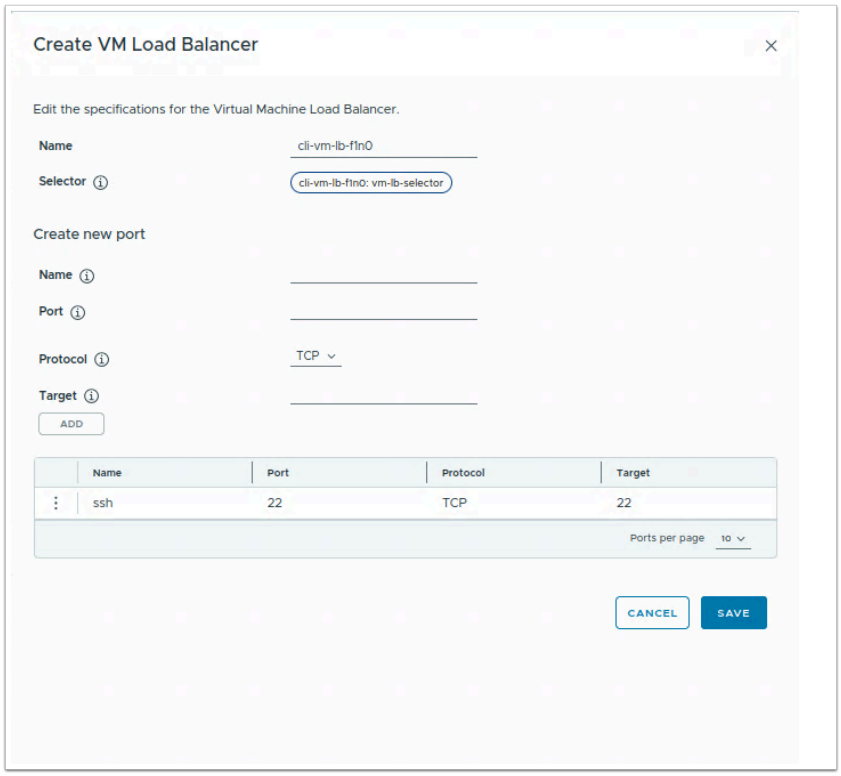
Create a VM Load Balancer - Step 1



Use the Following Settings to Create a new Loadbalancer for this VM

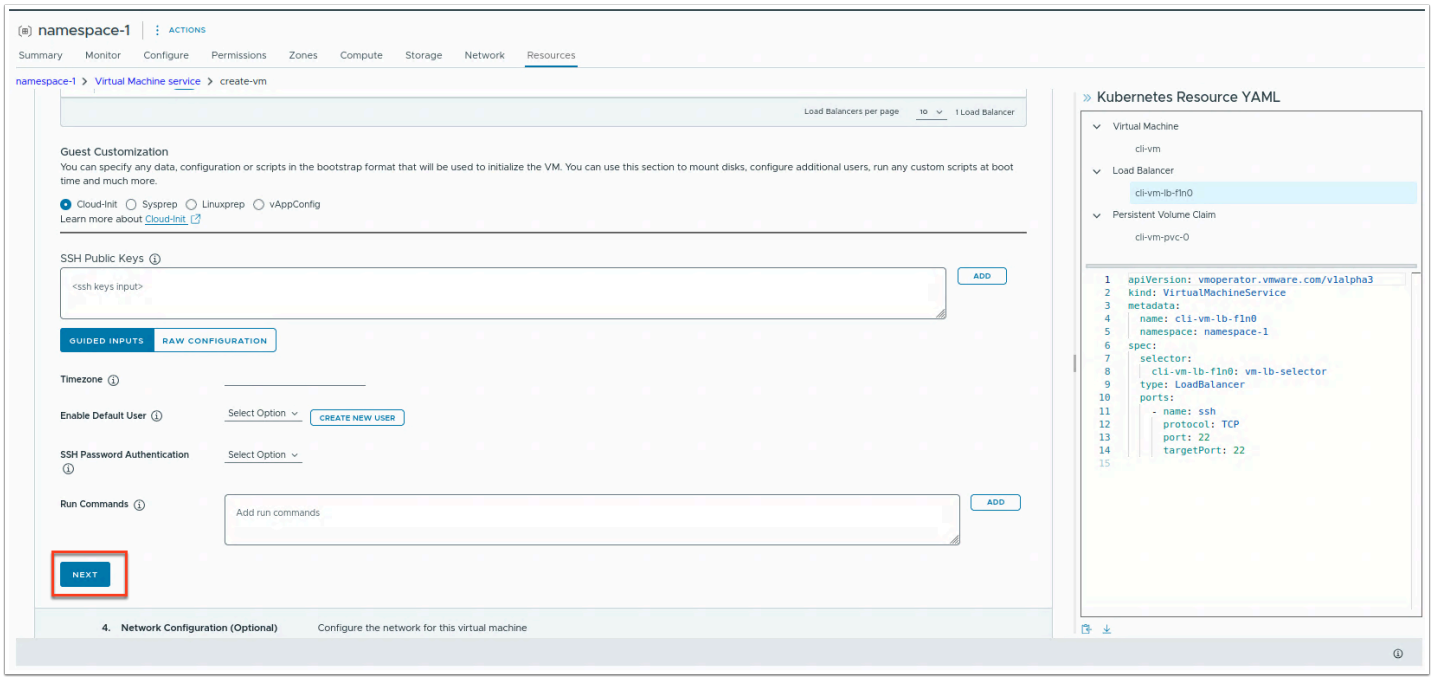
- **Name:** Add ***cli-*** to the name and leave the unique identifier in tact
- **Create new Port:**
  - **Name**: ***ssh***
  - **Port**: ***22***
  - **Protocol**: select ***TCP***
  - **Target**: ***22***

1. Click **ADD**.



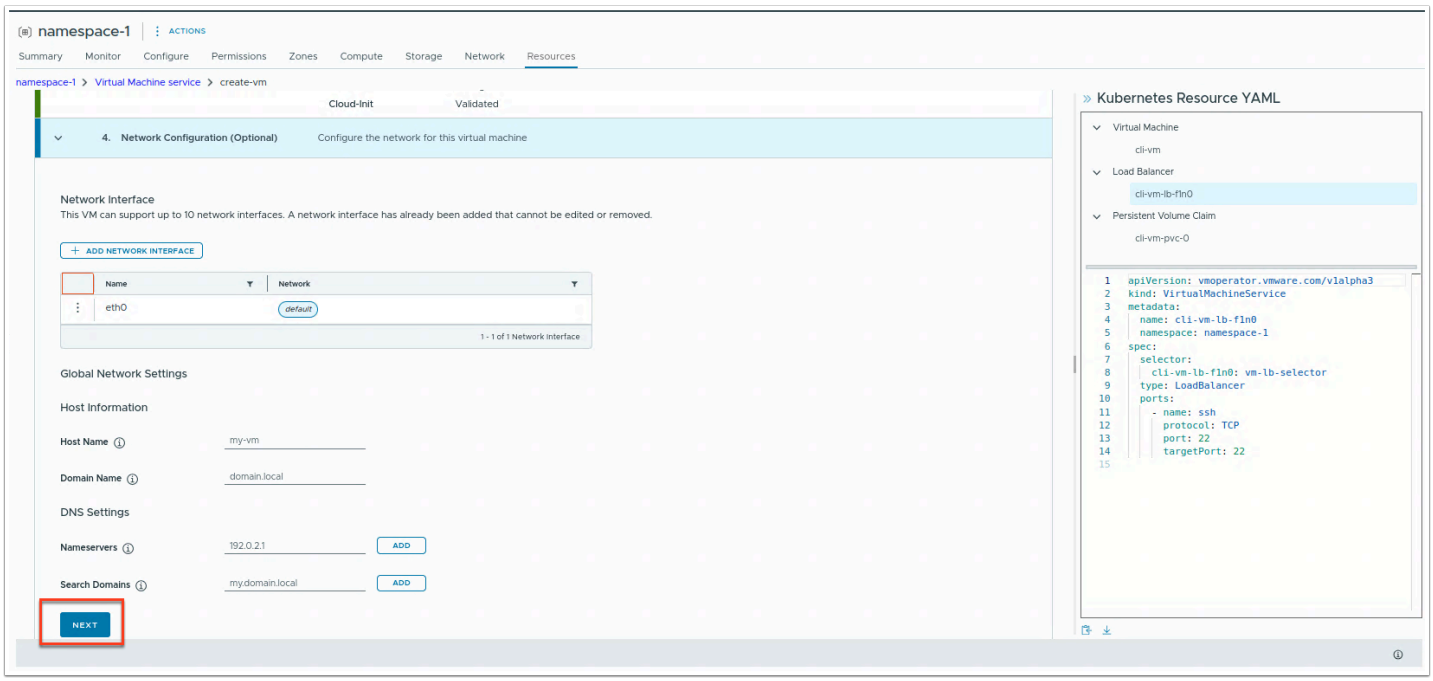
1. Select **SAVE**.

Guest Customization



1. Scroll down and select **NEXT**.

Network Configuration



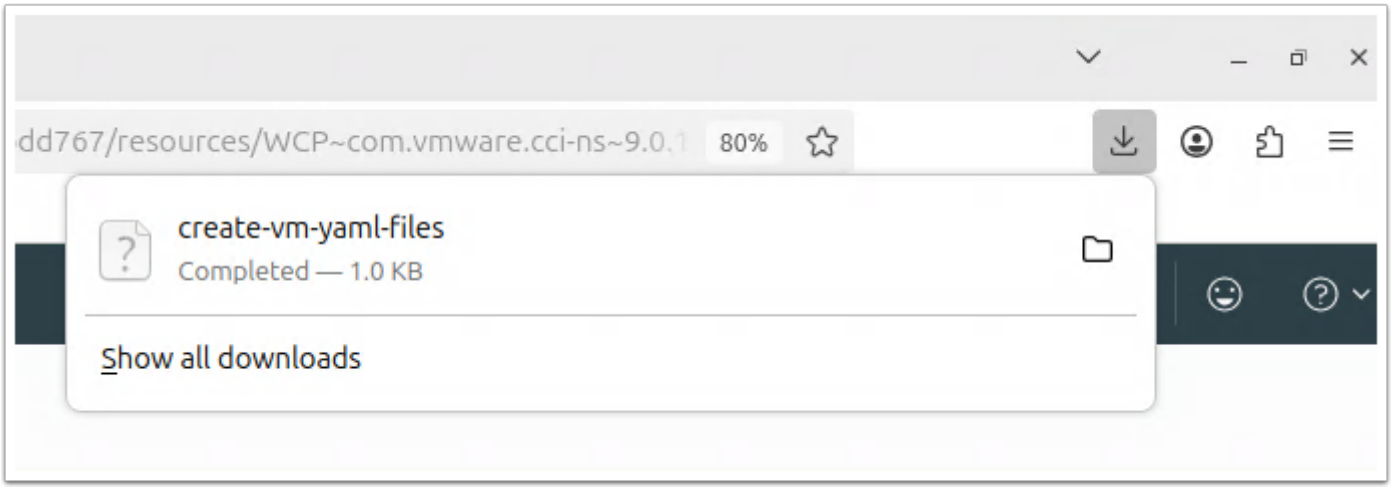
Accept the default values.

1. Scroll down and select **NEXT**.

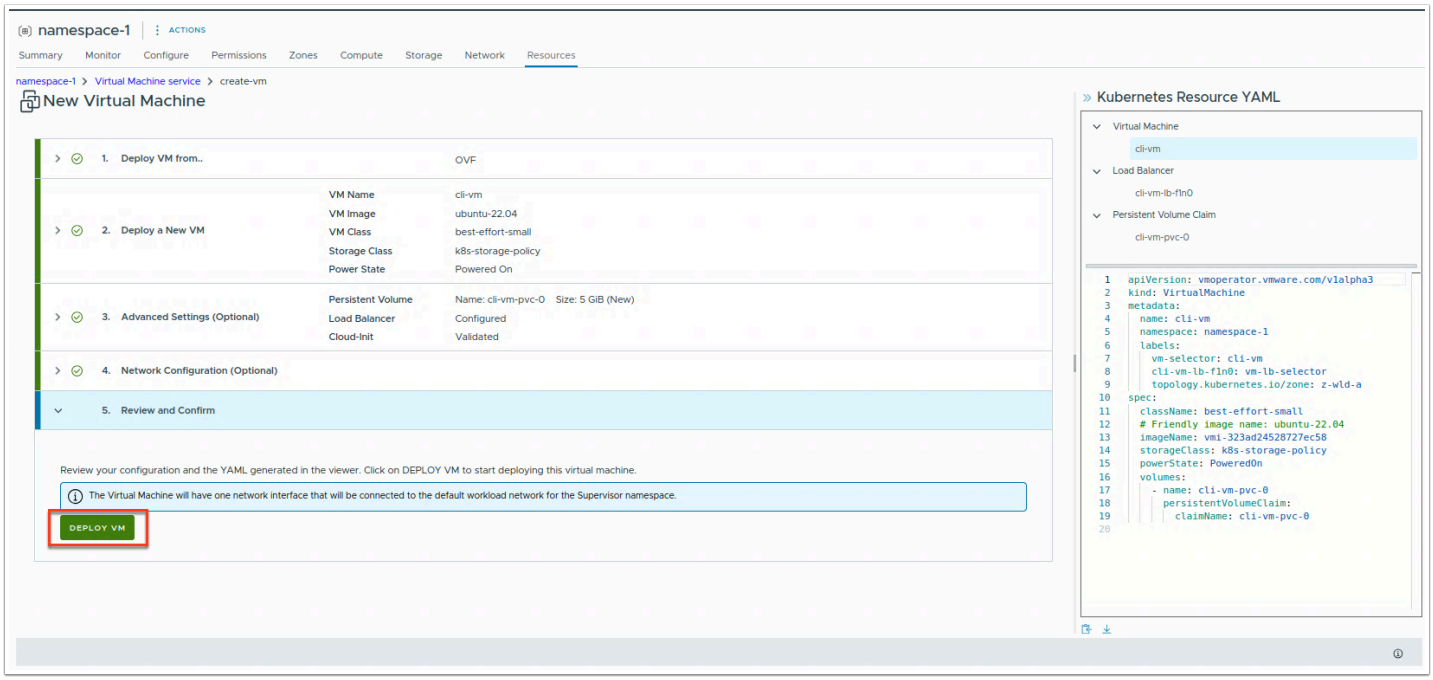
Review the YAML

Here you can review the YAML file that will be used to deploy the VM. You can copy or download the YAML file to use as a template for additional deployments.

Click the download icon in the RED box to download the YAML file.



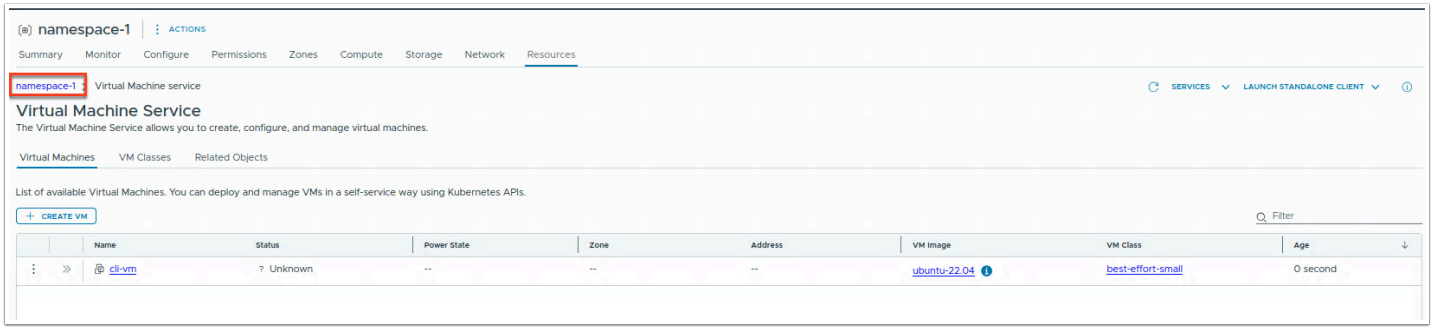
Create a VM - Review and Complete.



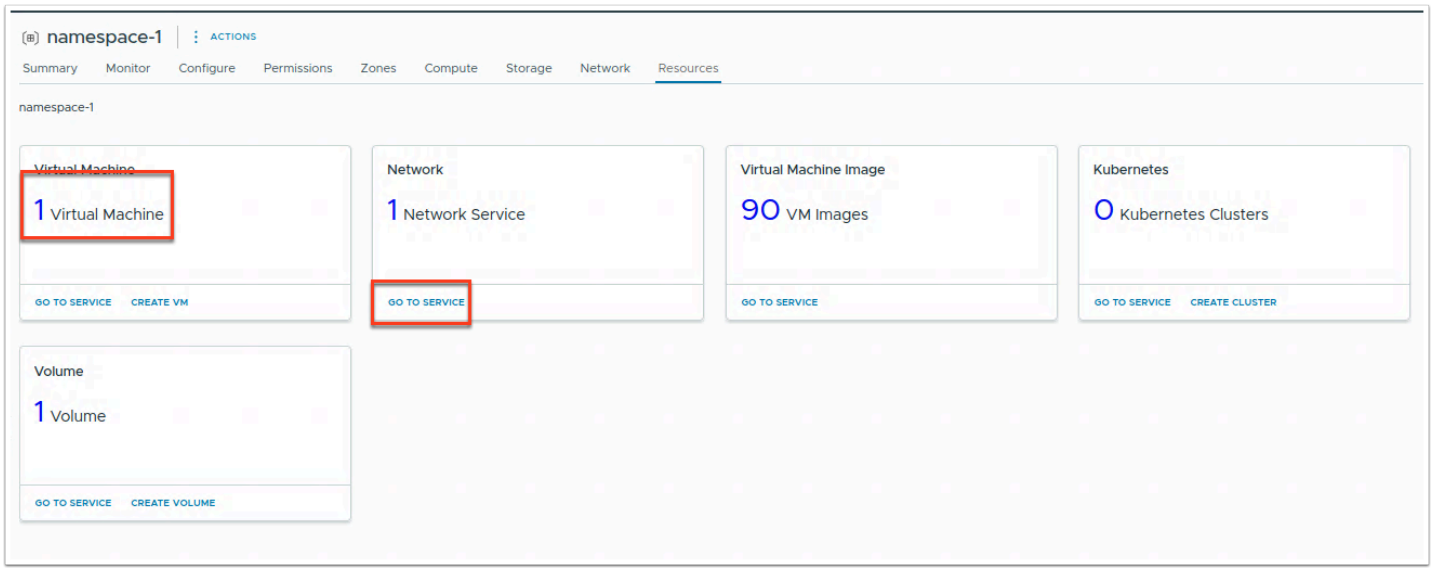
Review the settings.

1. Select **DEPLOY VM** to deploy the VM .

Review The VM Load Balancer Deployment

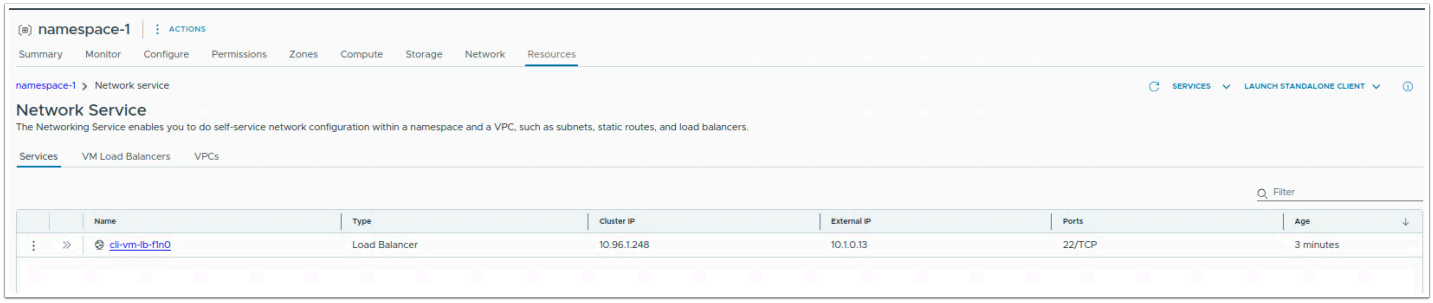


Click **namespace-1** in the breadcrumbs menu to go back to the root of the **Resources** tab



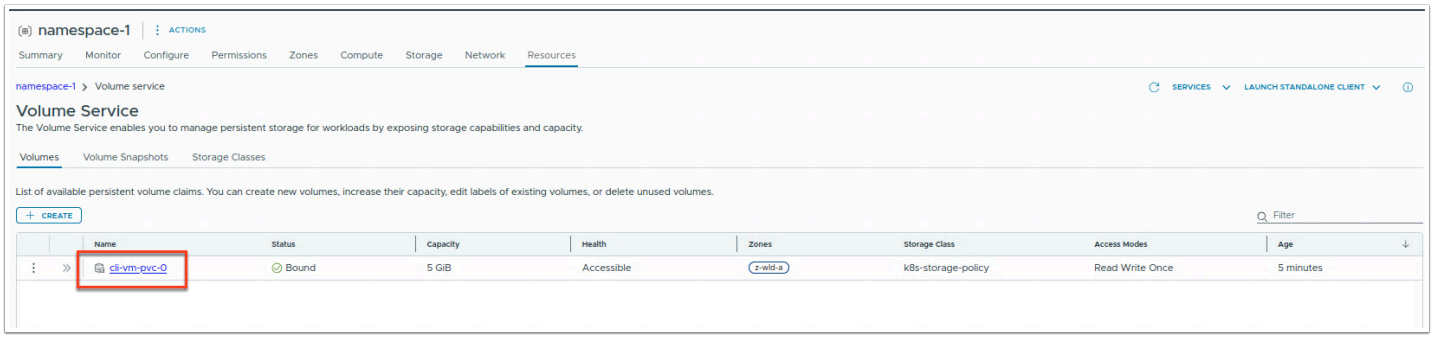
You should now see 1 **Virtual Machine**

Click **GO TO SERVICE** in the **Network** tile



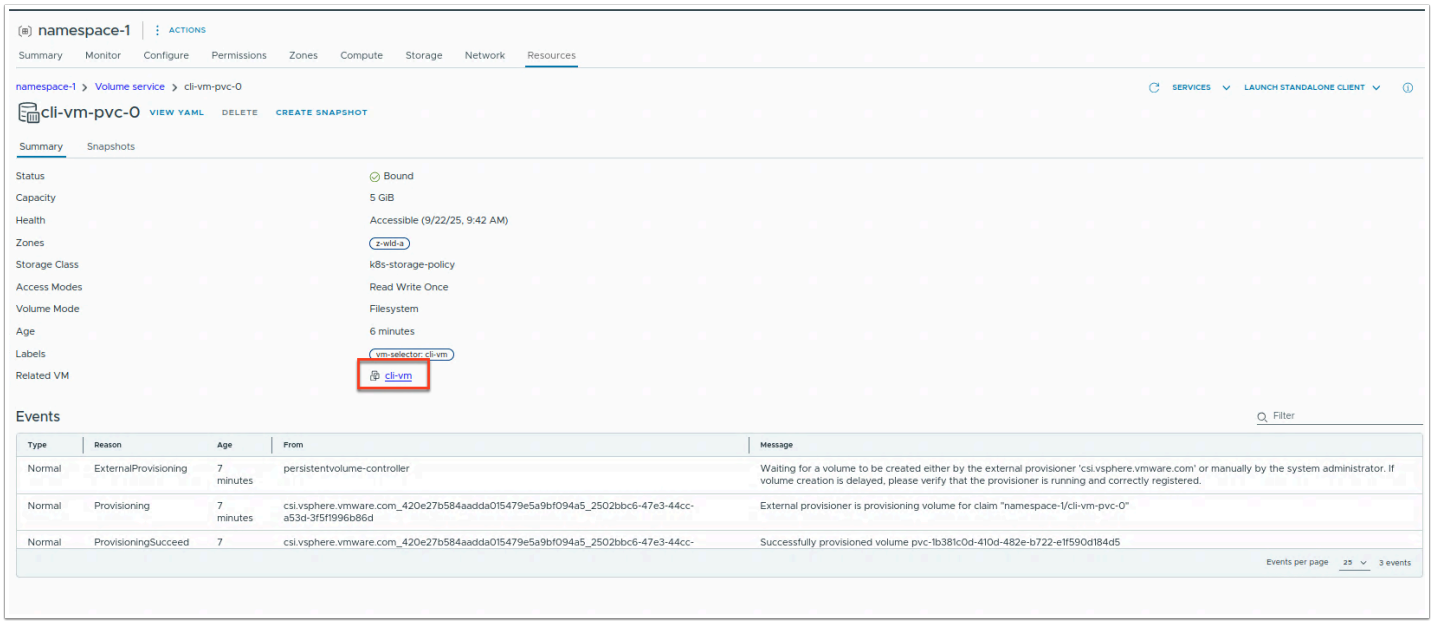
You will see the LoadBalancer is created

Click **namespace-1** in the breadcrumbs menu to go back to the root of the **Resources** tab. In the **Volume** tile click on **GO TO SERVICE.**



You will see the new **Persistant Volume** has been created. It also shows that it's current status is **Bound.** Meaning this PVC is currently in use.

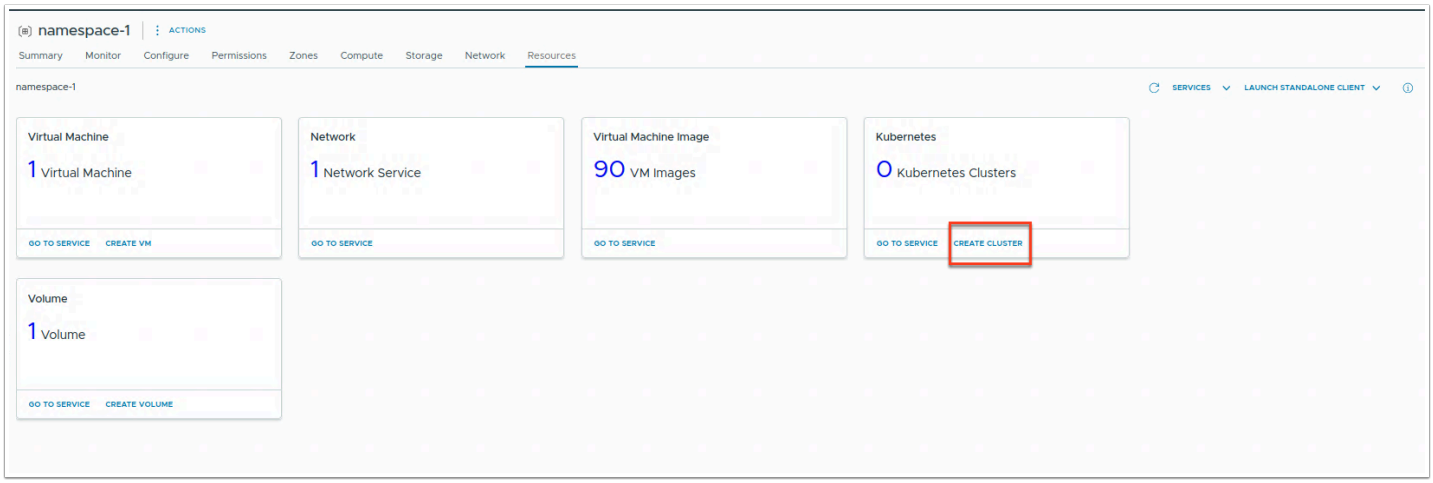
Click on **cli-vm-pvc-0**



Here you can see information on the PVC, including what VM it is bound to.

Deploy a Kubernetes Cluster (VKS)

Switching from a Traditional **VM**, in the following section you will deploy a **Kubernetes** Cluster using the **vSphere Supervisor Service - VKS**

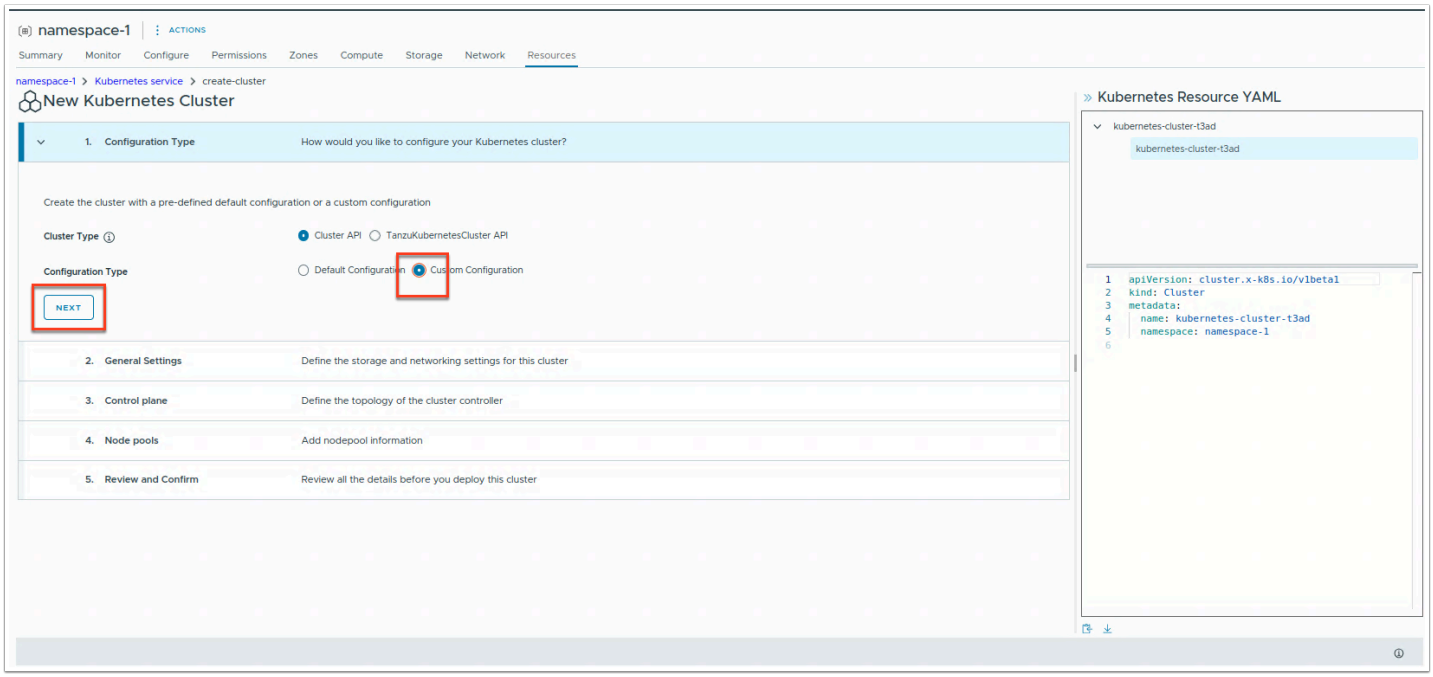


Filling out the cluster configuration has 5 steps. Be sure to scroll to the bottom of each step even if you don't fill anything out. Pay attention to each option.

Return to **namespace-1** if you are not already there and click on the R**esources** tab

1. Click **CREATE CLUSTER** option within the **Resources** tab of the **Kubernetes** tile

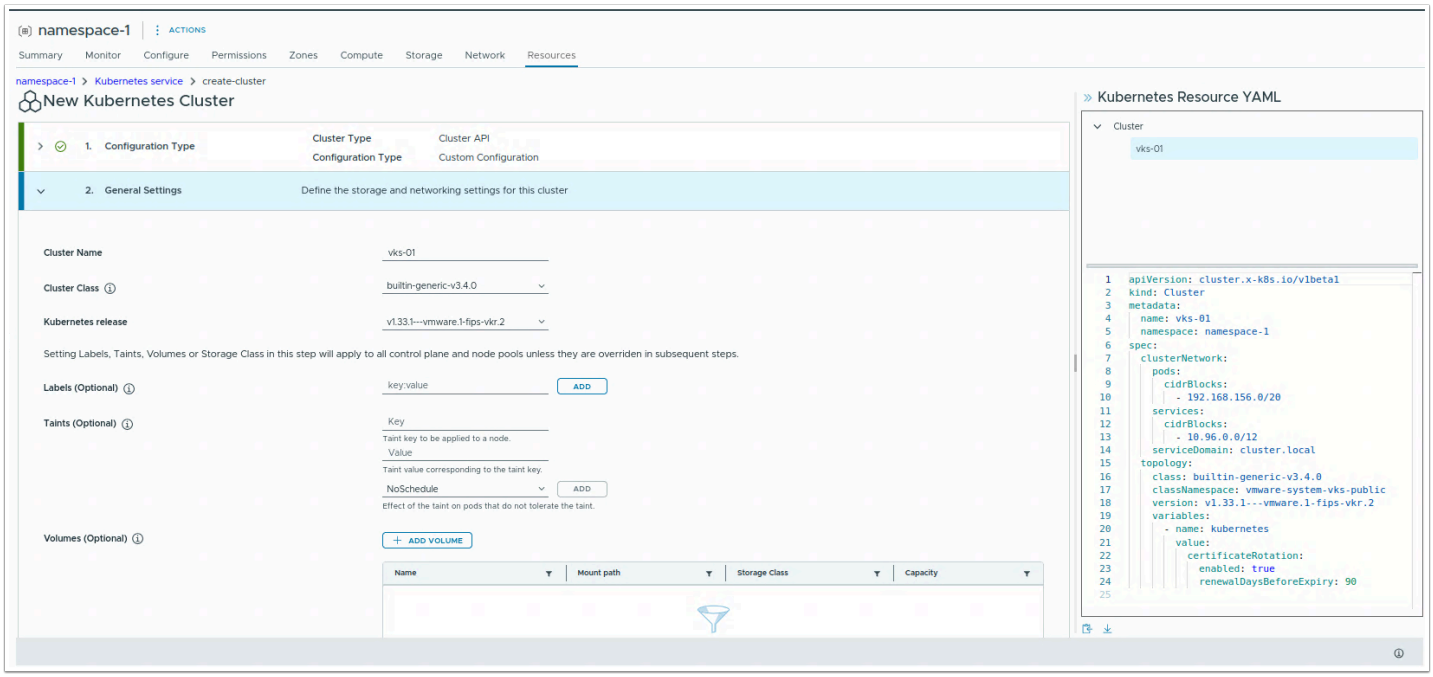
Create a New Kubernetes Cluster



1. Verify that the **Cluster API** is selected
2. Select **Custom Configuration**
3. Select **NEXT**

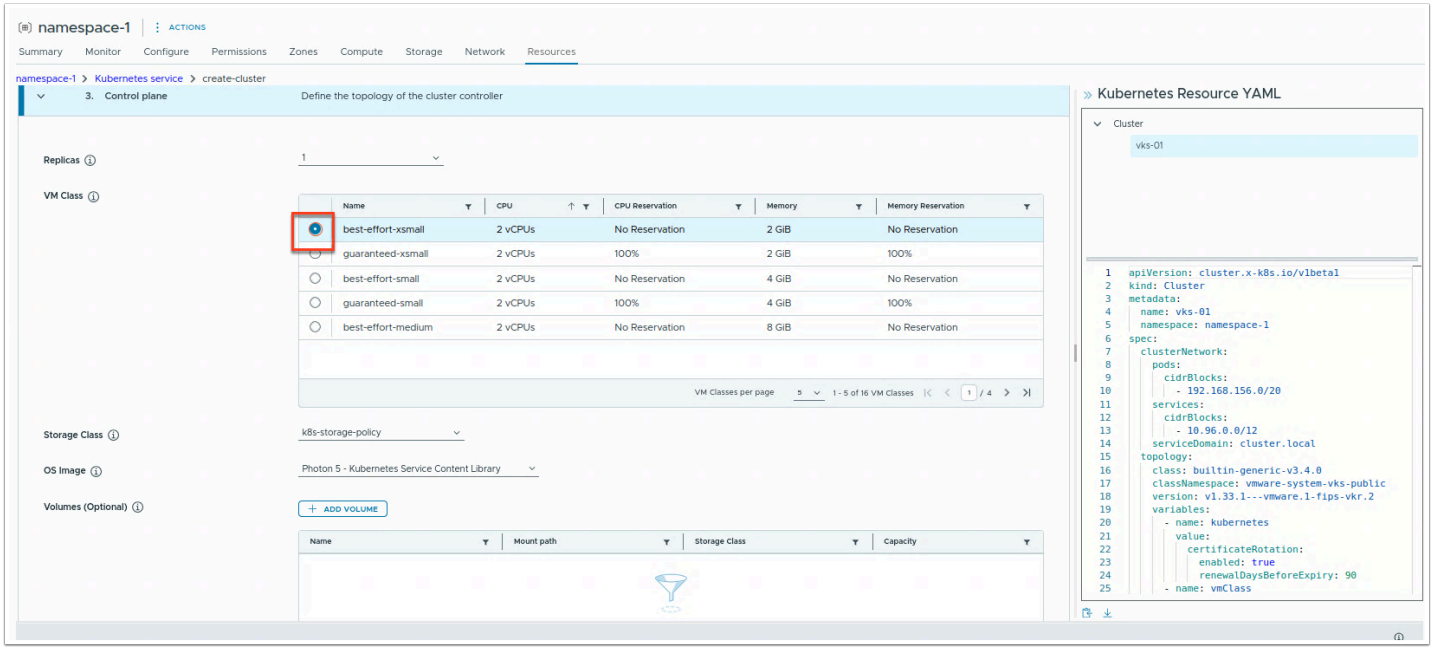
Once Again take note of the YAML as it gets created by your choices

Name the Cluster



1. **Cluster Name**: ***vks-01***
2. Scroll to the bottom of the **General Settings** section
3. Click **NEXT**

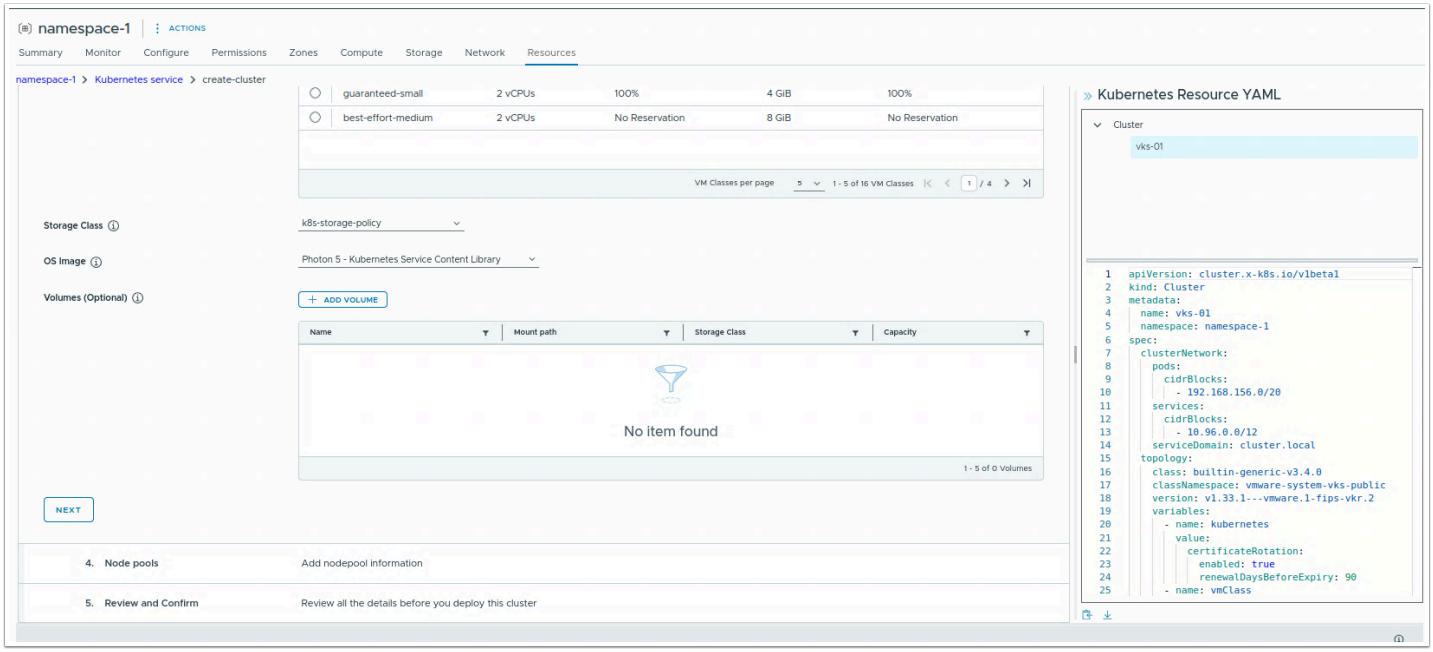
Select Control Plane Node Size



For VM Class

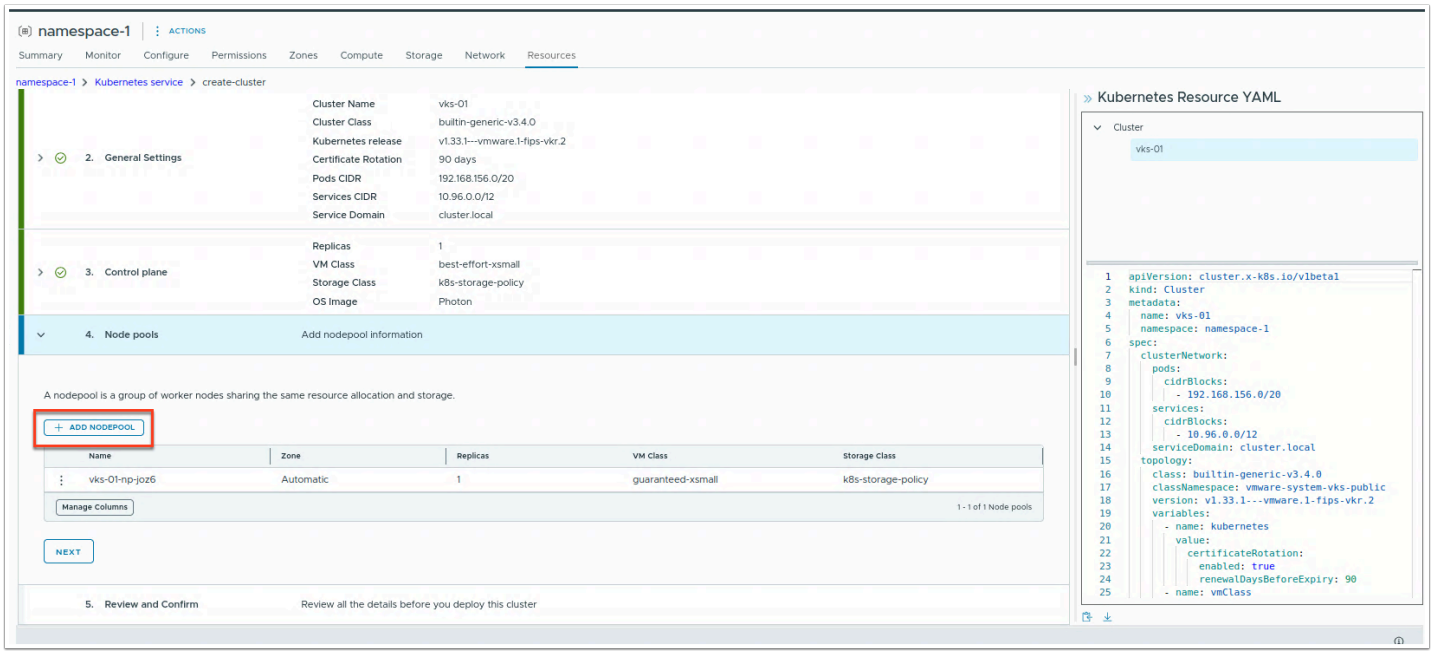
1. Select ***best-effort-xsmall*** for **VM-Class**

2. Scroll to the bottom of this section (Be sure to browse through each option).



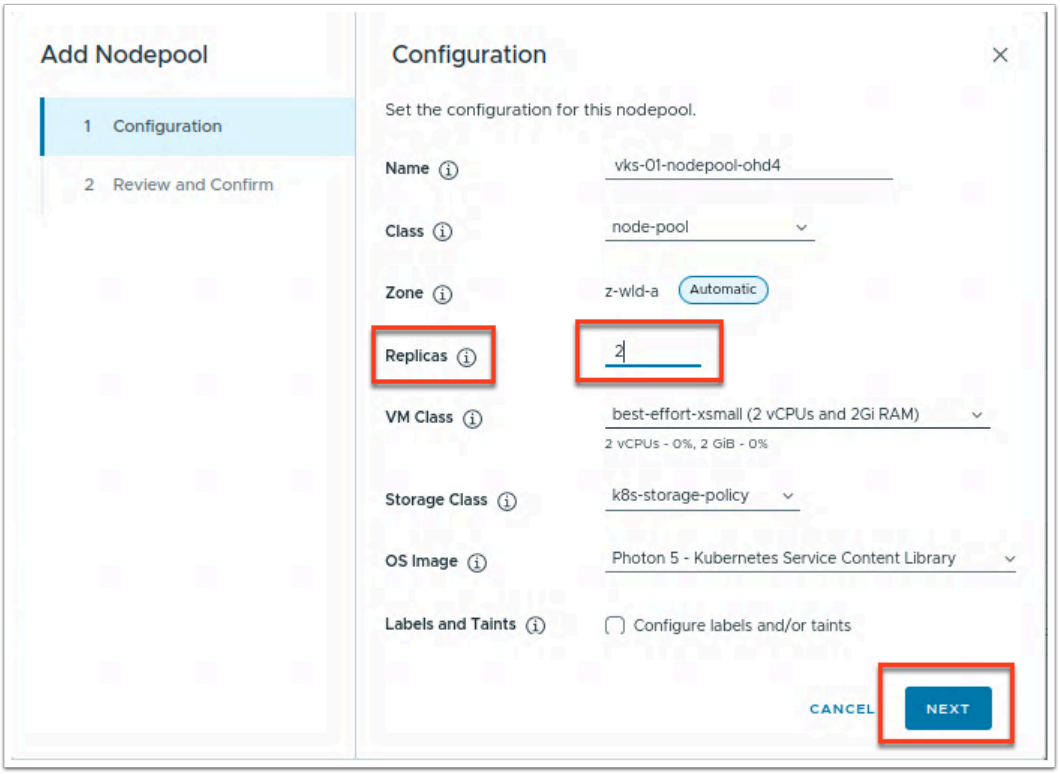
3. Click **NEXT**

Add a Nodepool



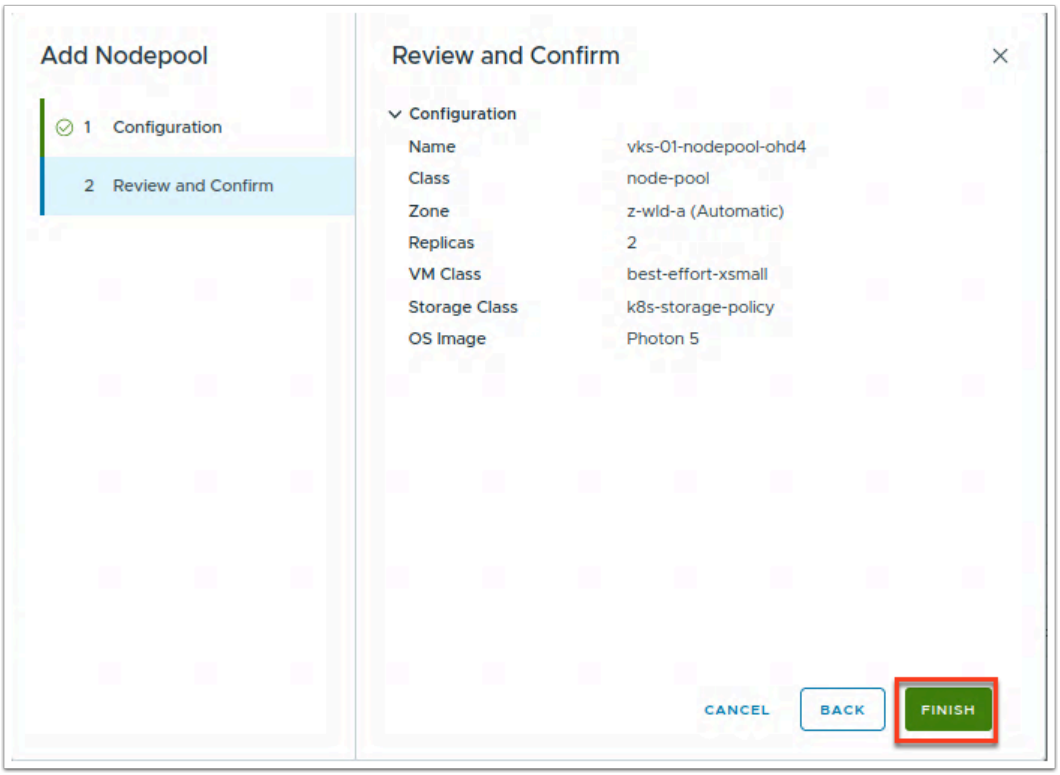
1. Click **+ ADD NODEPOOL**

Add Nodepool Configuration



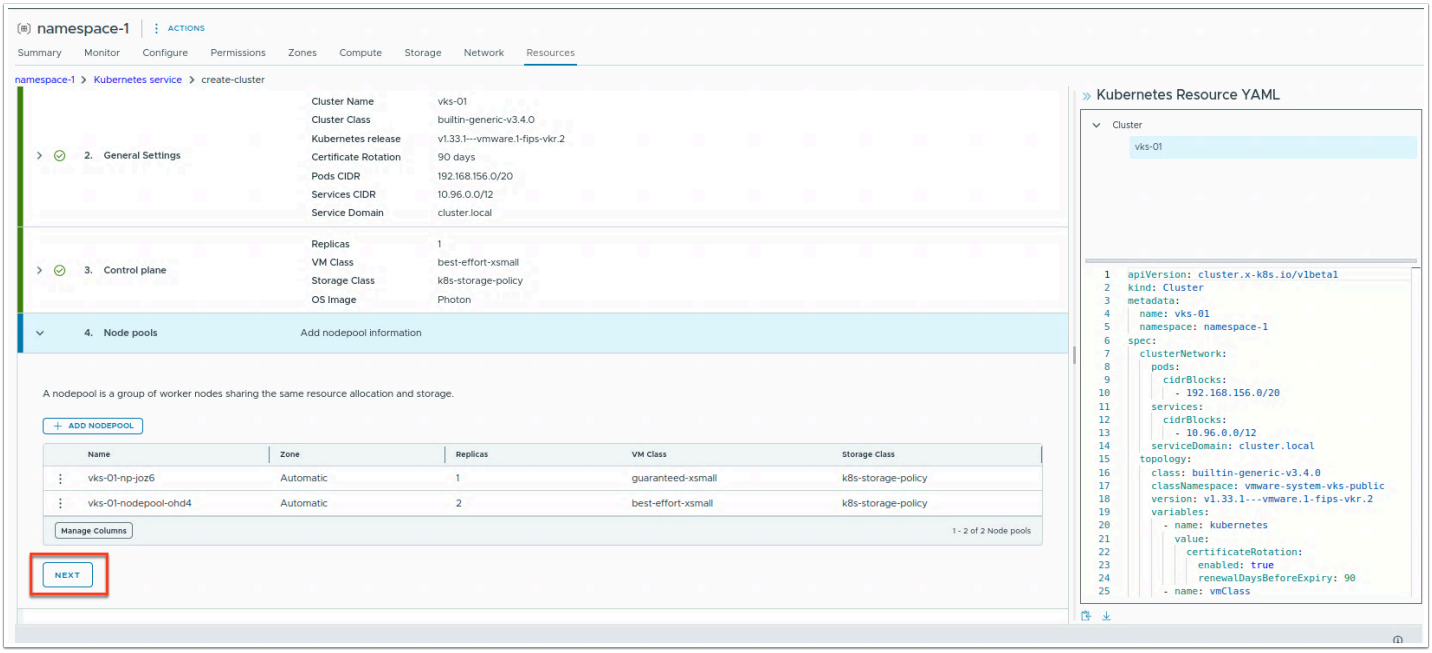
1. Set **Replicas** to2
2. Click **NEXT**

Review and Confirm Nodepool



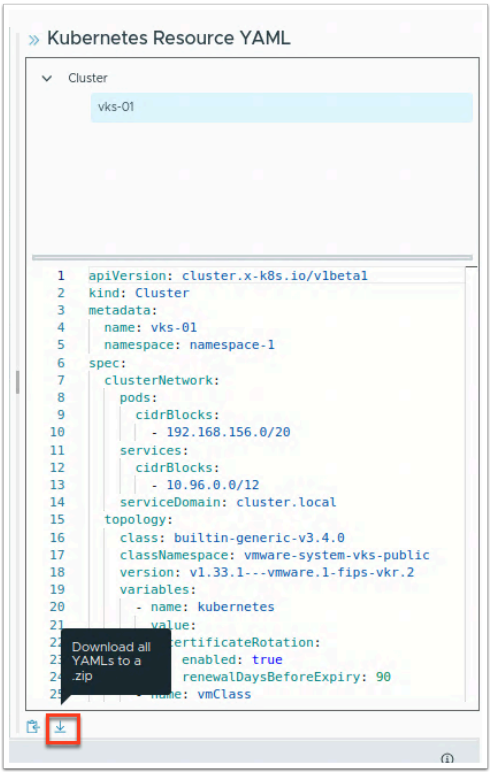
1. Verify everything looks correct and click **FINISH**

Create Cluster Cont'd



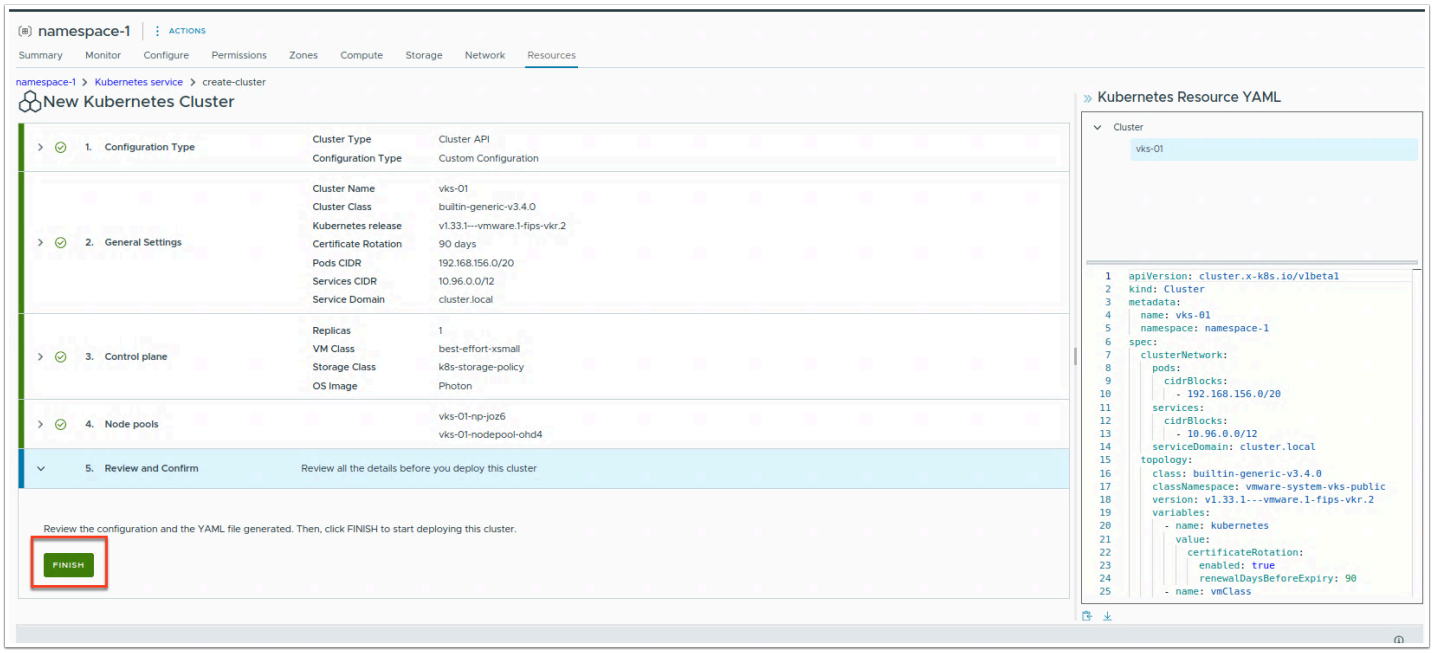
1. Click **NEXT**

Review and Confirm Cluster Create



1. While reviewing the cluster create settings, you can hover over this button to download a YAML file of the entire cluster creation options to use later via CLI/API to deploy more clusters rapidly.

Review and Confirm Cluster Creation Cont'd



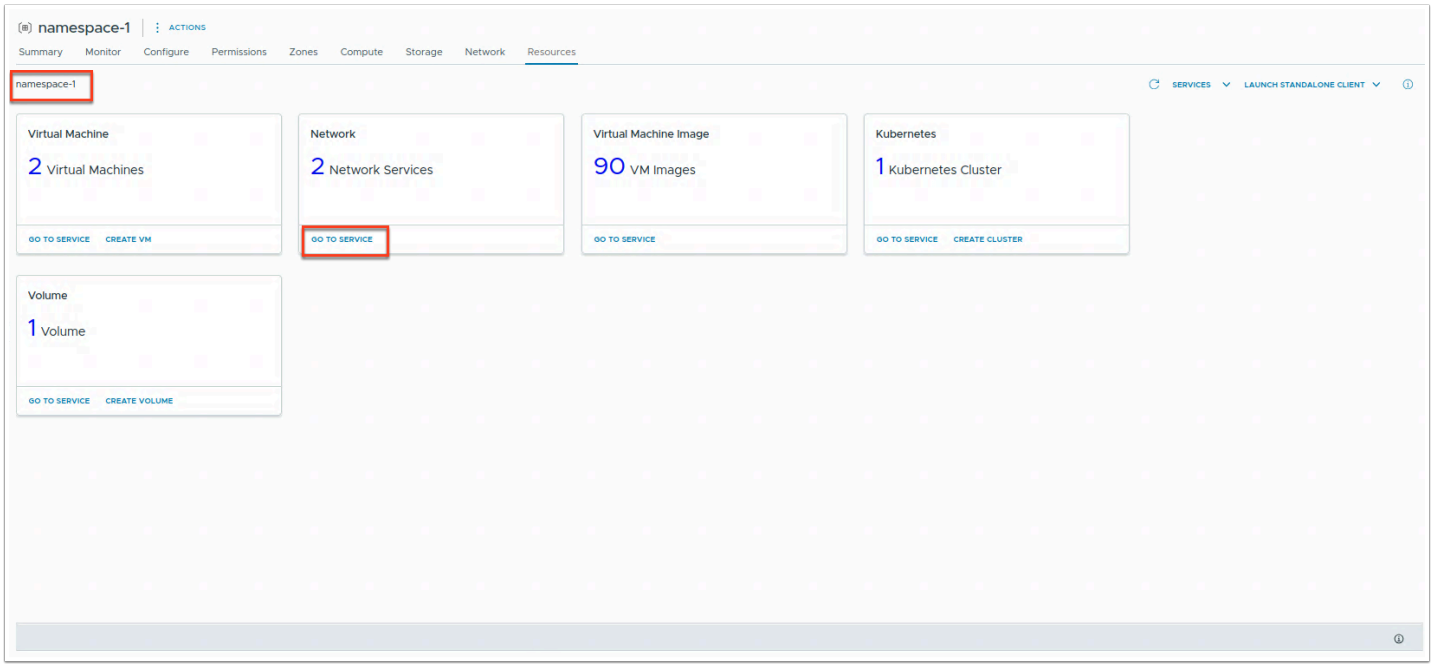
1. Verify everything looks correct and click **FINISH**

Cluster Creation Complete

The VKS cluster will now be deployed. Note:this will take some time to complete.

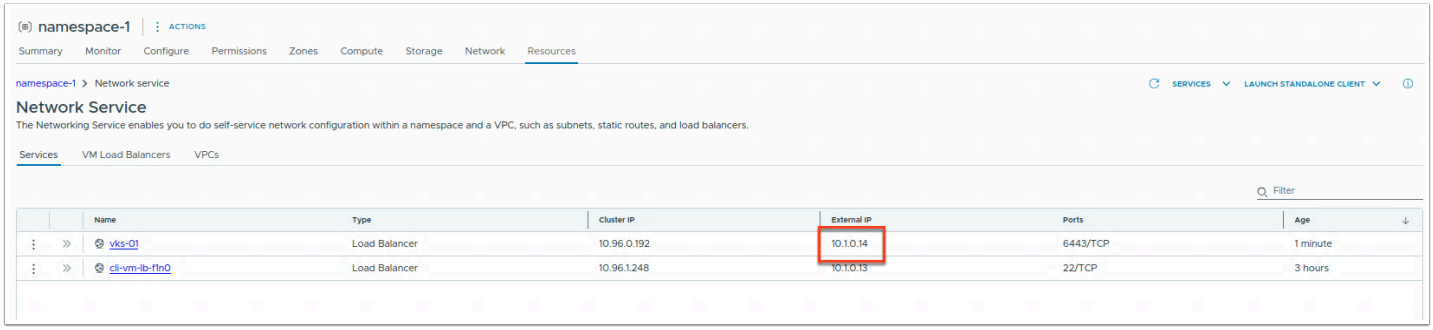
1. Wait for the status to turn toReady
1. This could take up to **5 minutes**

Network Service



1. Return to the main **Resources** tab page for **namespace-1**
2. In the resources tab of **namespace-1**, select **GO TO SERVICE** on the **Network** tile

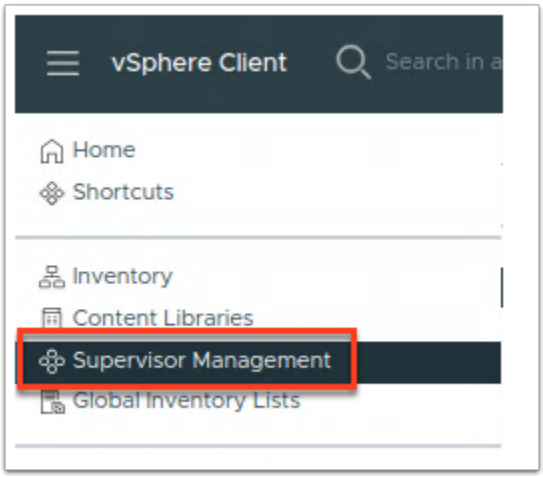
Network Service



Here you can see the **External IP** that was given to your **VKS** cluster upon creation.

Add services in a modular approach

Next, we will show you how quickly you can add different Services to a **Supervisor**. We will show you how to install the Grafana service, but will not actually deploy it in the lab.



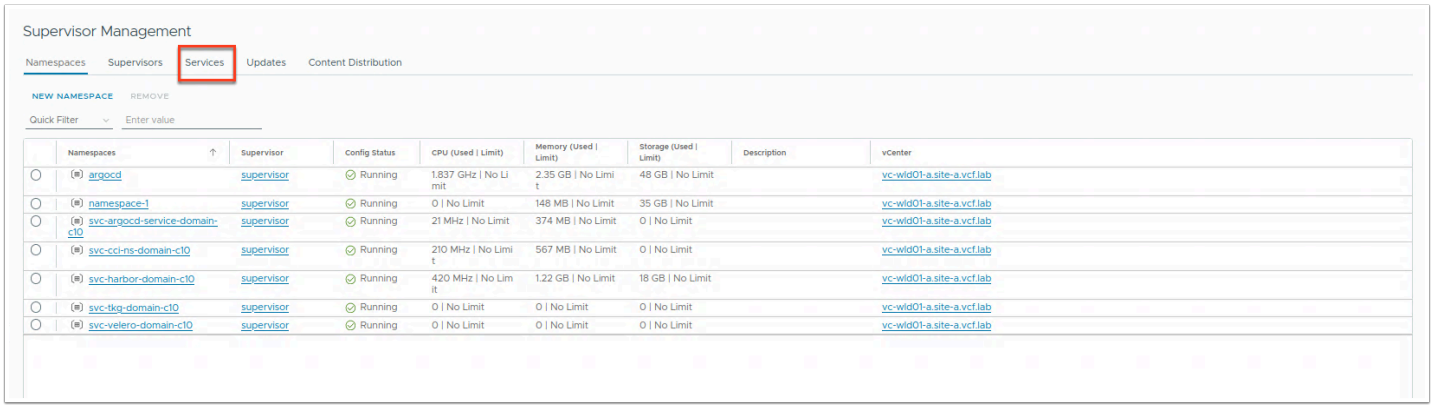
1. On the upper-lefthand dropdown, click **Supervisor Management**

Supervisor Management Services

vSphere Kubernetes Service (VKS) Supervisor Services are components that run within the vSphere Supervisor and are used to provision and manage Kubernetes clusters. These services streamline the deployment and operation of modern applications alongside traditional workloads on vSphere. They offer features like automated lifecycle management, built-in security, and enterprise-grade scalability. Essentially, VKS Supervisor Services simplify the process of running Kubernetes on vSphere, making it more accessible and efficient for IT teams.

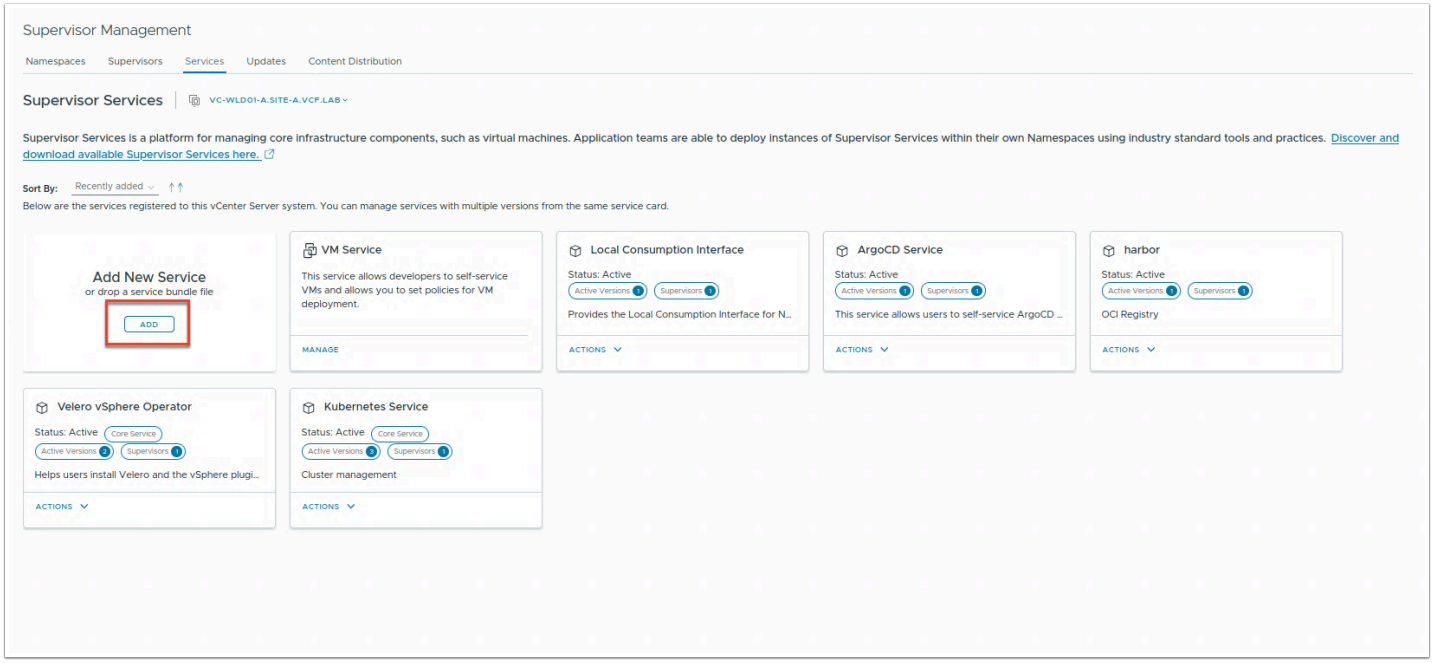
In this lab we have deployed a few supervisor services for you, the Consumption Interface which you have used to deploy resources in a namespace and ArgoCD which we will use later in this lab in Module 5.

Due to the limited internet access within this lab we will not be deploying a supervisor service, the following steps will walk you through how a supervisor service is deployed.



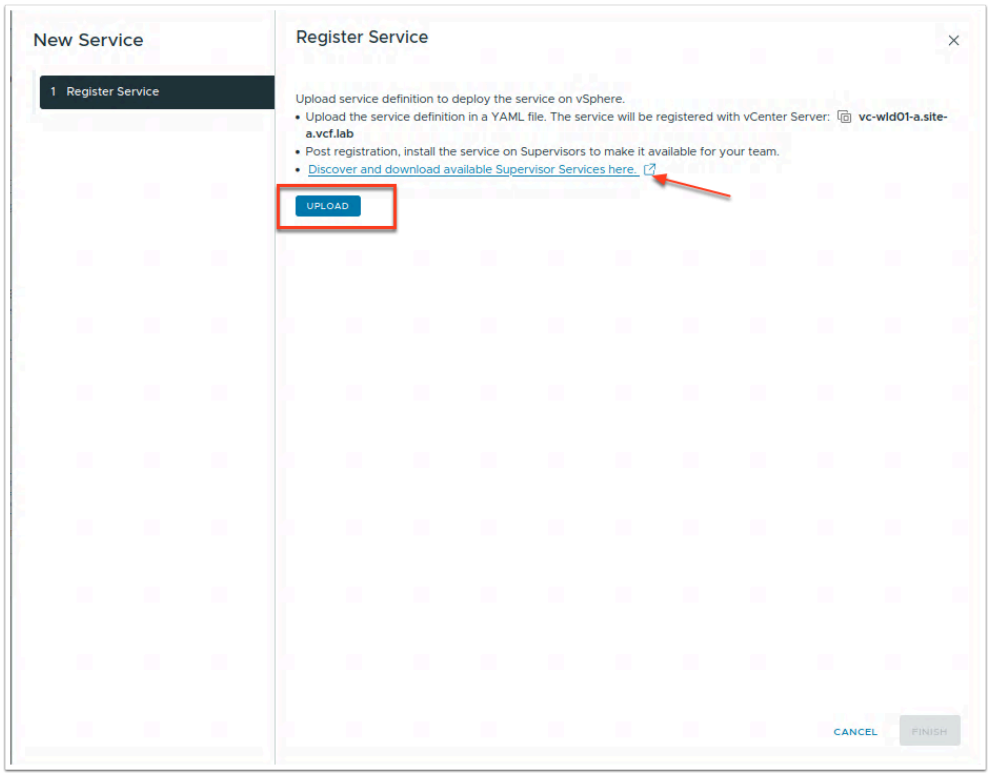
1. Click the **Services** tab

Add Service



1. Click **ADD** in the **Add New Service** tile

Register Service

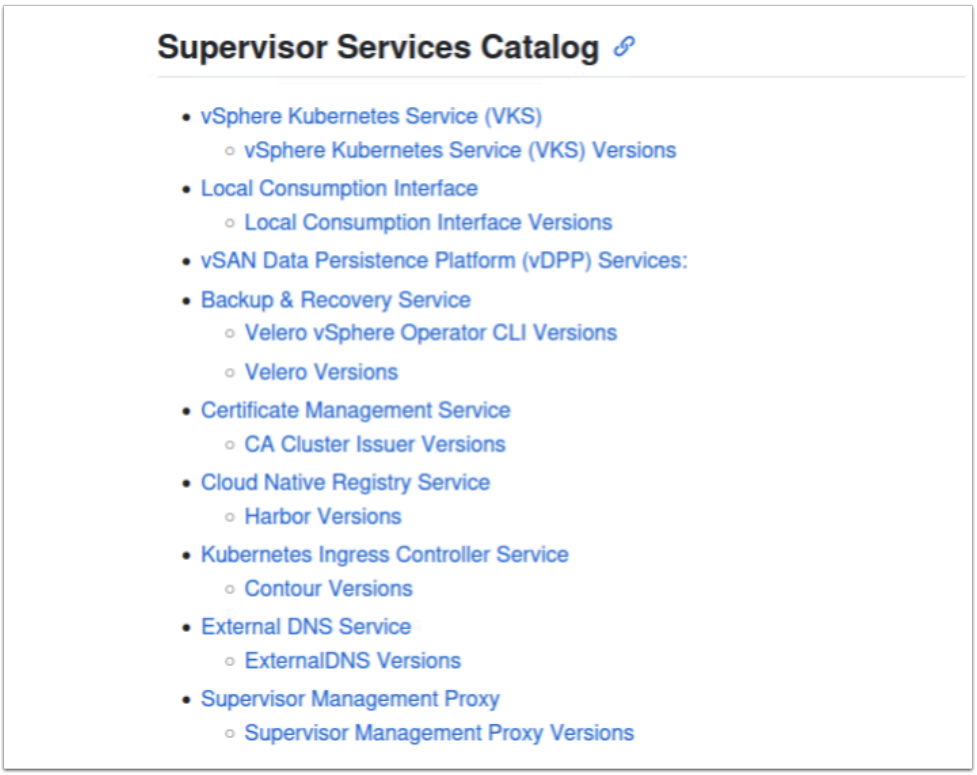


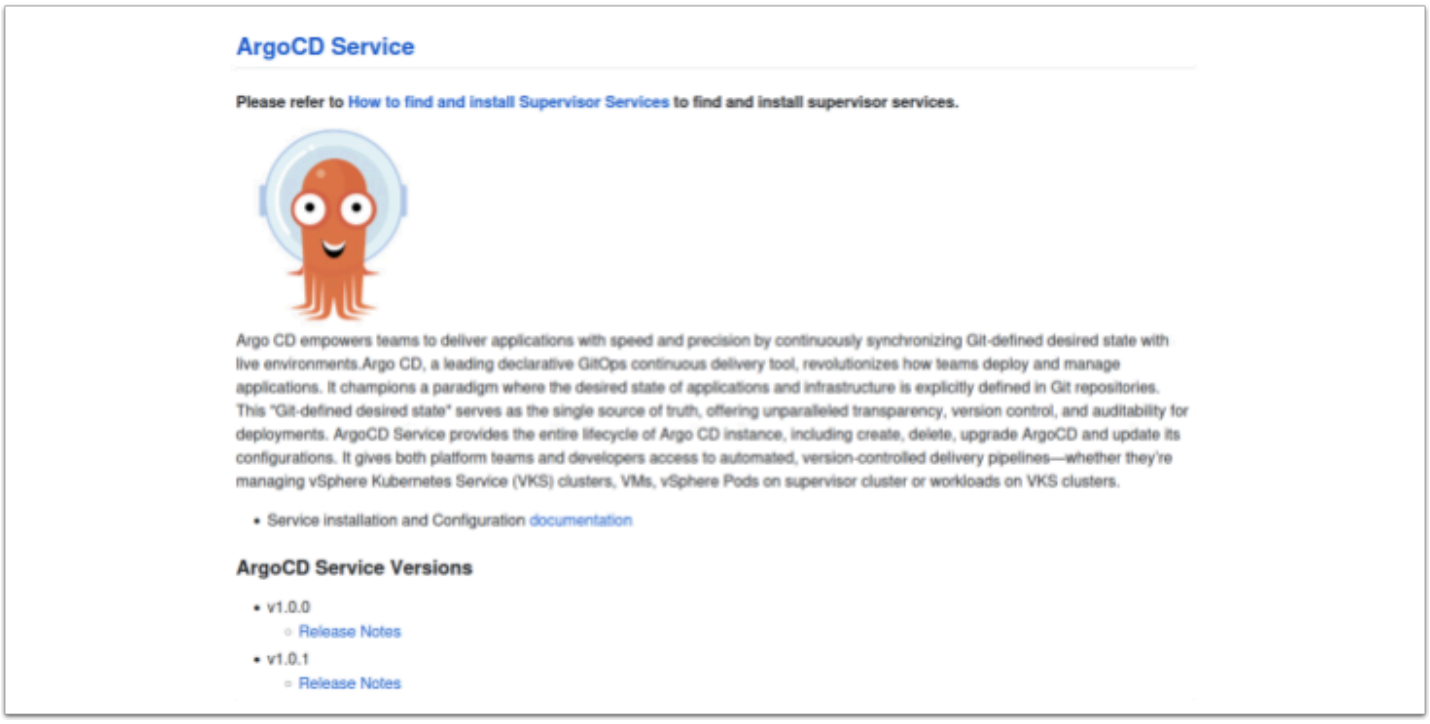
1. You can get a list of the currently supported services by clicking the ***Discover and download available Supervisor Services here*** link.
2. Click **UPLOAD** to find a service file to add.

> NOTE : Since this lab sits behind a proxy and firewall you will not be able to access these links from this lab these are for illustrative purposes only. The next few steps are illustrated for you since this lab is behind a firewall you can not access the supervisor services repository.

Download The ArgoCD YAML File

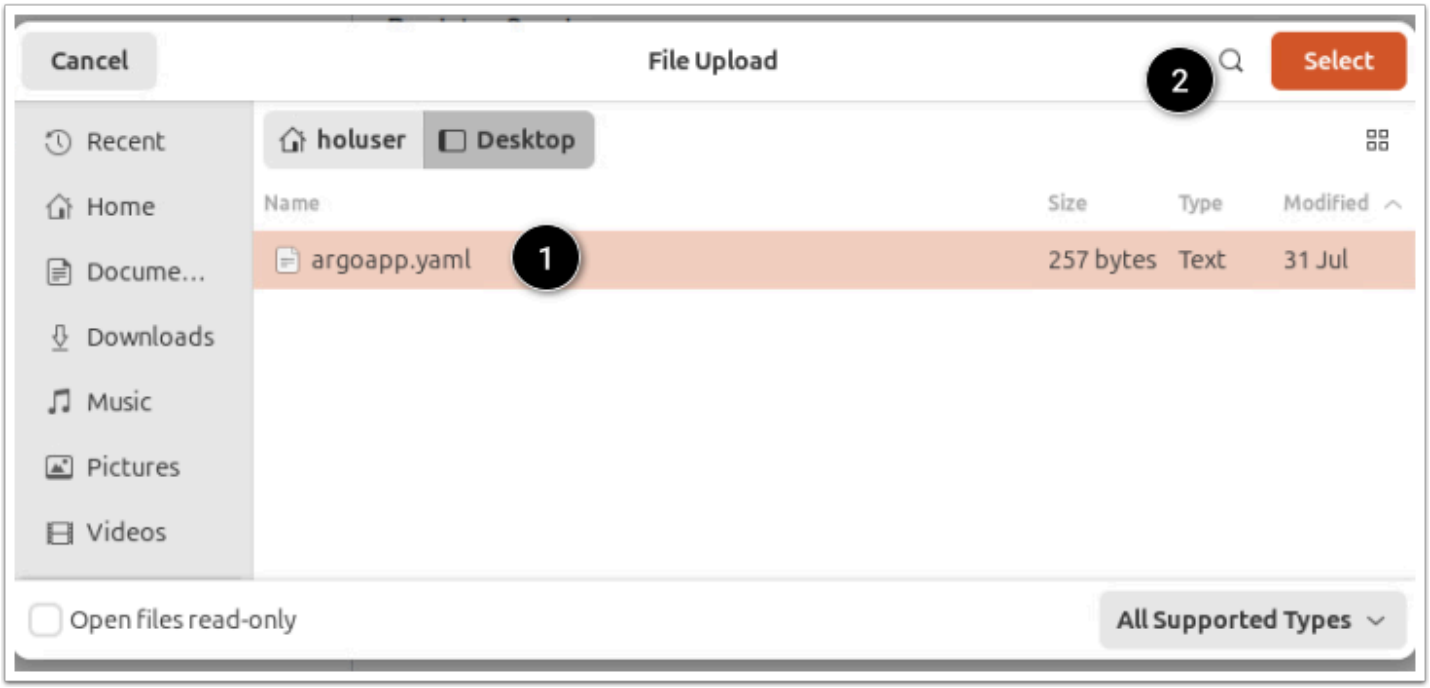
This is an image showing the the Supervisor Services Catalog.





Here you would scroll down until you see **ArgoCD** and you would select the version of the **ArgoCD Service yaml** and **cli** from the **Broadcom support portal**.

The ArgoCD YAML File

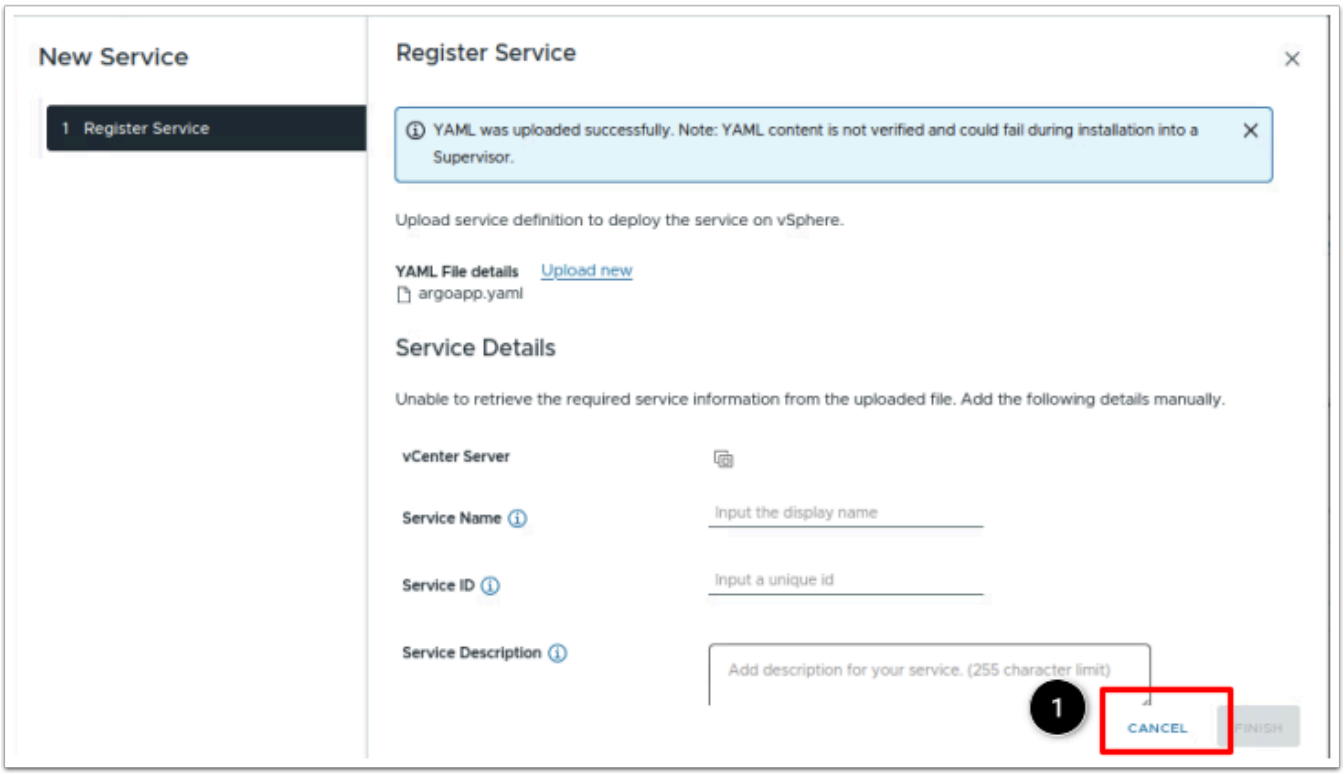


After downloading you would then:

1. Select the **argoapp.yaml** file.
2. Click **Select**.

> *NOTE : Since this lab sits behind a proxy and firewall you will not be able to access these links from this lab these are for illustrative purposes only.*

Register Service Cont'd



As you can see above, the YAML file has been decoded and shows that an ArgoCD service will be installed.

To proceed with the lab you must cancel out of the register service wizard.

1. Select **CANCEL** to close the wizard

Module 3 - Login to The Cluster and Deploy An Application (30 minutes) Basic

Module 3 - Login to The Cluster and Deploy An Application (30 minutes) Basic

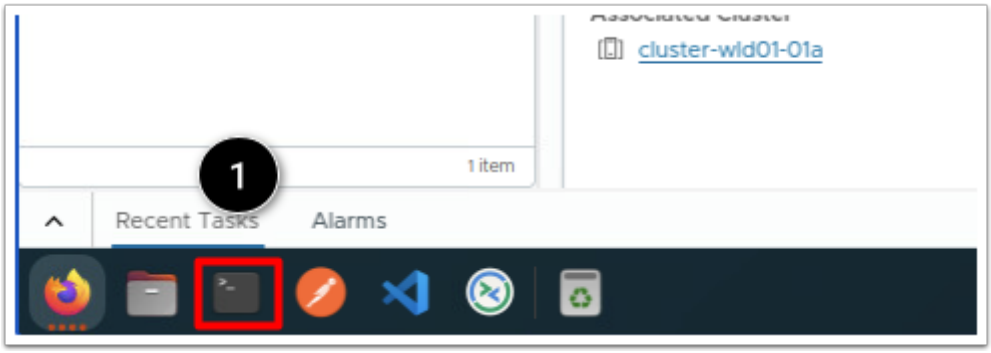
Deploy The nginx Application

In this module we will walk through the steps to deploy the **nginx** application in the namespace and cluster we created previously.

Verify the version of the VCF Command Line Interface (VCF CLI)

You need to complete Module 2 before continuing with Module 3. At a minimum, you need to create the VKS cluster

Please remember that you can copy and paste any of the following commands into the terminal window



1. Open a terminal window by clicking the terminal icon.

Check that you are logged into the supervisor by typing the following command

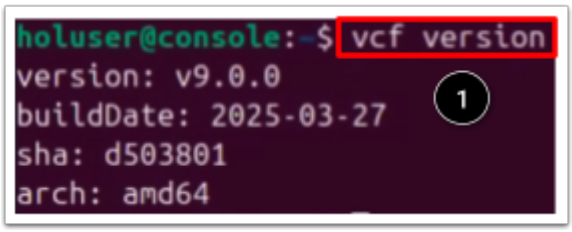
```
kubectl cluster-info
```

Click to copy

If you do not see the response control plane is running to the previous command, log into the Supervisor by typing the command

```
kubectl vsphere login --vsphere-username administrator@wld.sso --server=https://10.1.0.2 --insecure-skip-tls-verify
```

Click to copy

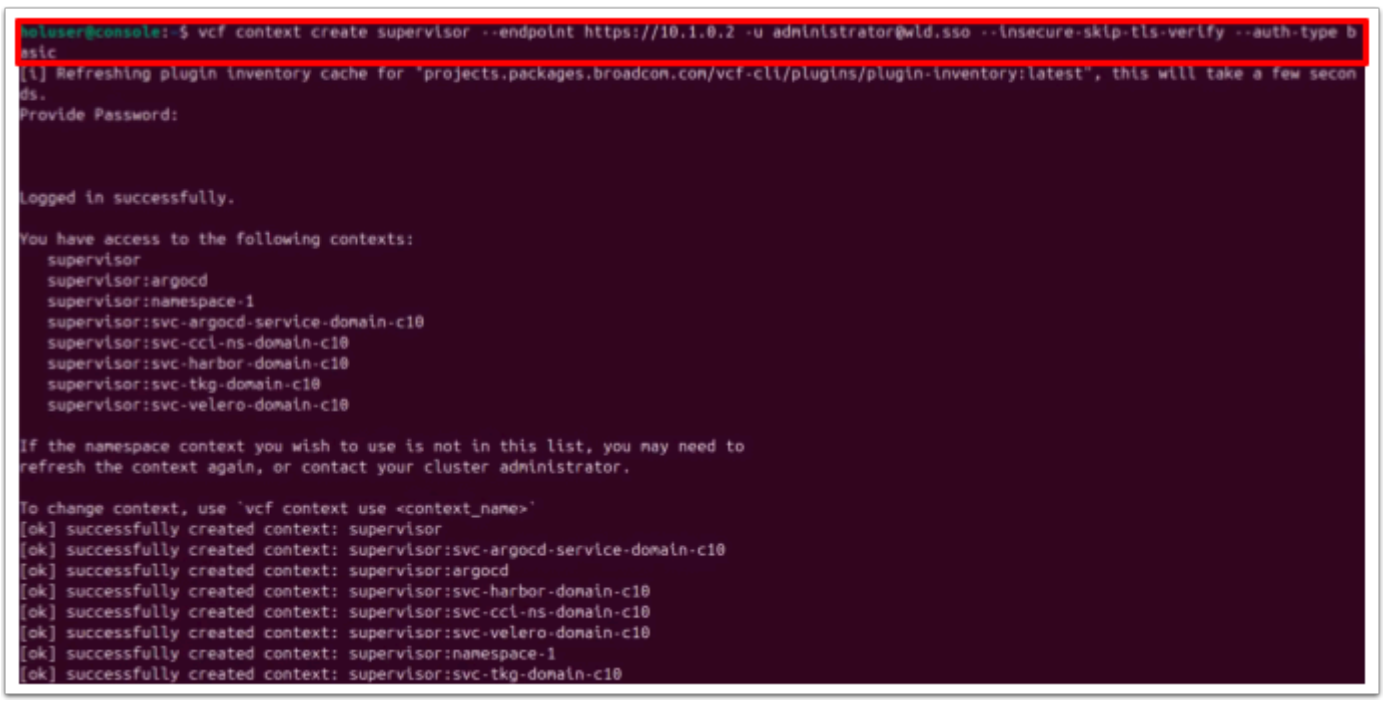


Validate the VCF version by typing the following command

```
vcf version
```

Click to copy

Create a VCF Context for Supervisor



Create a VCF context for Supervisor by typing the following command

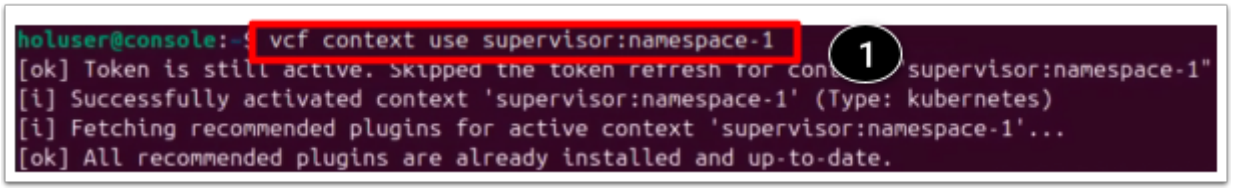
```
 vcf context create supervisor --endpoint https://10.1.0.2 -u administrator@wld.sso --
insecure-skip-tls-verify --auth-type basic
```

Click to copy
Password:

```
VMware123!VMware123!
```

Click to copy

Set Context to Namespace-1

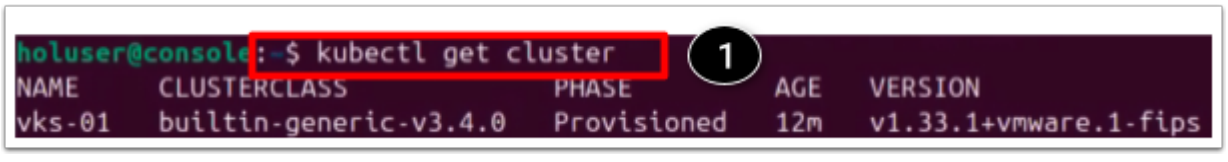


Set your newly created context to use the namespace you created in Module 2 (**namespace-1**) by typing the following command:

```
 vcf context use supervisor:namespace-1
```

Click to copy

Check Your Kubernetes Cluster

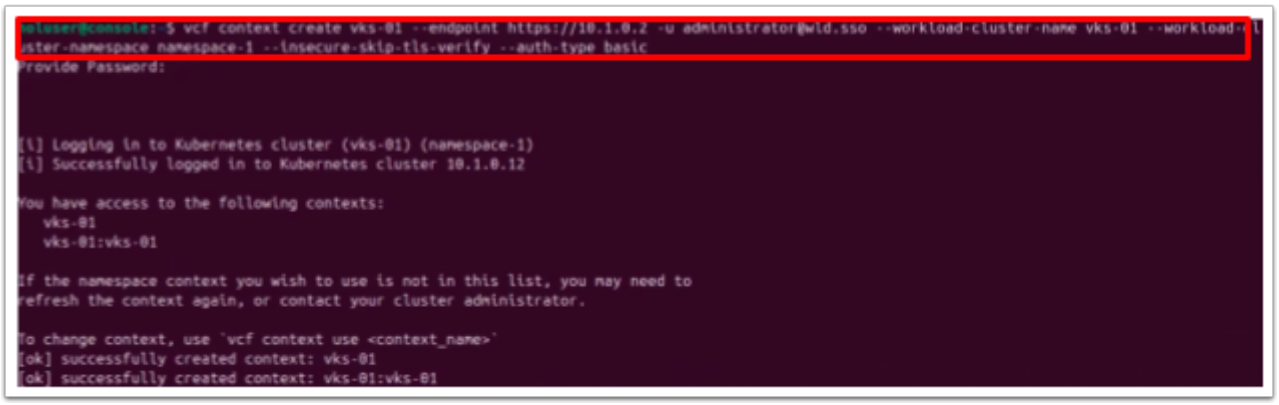


Check you kubernetes cluster by typing the following command

```
kubectl get cluster
```

Click to copy

Create the Context for the VKS Cluster



Create a VCF context for theVKS-01cluster by typing the following command

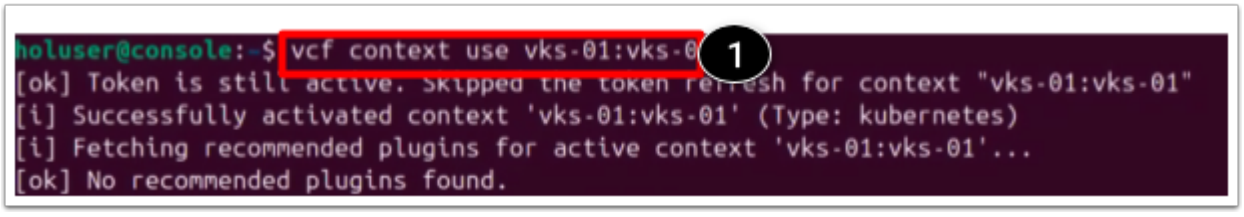
```
vcf context create vks-01 --endpoint https://10.1.0.2 -u administrator@wld.sso --
workload-cluster-name vks-01 --workload-cluster-namespace namespace-1 --insecure-skip-
tls-verify --auth-type basic
```

Click to copy
If prompted for a password use :

```
VMware123!VMware123!
```

Click to copy

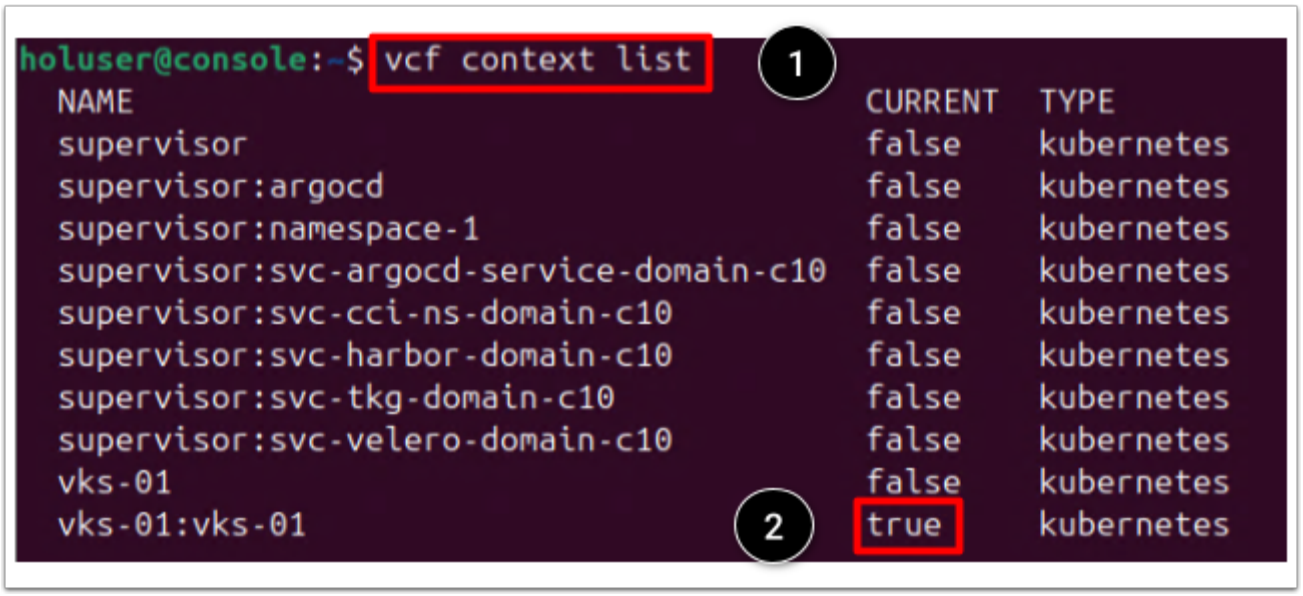
Set the Context to Use the VKS-01 Cluster



Set the context to the VKS-01 Cluster by typing the following command

```
vcf context use vks-01:vks-01
```

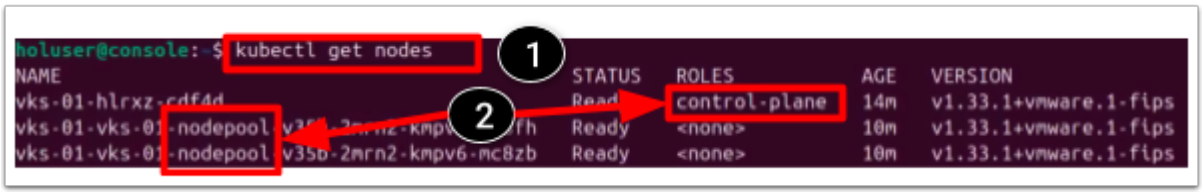
Click to copy



To verify that you are using the correct context type the following command

```
vcf context list
```

Click to copy
Verify that the c**ontext vks-01:vks-01** is set to **true**

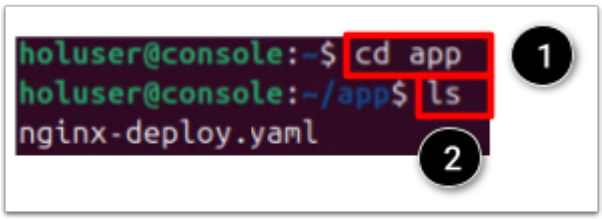


You can also validate the context is correct by typing the following command

```
kubectl get nodes
```

Click to copy
Verify that you can see the **control-plane** and **nodepool nodes**

Let's Deploy An Application



We'll deploy the **nginx** application in the **vks-01cluster**. Let's start by changing to the app directory, type the following command

```
cd ~/app
```

Click to copy

Let's review the files for the application we are going to deploy, type the following commands:

```
cat nginx-deploy.yaml
```

Click to copy
Review the yaml file and move on. No changes to the file are necessary.

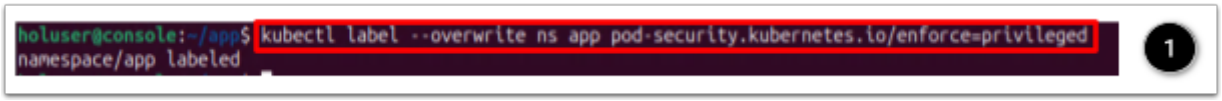
Create a Namespace for the Application on the VKS-01 Cluster

Create a kubernetes namespace by typing the following command

```
kubectl create ns app
```

Click to copy

Set the Pod Security Policy for the Namespace

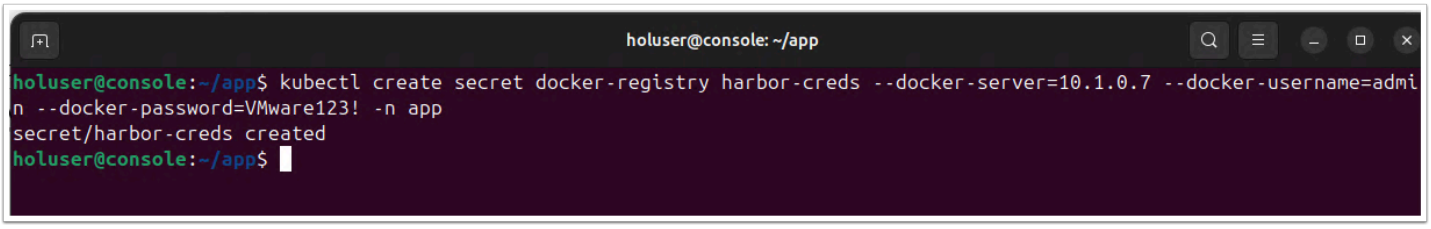


Set the pod security policy by typing the following command

```
kubectl label --overwrite ns app pod-security.kubernetes.io/enforce=privileged
```

Click to copy

Create A Pull Secret For Harbor

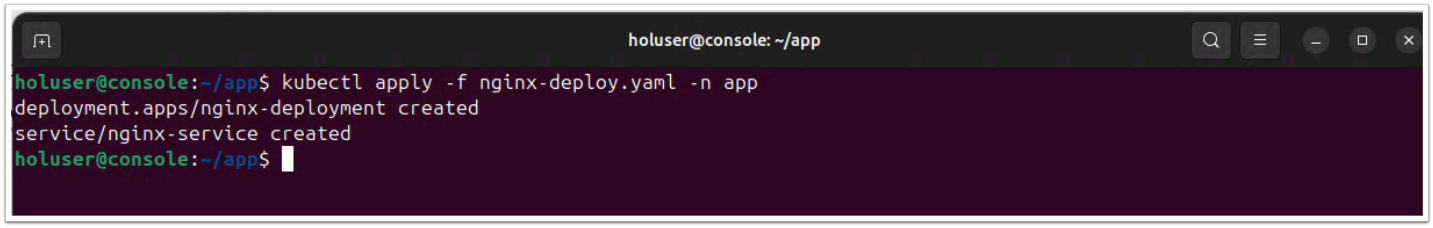


To create the pull secret for Harbor type the following command

```
kubectl create secret docker-registry harbor-creds --docker-server=10.1.0.7 --docker-
username=admin --docker-password=VMware123! -n app
```

Click to copy

Deploy the nginx Application in the app Namespace

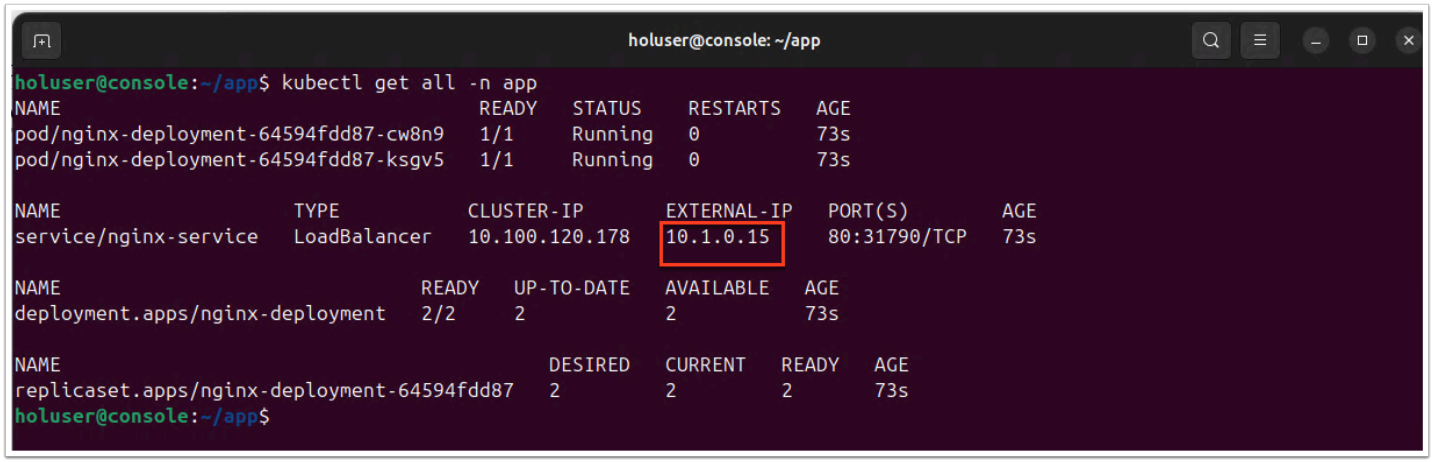


To deploy the **nginx** application type the following command

```
kubectl apply -f nginx-deploy.yaml -n app
```

Click to copy
You should see the message above that the service was created.

List All of the Resources Deployed for the Application In the Namespace



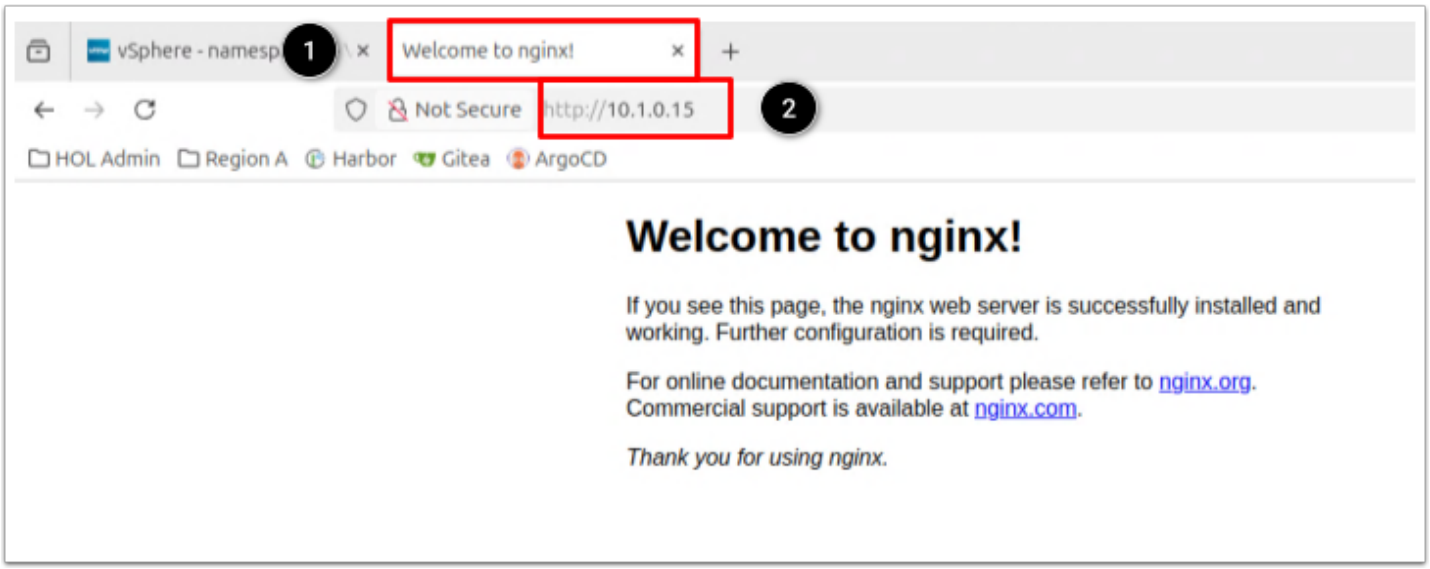
Check the details of the deployed **nginx** application. To see all the services deployed type the following command

Take note of the **external** IP address of the **nginx** application

```
kubectl get all -n app
```

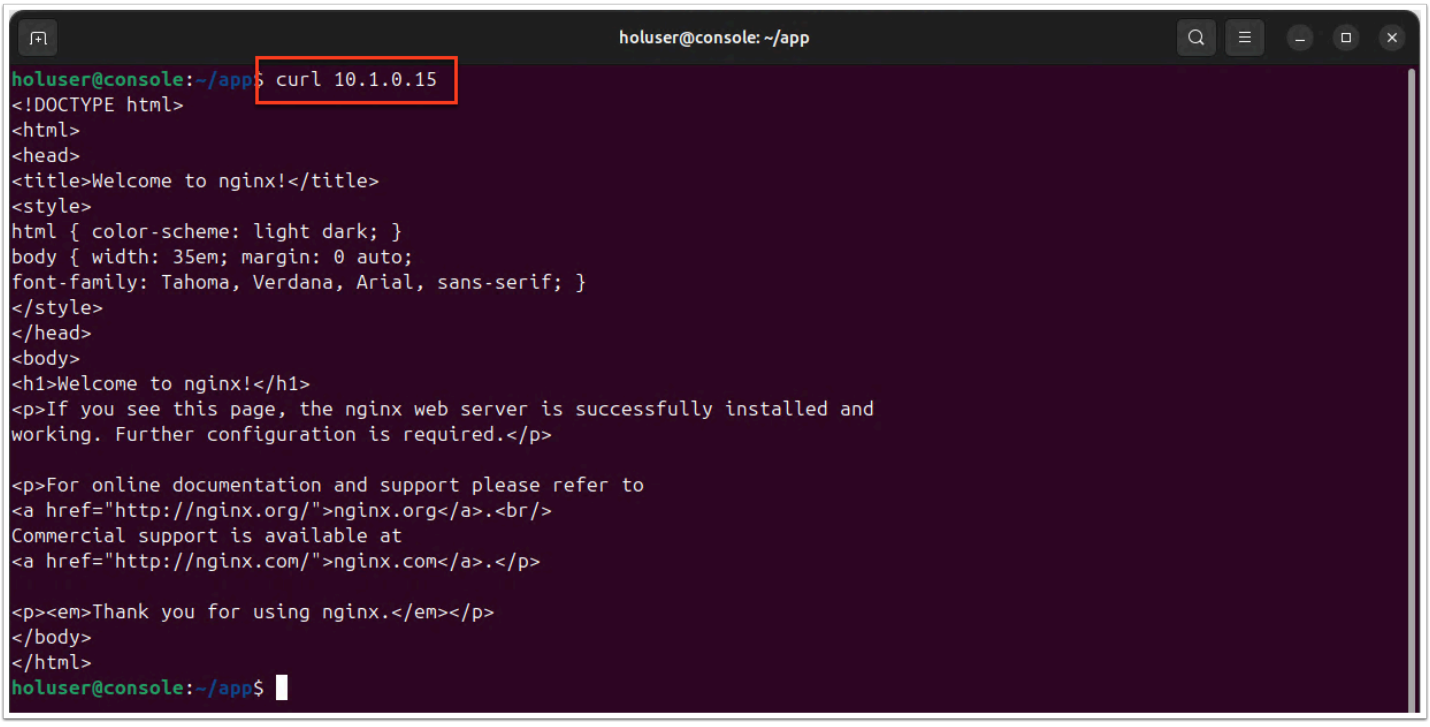
Click to copy

Check that nginx Is Working



Check that **nginx** is running. Return to the browser

1. Open a new tab in the browser
2. Enter the external IP address gathered in the previous step



You can also use the curl command to validate **nginx** is running by typing the following command

```
curl xx.xx.xx.xx
```

Click to copy
Use the external IP address from the previous step

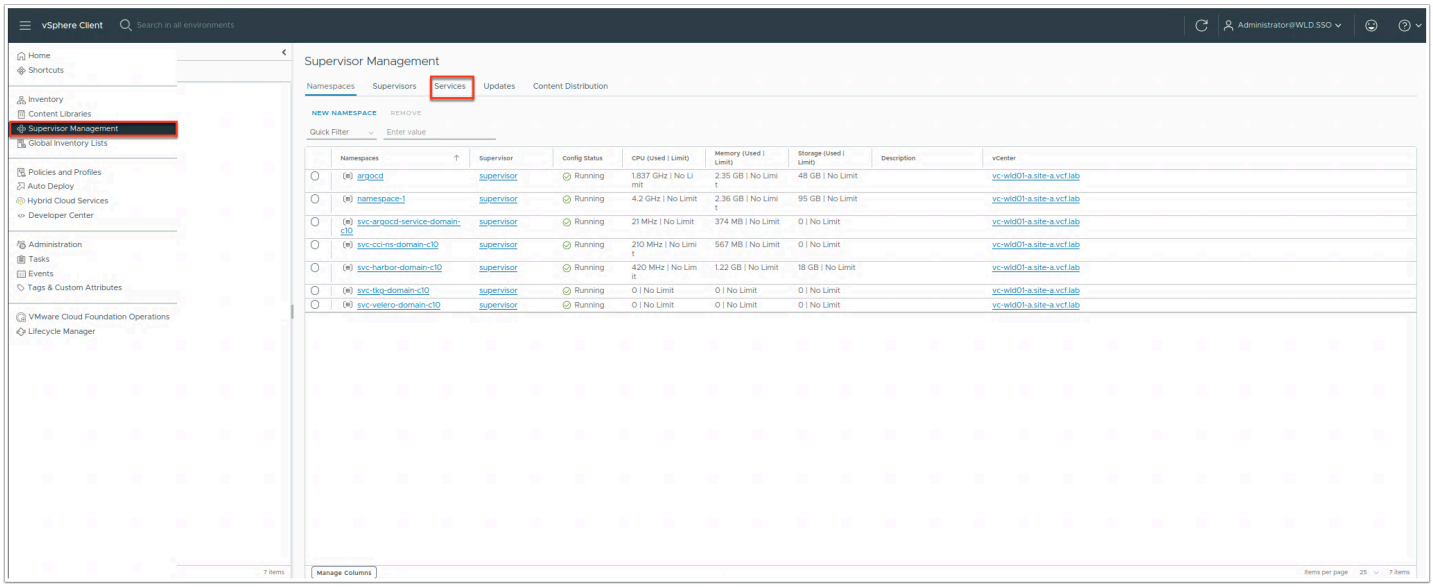
Module 4 - Updating The Supervisor and VKS Services (15 minutes) Basic

Module 4 - Updating The Supervisor and VKS Services (15 minutes) Basic

VKS Update

In Module 2 of this lab we deployed a version of the vSphere Kubernetes Service, we deployed the 1.33 version which at this time was the latest available. But, kubernetes versions change quickly so you will need to update the VKS service to stay in lockstep with the rapid pace of new kubernetes releases.

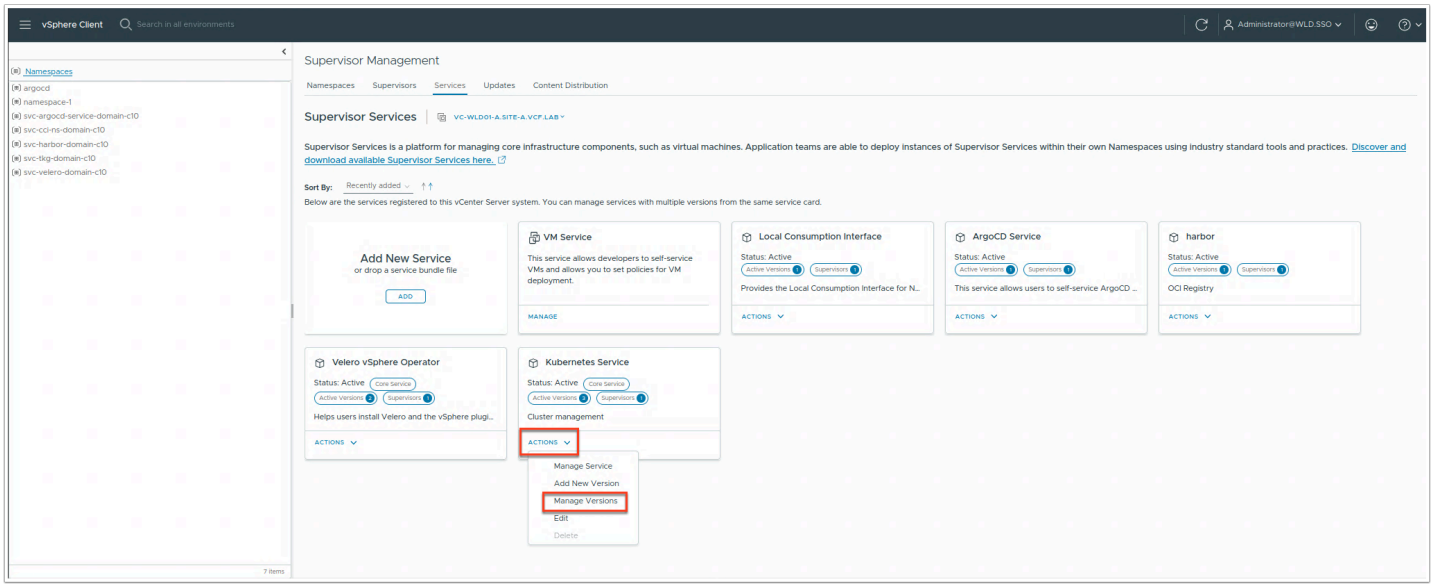
Update The VKS Service



At any given time there will be multiple versions of the VKS service available for download. You can review the available versions by executing the following steps:

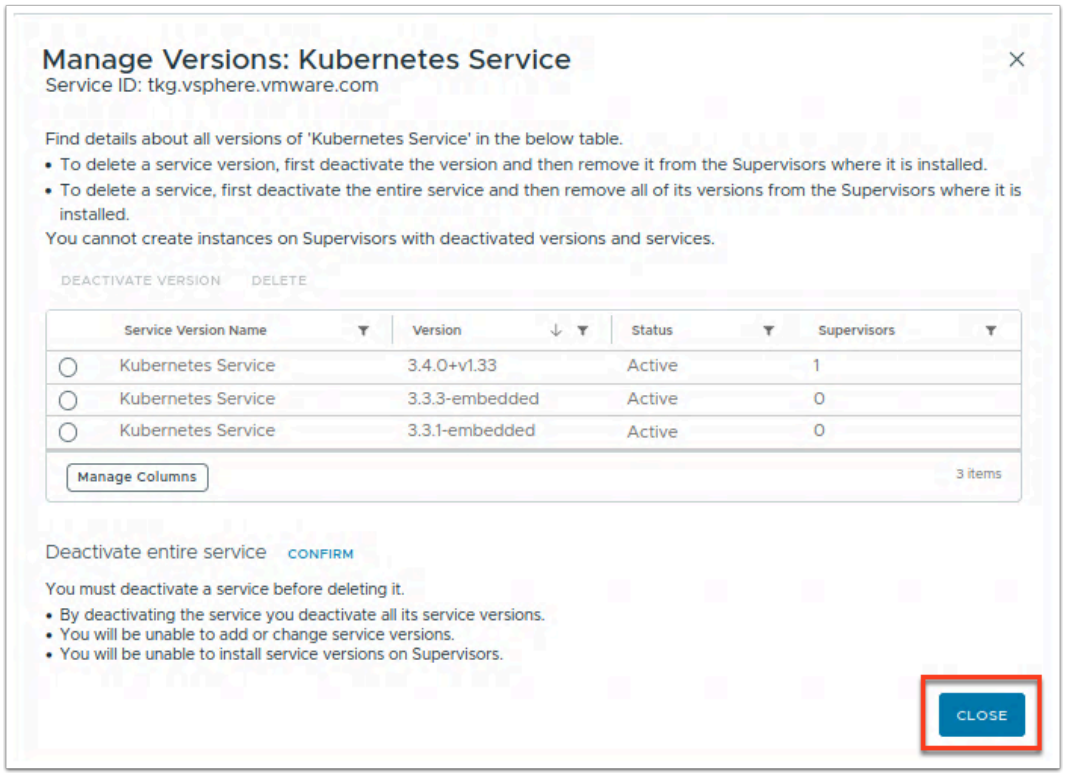
1. Navigate to S**upervisor Management**
2. Click **Services**

Update The VKS Service(Contd.)



Under Kubernetes Services:

1. Click **Manage Versions**

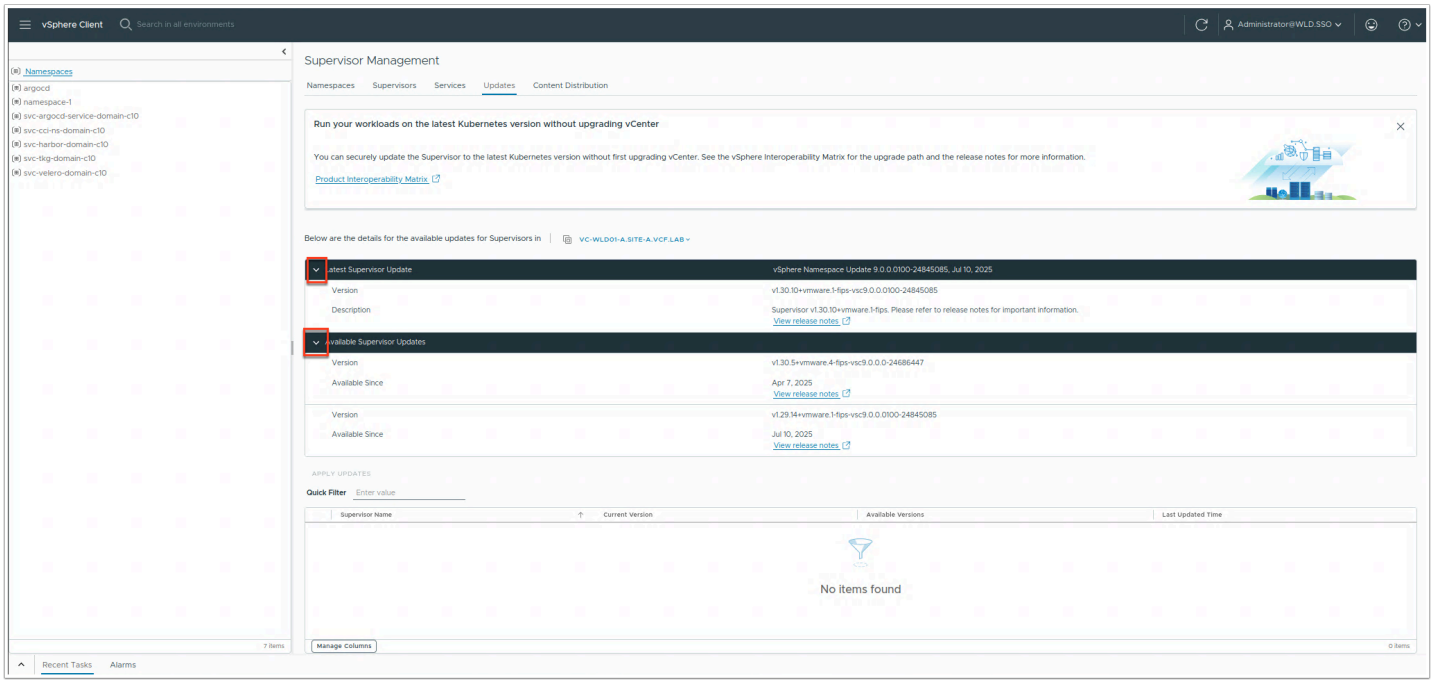


This is where Platform Admins will select the new version to install. The lifecycle of the various supervisor services are independent. Some services such as Velero are considered core/critical services and are installed by default while other such as ArgoCD which we will cover later are optional and installed by the Platform Administrator in either case, ,you have full flexibility to

update (or not) to a new version as soon as they are released, without having to update VCF/ vCenter, the supervisor or other supervisor services.

1. Click **CLOSE** when you are done reviewing.

Supervisor Updates

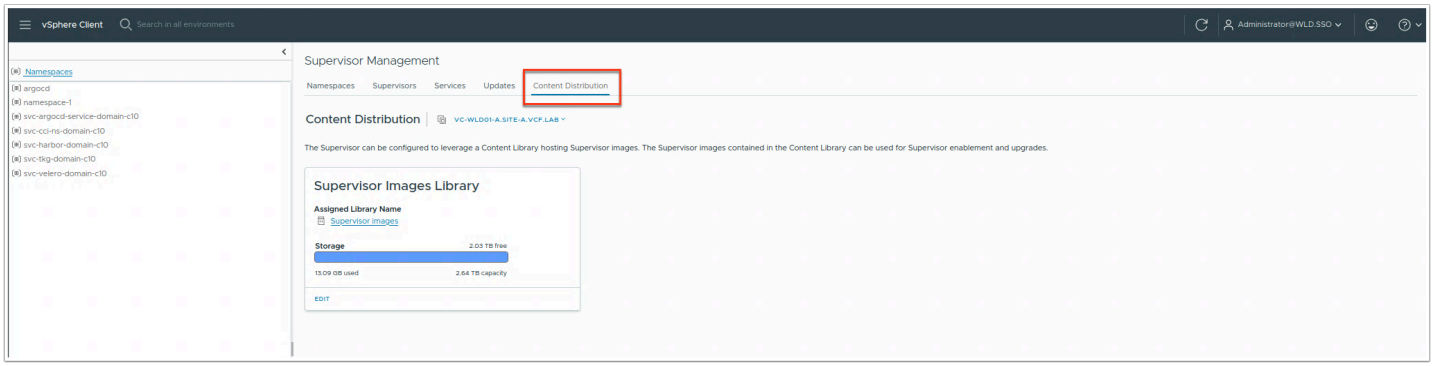


Navigate to **Supervisor Management - Updates**

This lab has been already updated for you but review the sequence to update the supervisor.

1. Always a good idea to review the compatibility matrix.
2. And always good to review the release notes.

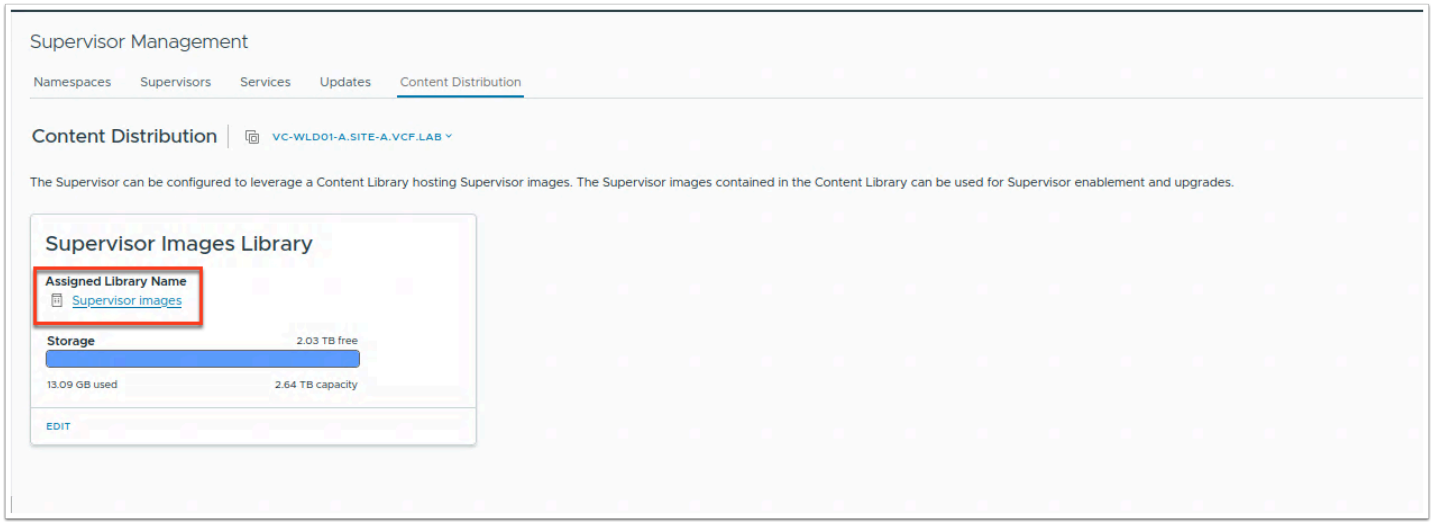
Content Distribution



1. Click **Content Distribution**.

Updates are distributed through a content library configured in Content Distribution tab.

Edit Settings

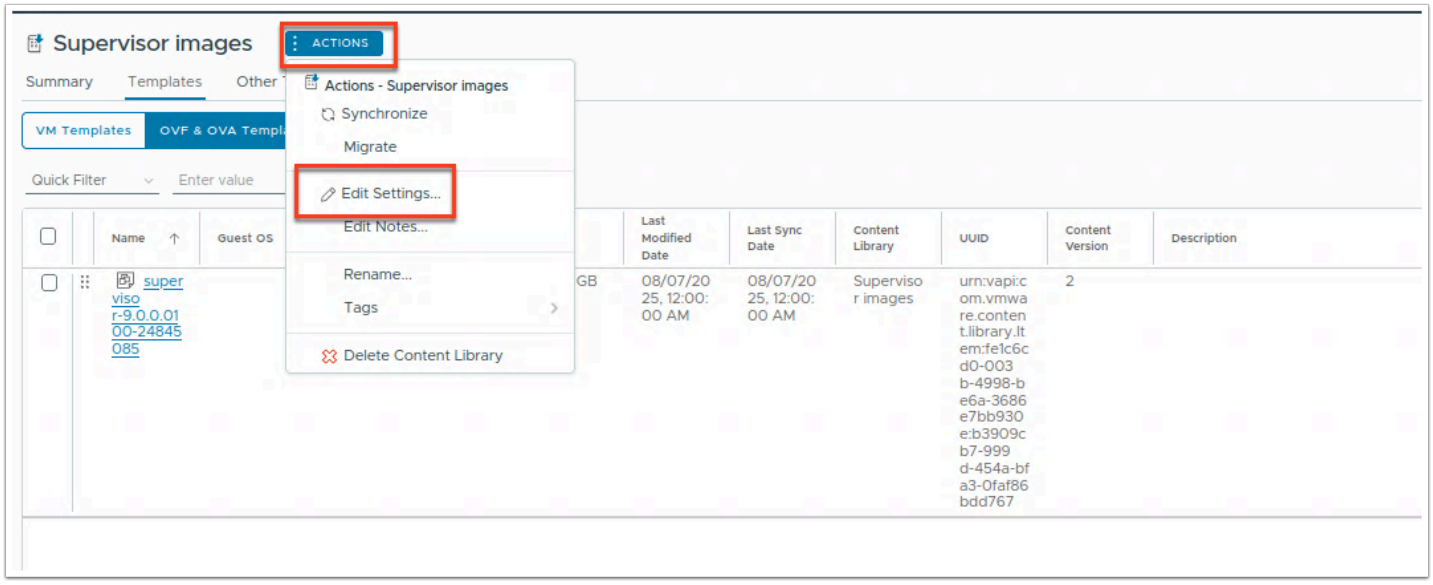


You can edit the settings for the supervisor services image.

1. Click **Supervisor images**.

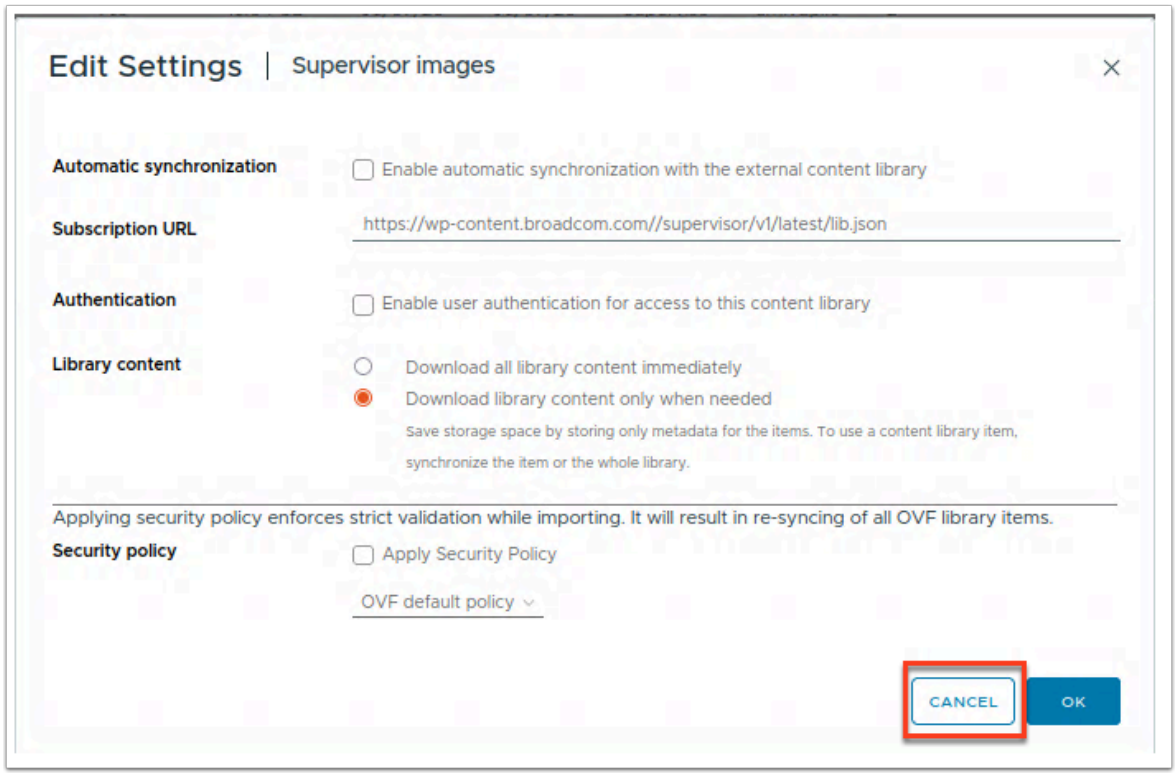
This is important moving forward as new Supervisor releases bring not only new version of kubernetes for control plane, but also new functionality and services. This first one brought ArgoCD Service.

Edit The Supervisor Image



1. Click the **Actions** dropdown.
2. Click **Edit Settings**.

Edit The Supervisor Images



Notice that here you can enable automatic synchronization, downloads and security policy.

1. Click **CANCEL**.

Module 5 -Unlocking GitOps with ArgoCD Service(20 minutes) Advanced

Module 5 -Unlocking GitOps with ArgoCD Service(20 minutes) Advanced

In this module we will be deploying an application using a combination of Gitea which is a is a free and open-source software forge for hosting Git repositories, providing features similar to GitHub but designed for self-hosting. And ArgoCD which s a declarative, GitOps continuous delivery tool for Kubernetes. It automates the deployment of applications to specified target environments by continuously comparing the running state of applications with the desired state defined in a Git repository. Argo CD then automatically synchronizes the applications to match the desired state, ensuring consistency and enabling automated deployments.

With VCF, ArgoCD is an optional supervisor service which would be deployed by the platform administrator in this lab the ArgoCD service has already been installed and an instance deployed for you.

Review the ArgoCD service

We'll start by checking the status of the service

1. Click the terminal icon in the taskbar and open a terminal window.

Switch context to argocd namespace

For this lab we have created the ArgoCD service in the namespace argocd. Switch context to use the argocd name space, type the command:

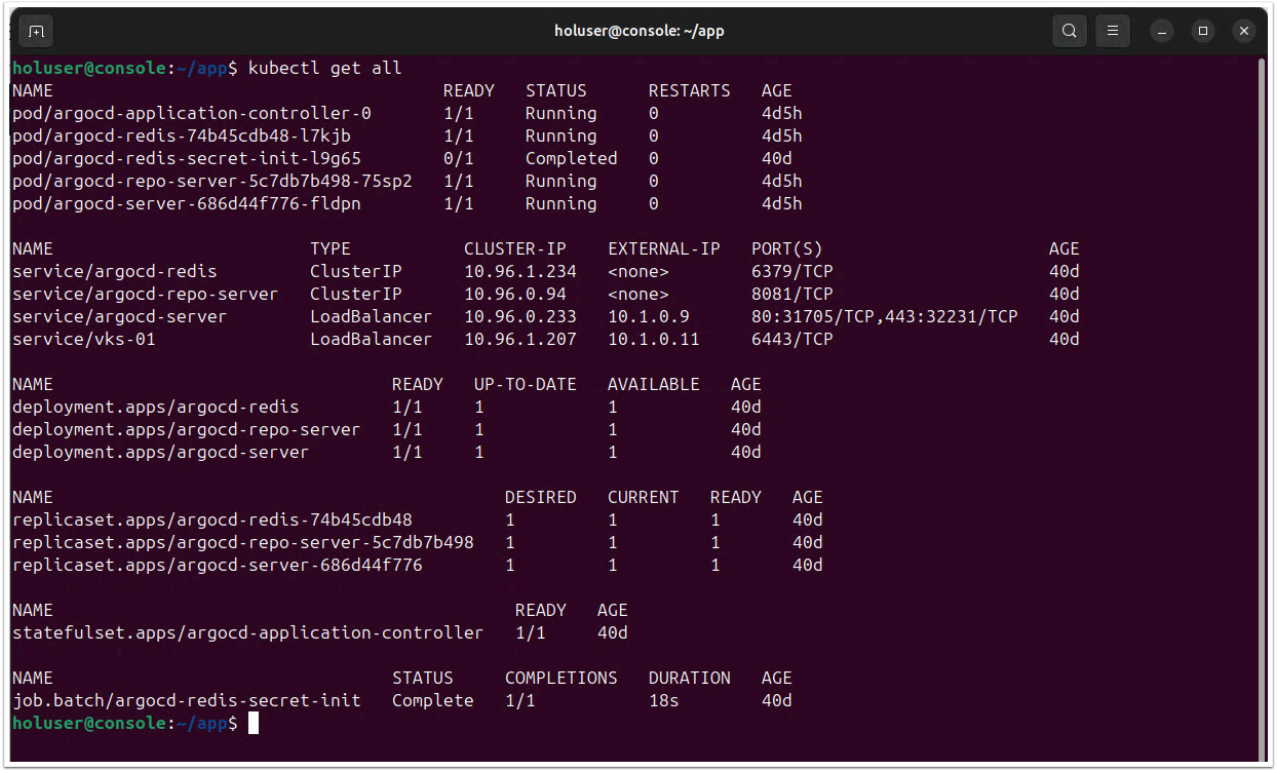
```
vcf context use supervisor:argocd
```

Click to copy
If prompted for a password use:

```
VMware123!VMware123!
```

Click to copy

Review the deployment

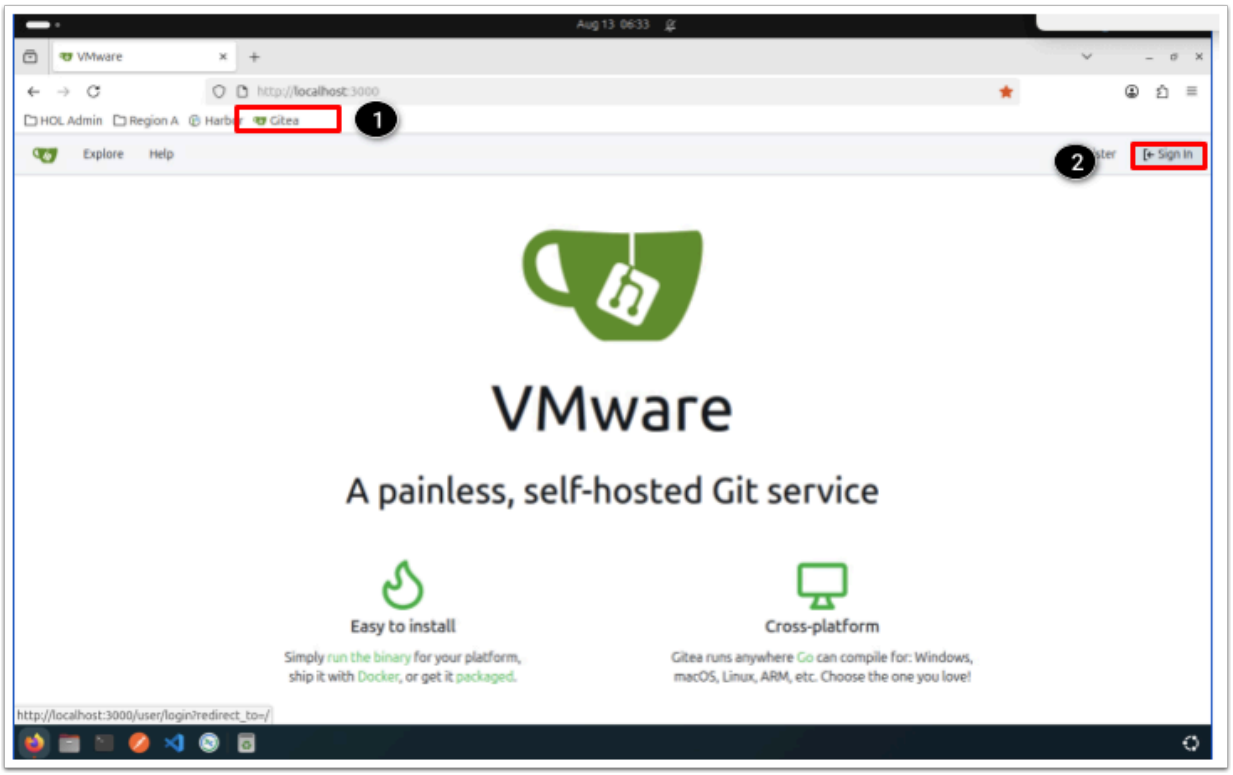


Review the deployment by typing:

```
kubectl get all
```

Click to copy

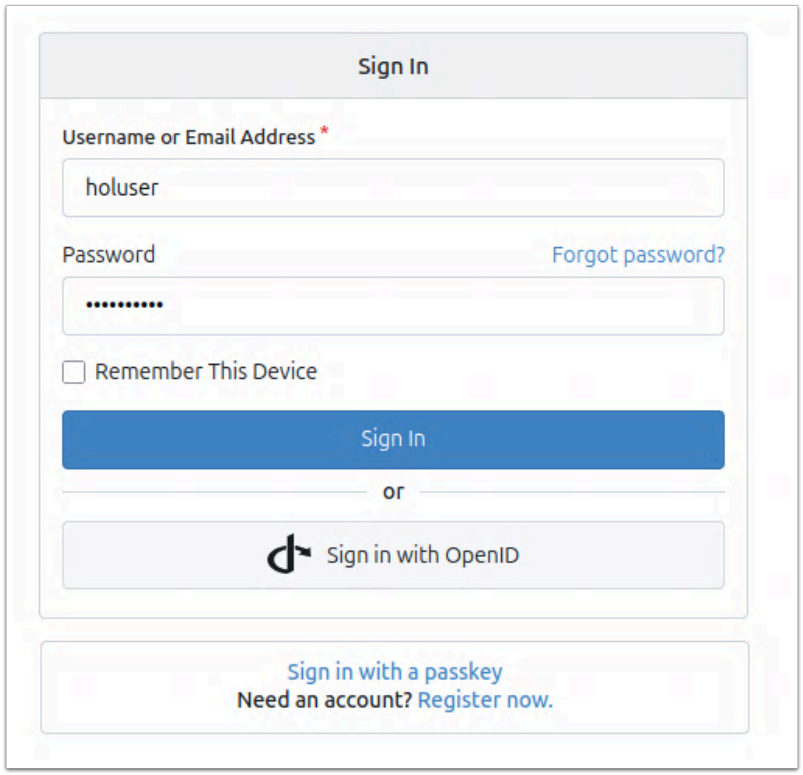
Open A Browser Tab to Gitea



If you are not familiar with Gitea it is a lightweight, self-hosted Git service that provides a platform for software development and version control, similar to GitHub. We have already deployed an instance of Gitea for you

1. Open the **giteea** browser tab in a new tab (you will need to return to gitea later)
2. Click **Sign In** .

Login to Gitea



1. Login with : Username

```
holuser
```

Click to copy

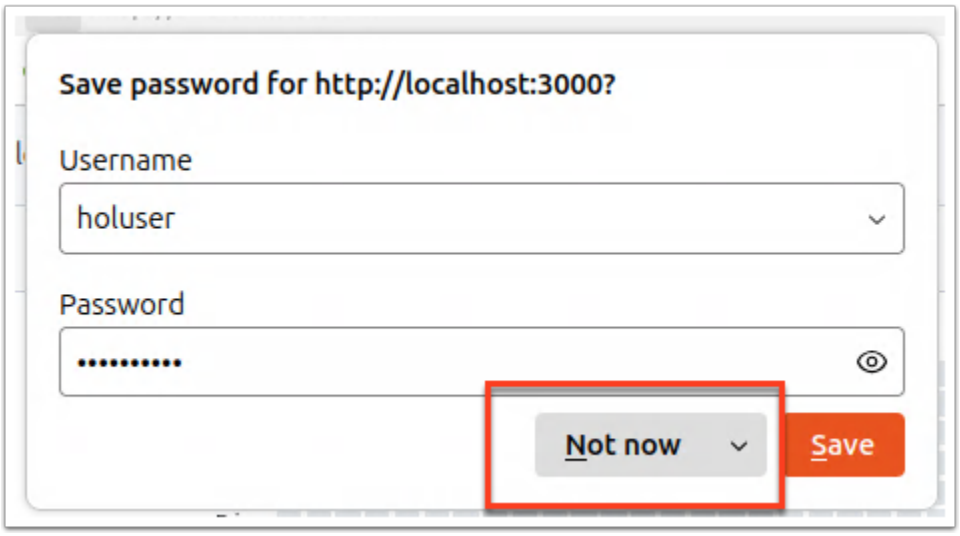
2. Password:

```
VMware123!
```

Click to copy

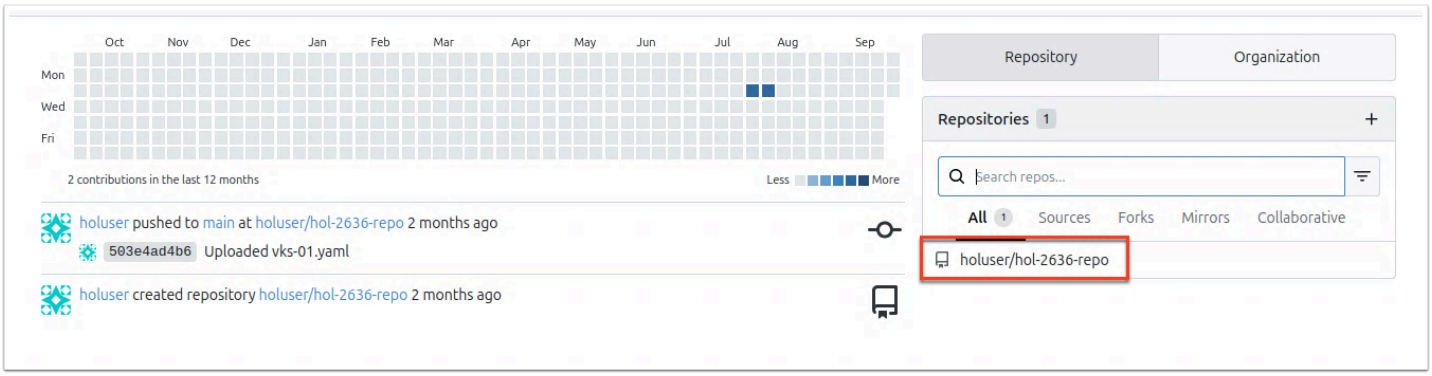
3. Click **Sign In**

Do not close the **gitea** tab as we will need it later in this module.



If this window pops up click **Not now**

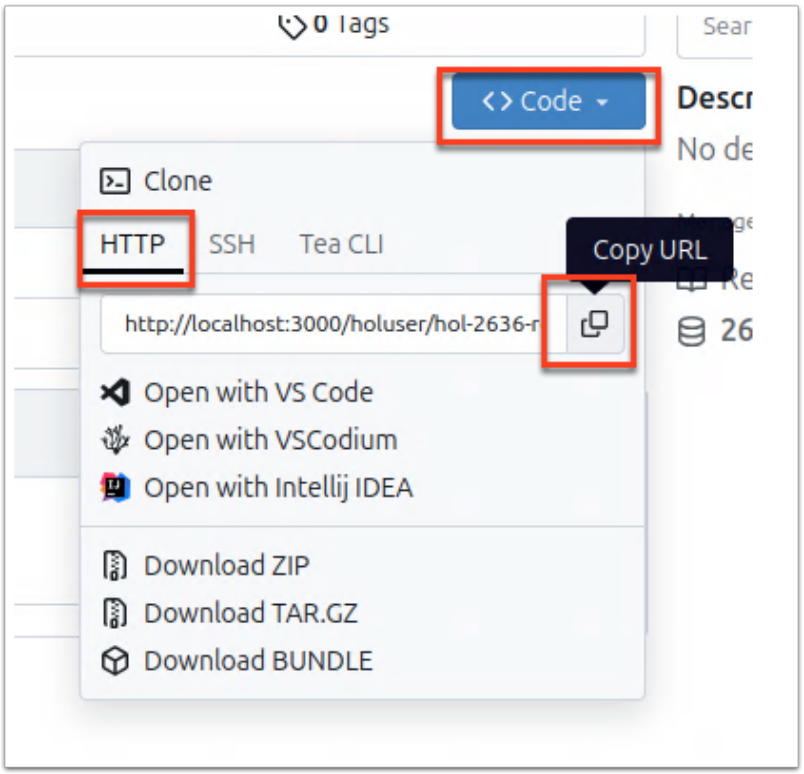
Switch to the holuser repository



The building team, to save time, has created and populated a repository ***holusser*** for you.

1. Click on **holuser/2636-repo**

Click the Code dropdown

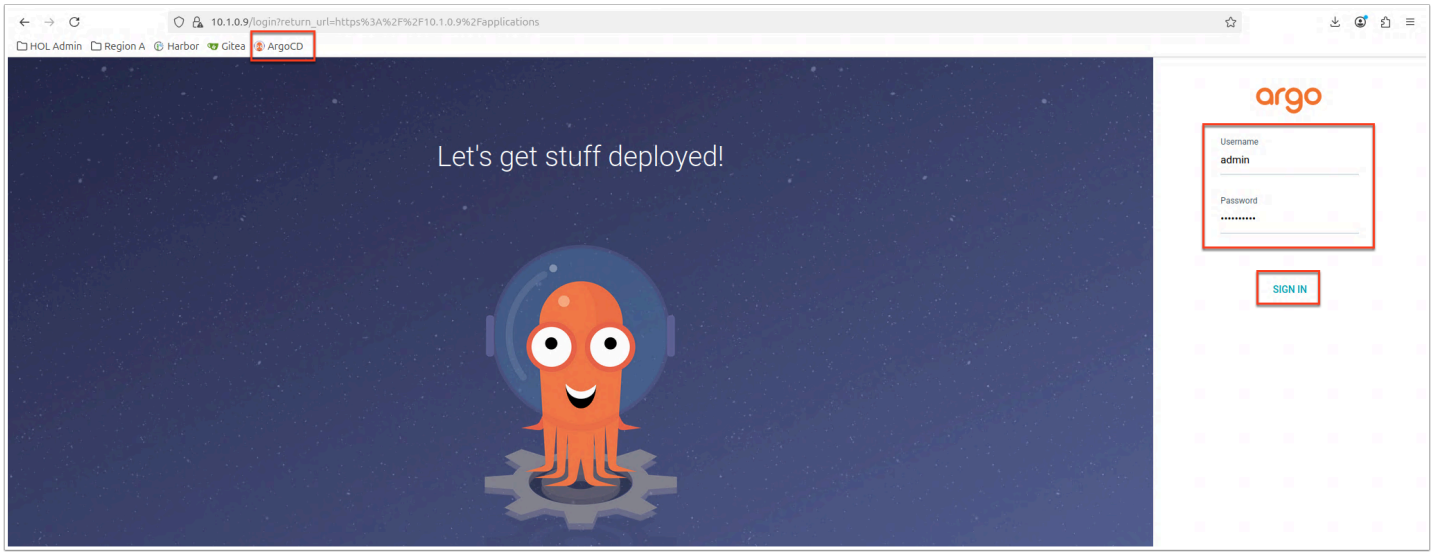


Here is where you would copy the URL for the repository which you would point ArgcCO to. In the YAML code which follows you will see we filled the repository URL in for you however, normally you would need to copy this URL to use when pointing ArgoCD to the code repository.

1. Click the **Code** dropdown
2. Copy the **URL** for the **holuser/2636-repo.**

> Note that we do not actually use this URL in the next steps; this is just an example of where/how you would obtain the URL for the repository for ArgoCD.

Login to ArgoCD



Now, let's login to the ArcoCD supervisor service we created for you for this lab. Remember that ArgoCD is an optional supervisor service which will not be installed by default.

We have created an app for you, VKS-1 which was created by pulling a file from the gitea repository.

1. Open the ArgoCD in a new browser tab.
2. Login to ArgoCD instance with:

Username:

```
holuser
```

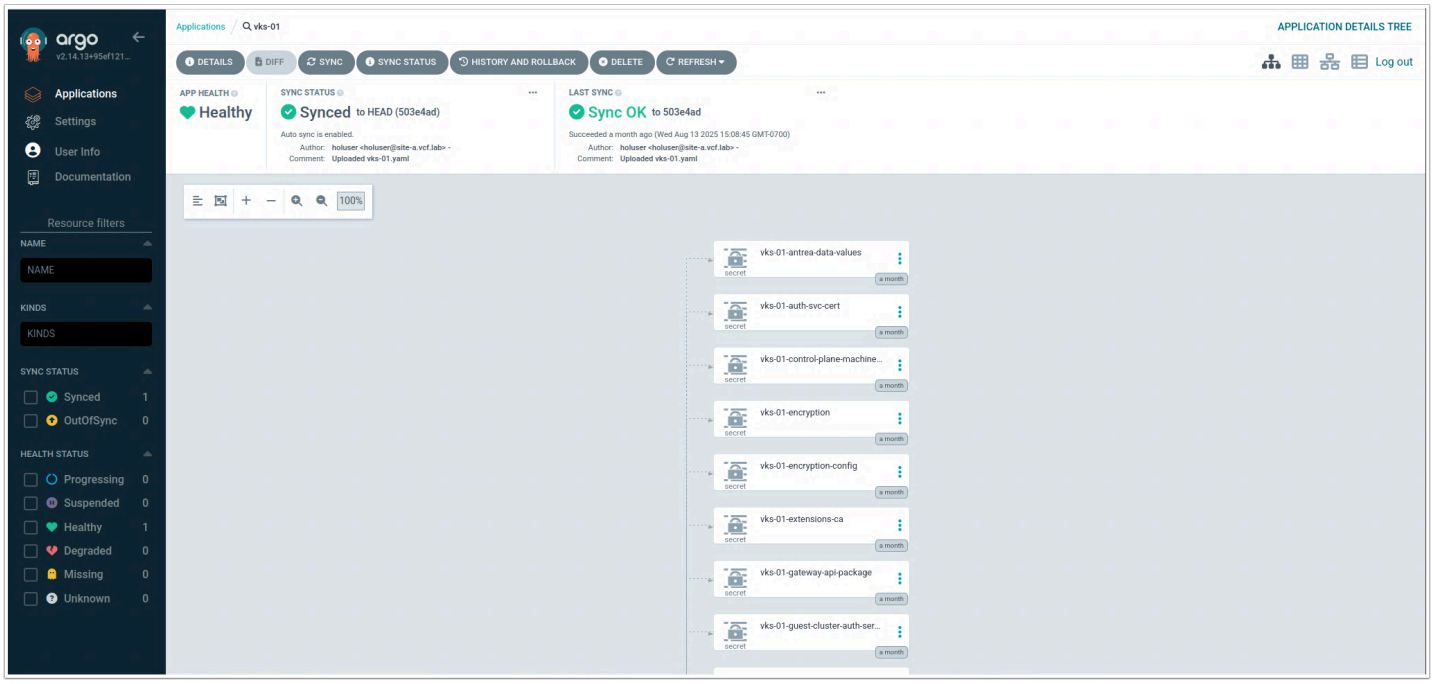
Click to copy

Password:

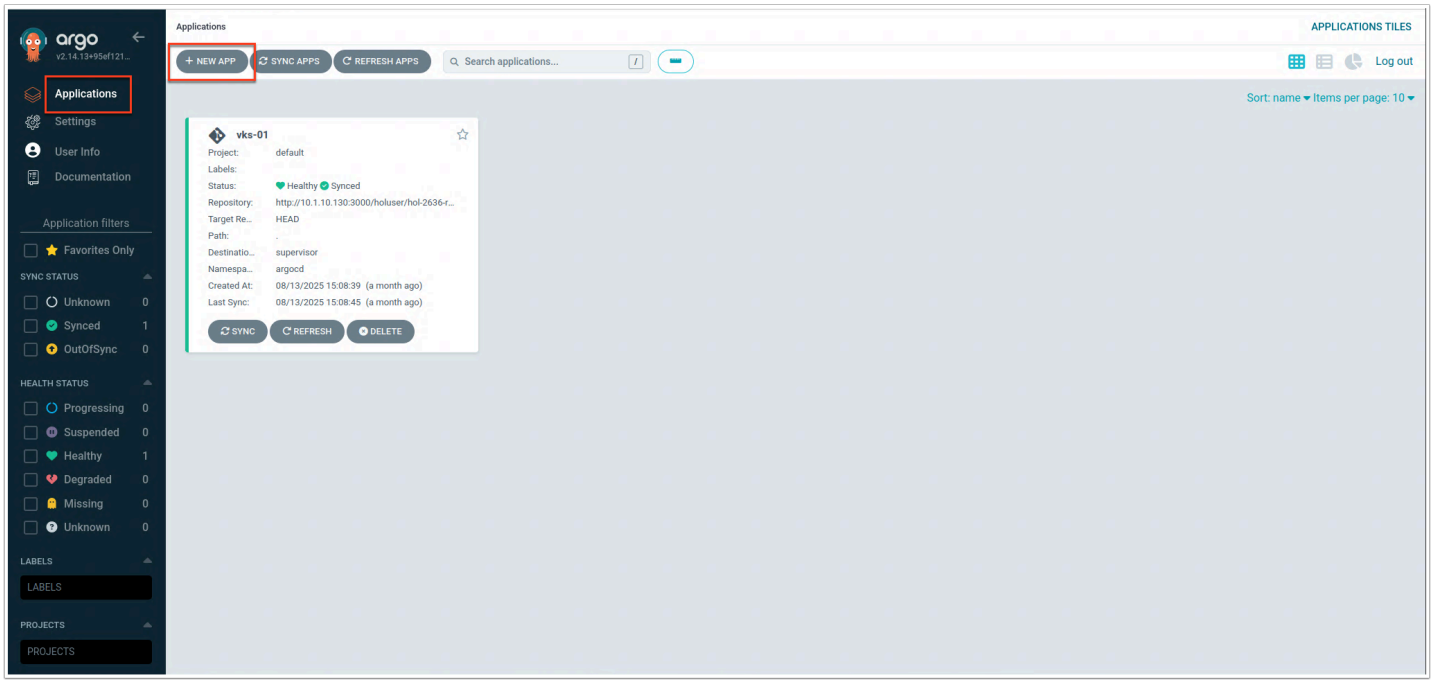
```
VMware123!
```

Click to copy

3. Click **Sign In**



How To Create an App Using ArgoCD

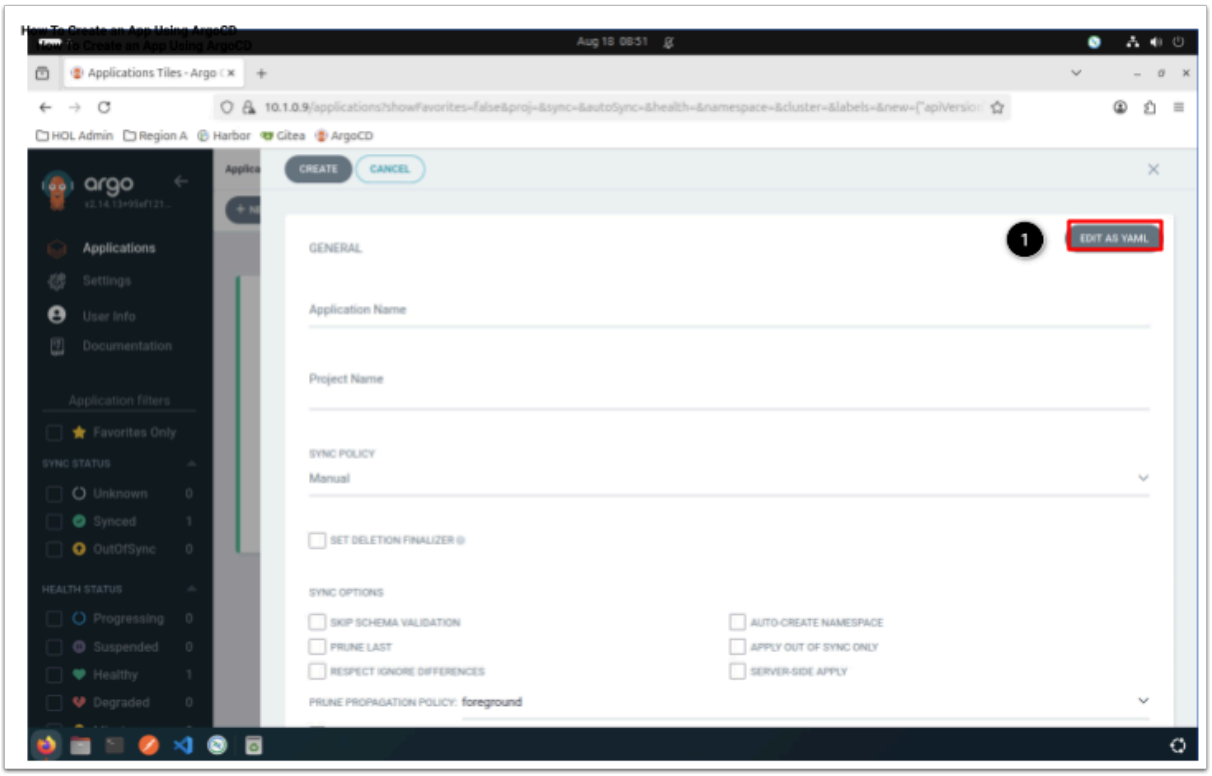


We have already created an app for you, **vks-01** using code pulled from the repository gitea. In the next few steps we will walk through the process by which you would create an application.

Note these steps are for reference and would fail if you tried to deploy the actual application (since vks-01 already exists).

1. Click on **Applications** in the left **menu** (under argo)
2. Click **+New App**

How To Create an App Using ArgoCD (contd.)



To create a new application in Argo CD you can walk through a guided wizard or enter YAML code.

By default the wizard is deployed. You can directly edit YAML code by clicking “EDIT AS YAML”

Here is an example using the YAML we used to create the VKS-1 application (note the repoURL is filled in for you in the code block, normally you would have to provide this, the repoURL being the URL you pulled from gitea previously)

Example only you do not need to use this code

```
apiVersion: argoproj.io/v1alpha1
 kind: Application
 metadata:
  name: vks-01
 spec:
  destination:
   namespace: argocd
   server: https://10.1.0.2:443
  source:
```

```
  path: .
  repoURL: http://10.1.10.130:3000/holuser/hol-2636-repo.git
  targetRevision: HEAD
  directory:
   recurse: true
 sources: []
 project: default
 syncPolicy:
  automated:
   prune: false
   selfHeal: false
```

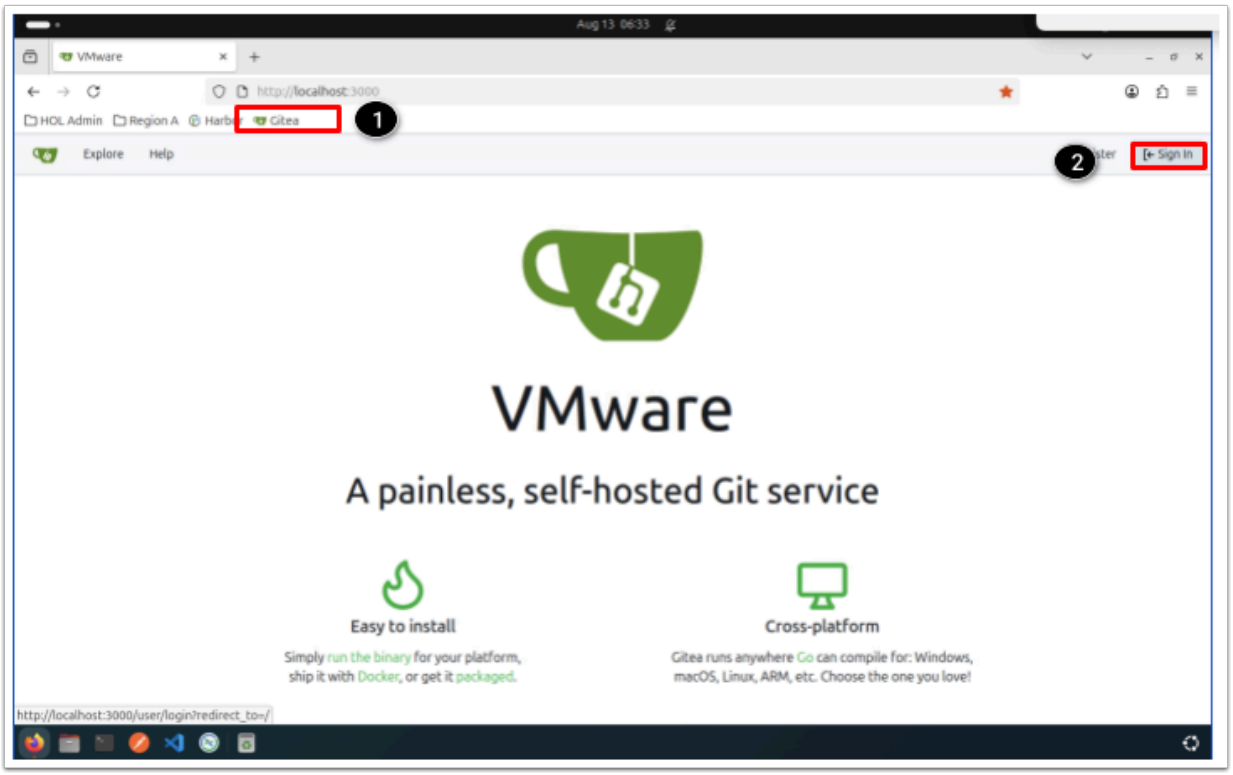
Click to copy

How To Create an App Using ArgoCD (contd.)

Note that the fields in the YAML code match the prompts in the wizard. Again note you are not deploying this.

Click **CLOSE** as this application already exists for you

Scaling The vks-01 Application



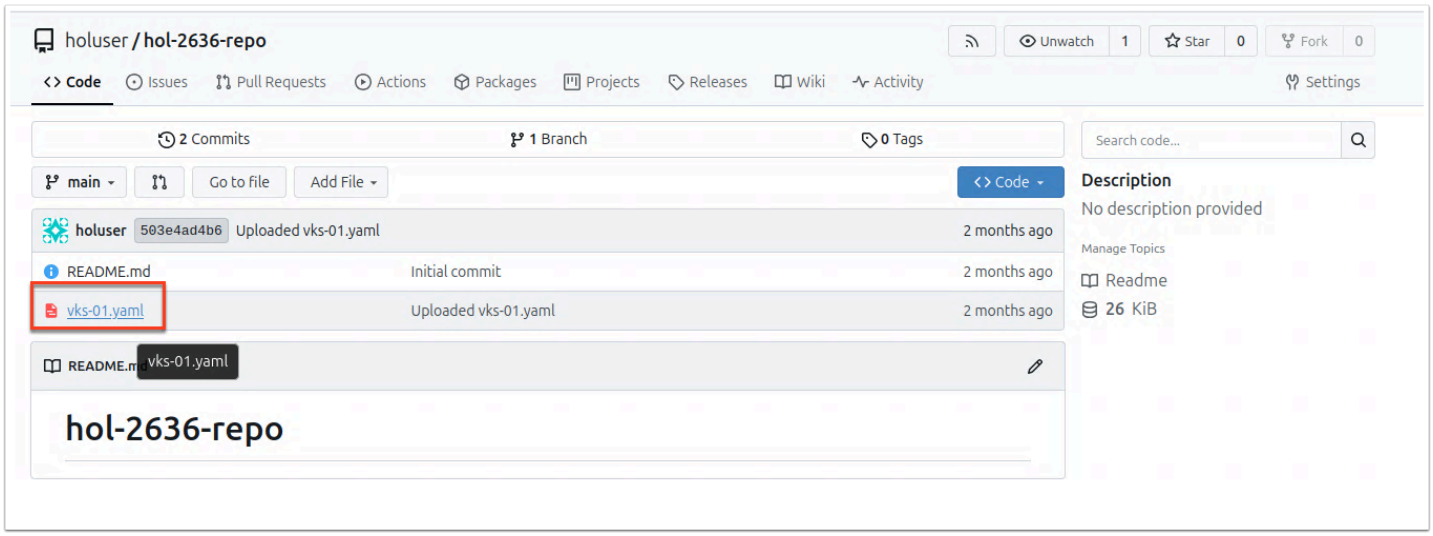
One of the largest benefits of containerized applications deployed via pipeline is the ability to rapidly iterate and scale. In the following steps we will walk through an example of scaling the **vks-01** application. In practice you would likely use an autoscaler but for informational purposes we will walk through the manual steps.

For example you determine that you need to deploy a second instance of the **vks-01** application.

First let's look at the **vks-01** YAML file which we used to deploy the application using ArgoCD.

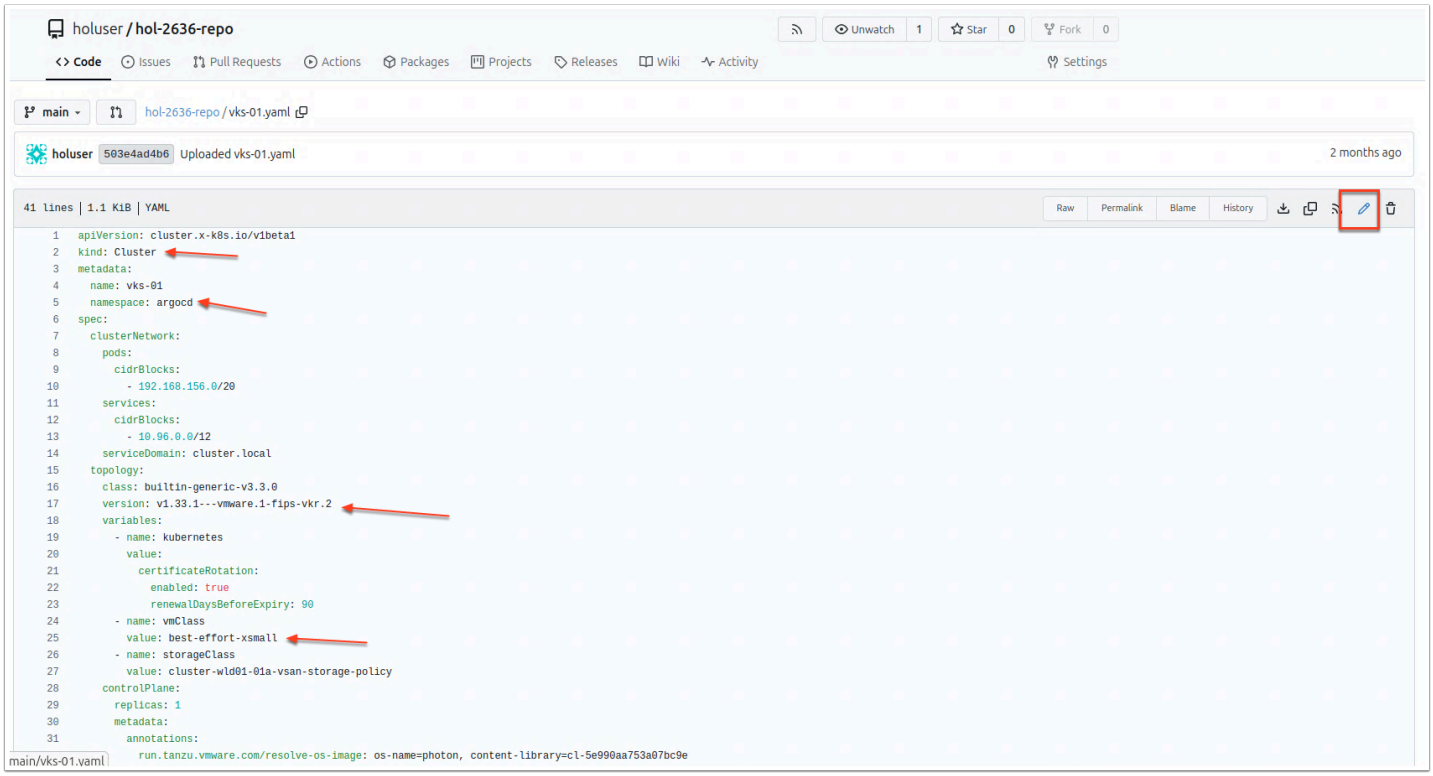
If you have closed the tab for gitea, open a new tab and log back in

Scaling The vks-01 Application (Contd.)



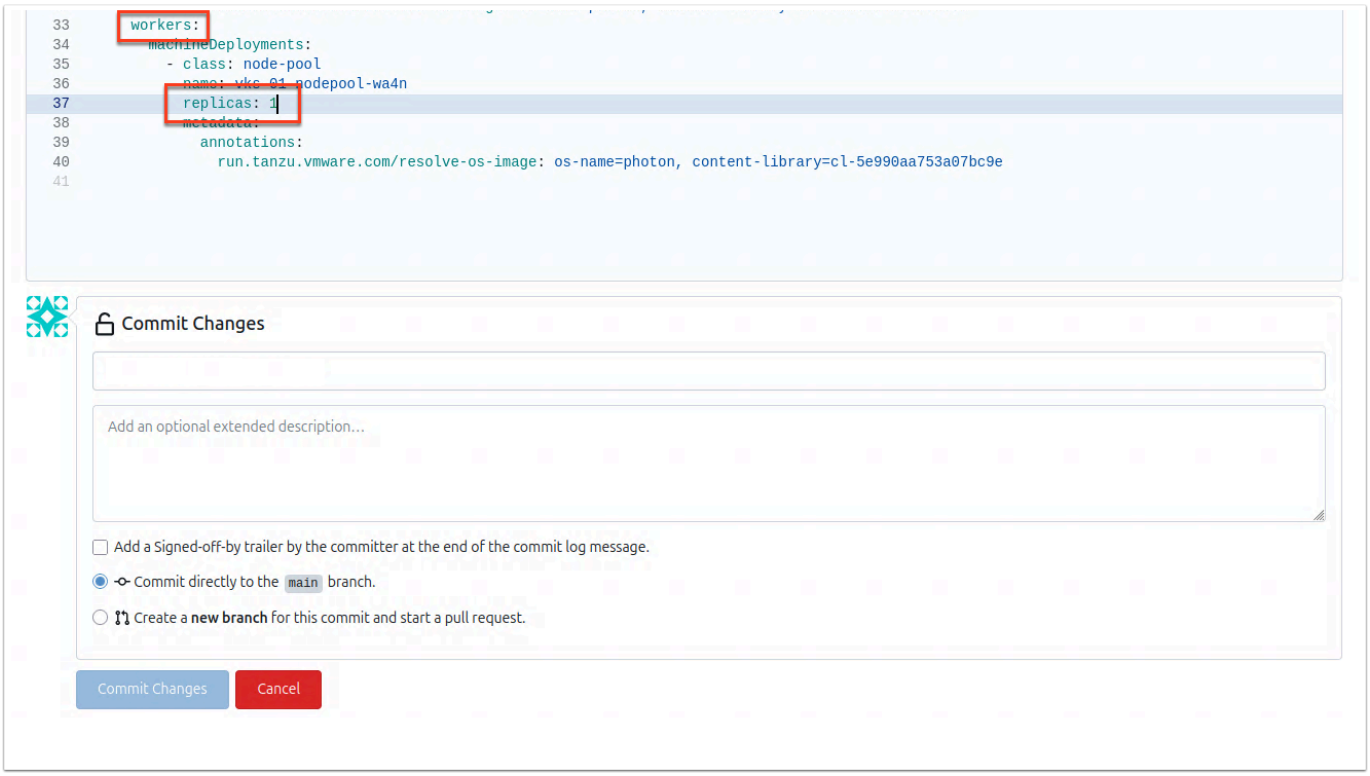
In gitea, in the previously opened repo ***hol-2636-repo*** there is deployment file for the application named ***vks-01.*** This app could just as likely be called something like ***account-update***. Maybe it is a containerized portion of a larger application for handling customer orders in a Web Store that sells T-Shirts. Maybe using microservices development methodologies.

In this case it is VKS CLuster deployment file. So it represents an infrastructure deployment. This file could be just one part of a much larger deployment, where tand entire test stack is deployed. Everything from the infrastructure, consisting of VKS (Kubernetes), Traditional Virtual Machines, Applications, Databases, test suites, etc. are deployed, logged and torn down. Any single part of this scenario and more can be completely automated with VCF and ArgoCD. This "application" ***vks-01*** is a simple example of how this level of Continuous Delivery can be handled.



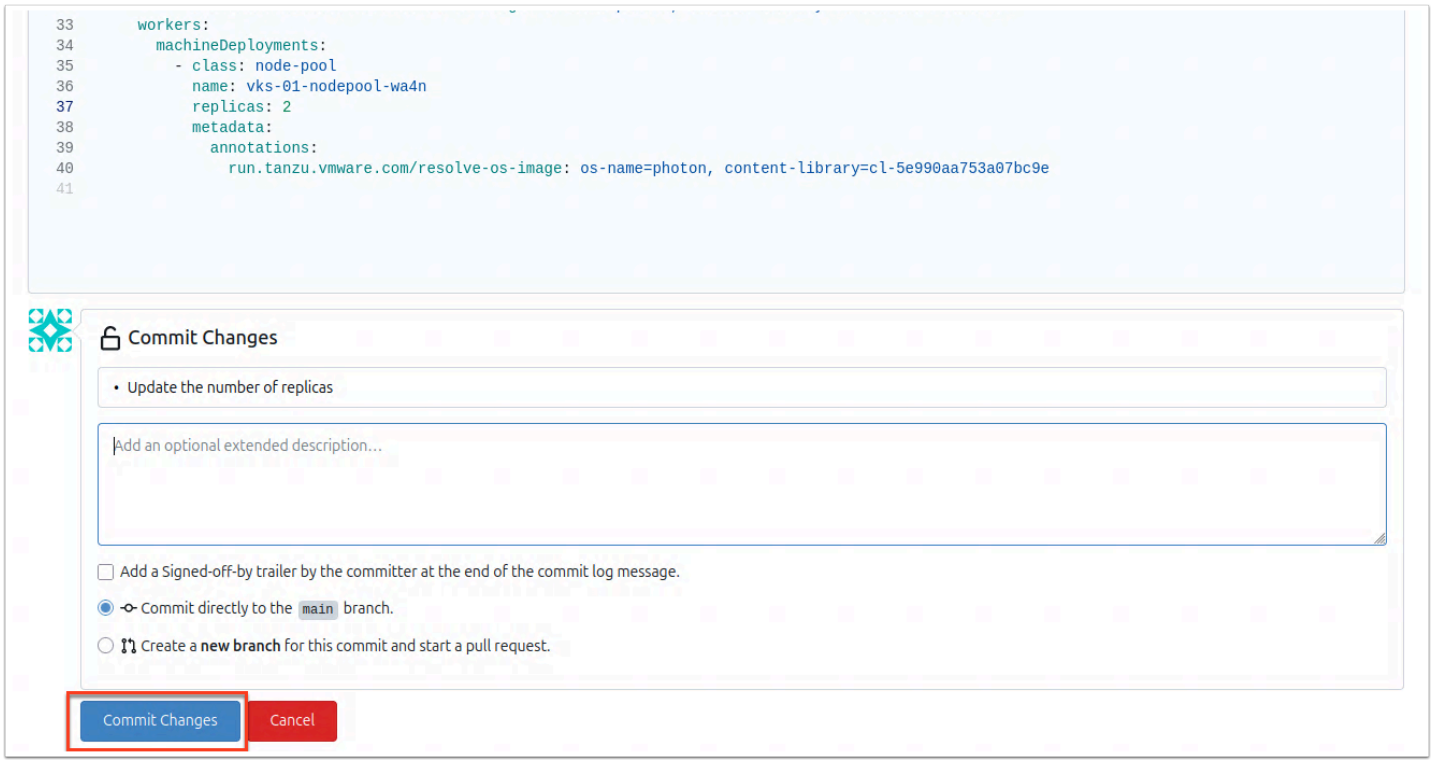
1. Click on the **pencil** icon to edit the **vks-01** yaml.

Scaling The vks-01 Application (Contd.)



1. Scroll down to the bottom of the ***vks-01*** yaml file until you see ***workers:***
2. On the ***replicas*** line change the ***1*** to a ***2***

Essentially as you might guess, we are going to deploy a new ***vks-01*** **VKS** cluster to the ***argocd*** **vSphere Namespace**. A simple example of "*declarative end state configuration*". The desired state is a 2 worker node cluster in the namespace ***argocd***. We **declare** this desire to the platform and it does it for us. More importantly, it will do it's best to keep the system constantly baed on the stated desired end state. We will demonstrate how the system responds when an "out of band" change is made a bit later.

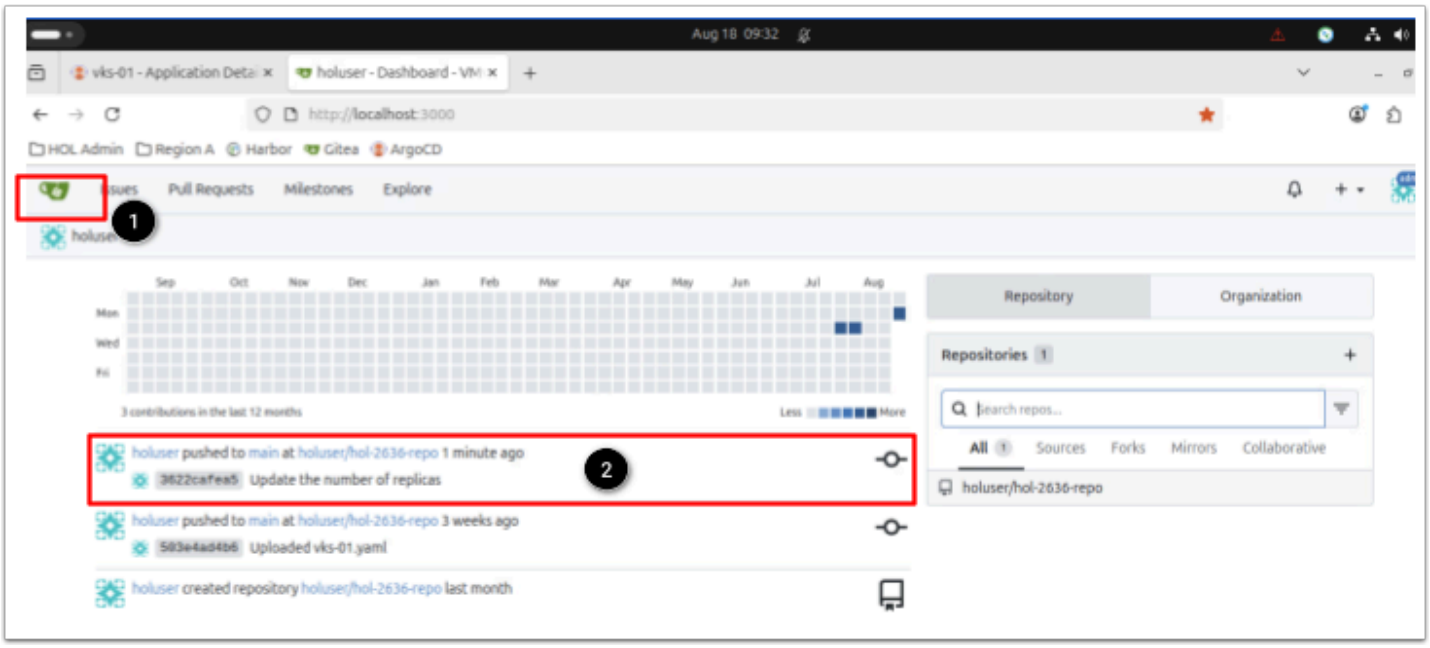


After changing the 1 to a 2 scroll down again.

- Under Commit Changes type something in the top text box like “Update the number of replicas”
- Hit “Commit changes’

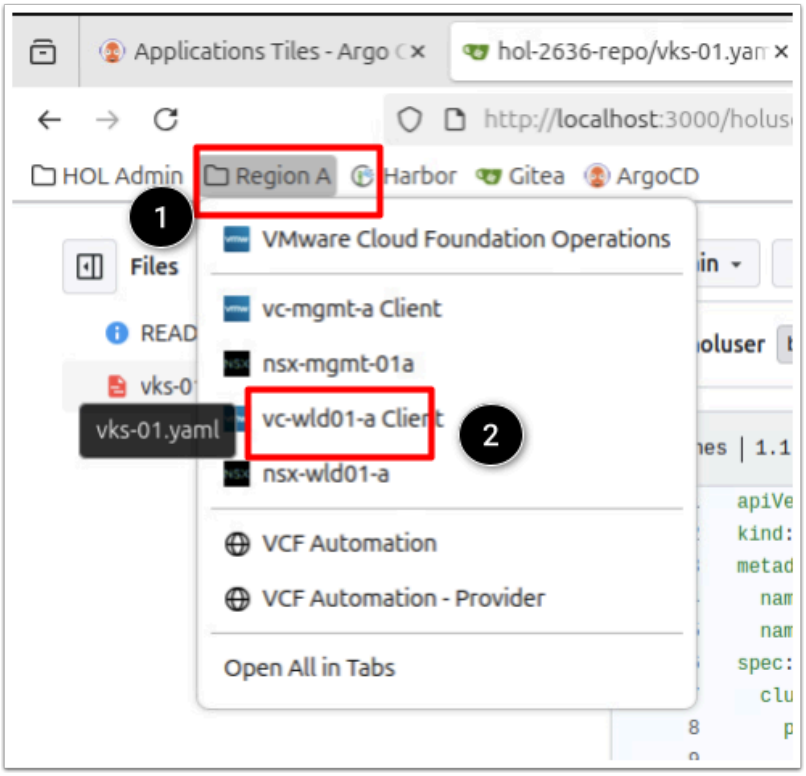
> Normally this would be done from a tool like VSCode, of some external system. For demo purposes it works fine.

Scaling The vks-01 Application (Contd.)



1. The dashboard in gitea will show that the file was updated.
2. Return to the gitea Home Page To Review Changes by clicking the coffee cup.
3. You will see that the change you just made has been reflected here.

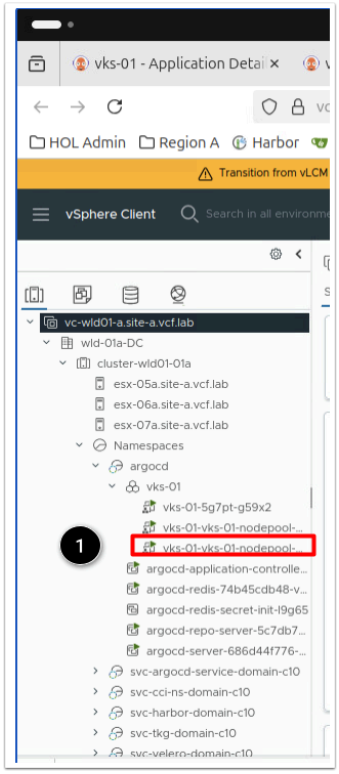
Return To vCenter



If you do not have the vCenter browser tab open:

1. Under the **Region A** tab
1. Select **vc-wld01-a Client** to open vCenter

Now ArgoCD is continuously synching with the gitea repo and will see the change to the vcf-01 file and upload it and re-run it. Which in turn results in another replica of the vcf-01 pod being deployed. If we open vCenter and look at the ArgoCD tab we'll notice that there is a new replica deployed.



You could scale vks-01 up or down using the same method.

For Example if you want to **play** a bit, wait for the vks cluster to finish deploying and in gitea change replica to 1 or 3. Don't go too crazy, as this is a lab with limited resources and you might just break something.

Module 6 - Using ArgoCD to Manage VMs

Module 6 - Using ArgoCD to Manage VMs

Declarative vs Imperative

A declarative end-state configuration is a model for system management that focuses on describing the **desired state** of a system, rather than the step-by-step instructions needed to achieve it. A "reconciliation engine" or "controller," is then responsible for analyzing the current state of the system and performing the necessary actions to converge it toward the declared end state. This approach is in direct contrast to an imperative configuration, which relies on a series of commands to detail ***how-to*** change the system state.

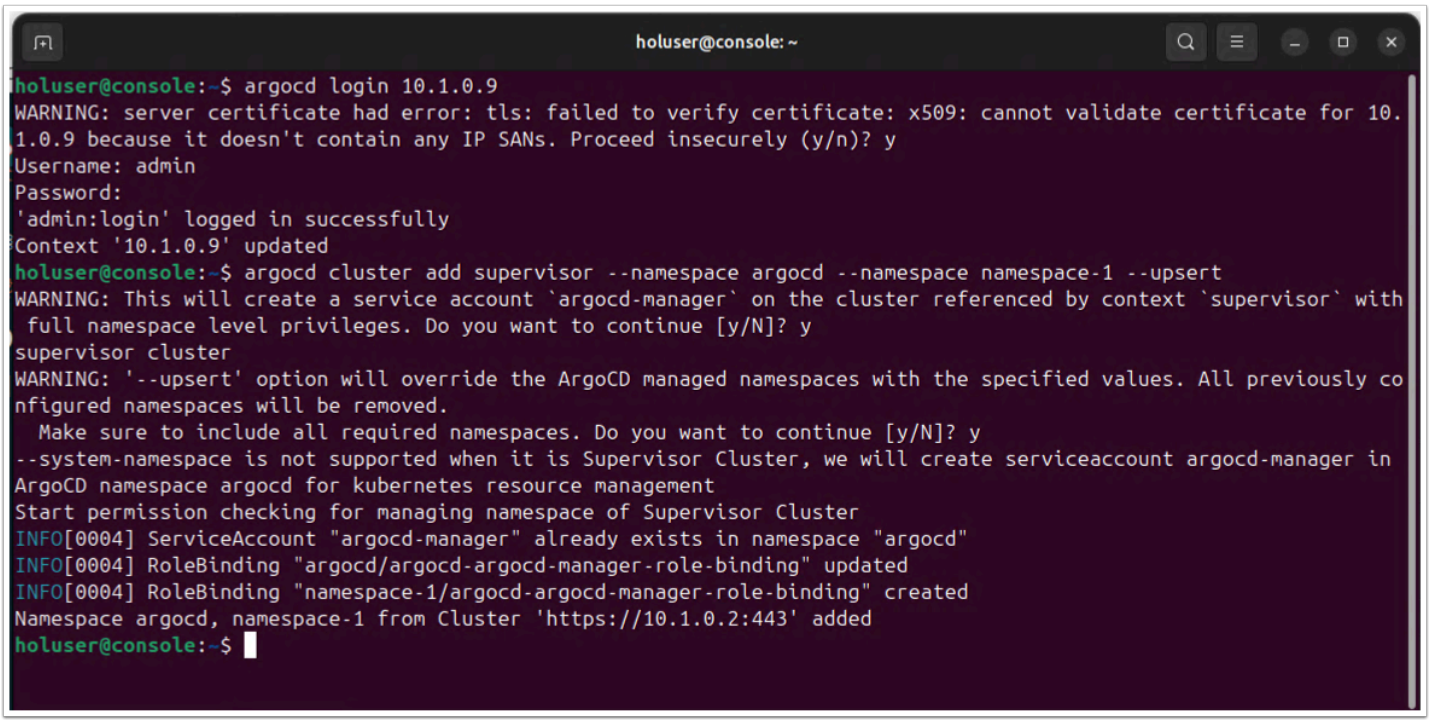
In this module we are going to step away from Kubernetes, microservices and modern apps, but continue to work with ArgoCD. You will use the VCF Consumption Platform to define and manage the ***desired end-state*** of, shall we say, a more Traditional Virtual Machine. As in the previous modules you will use the Local Consumption interface in vCenter to create the declarative ***yaml*** for a VM. You will then create a new repository the local git engine GitEA. You configure a new Application in ArgoCD. This new setup in GitEA and ArgoCD representing the tools gitOps teams will use to do their work. You will then use ArgoCD's continuous delivery in combonation with the **VM Service** of the **vSphere Supervisor** to demonstrate how they work in cooperation to create components of the "reconciliation engine" or "controller"

Lastly you will see how this platform will work to enforce this "desired endstate" of the VM even when external efforts try to create undesired change of state.

Add Namespace-1 to ArgoCD Cluster Config

Before you can move on, you need to add the **vSphere Namespace** `namespace-1` to the ArgoCD Cluster Settings. ArgoCD was deployed and configured for you as part of the lab, but it was only configured to use a the **vSphere Namespace** it was deployed to. Your shiny new **vSphere Namespace** needs to be added before ArgoCD can manage deployments in it.

The steps are quite simple.



As the ArgoCD admin you need to log in via the argocd cli. From a terminal window log into ArgoCD using the ArgoCD CLI

```
argocd login 10.1.0.9
```

Click to copy

Username: and **Password**:

```
admin - VMware123!
```

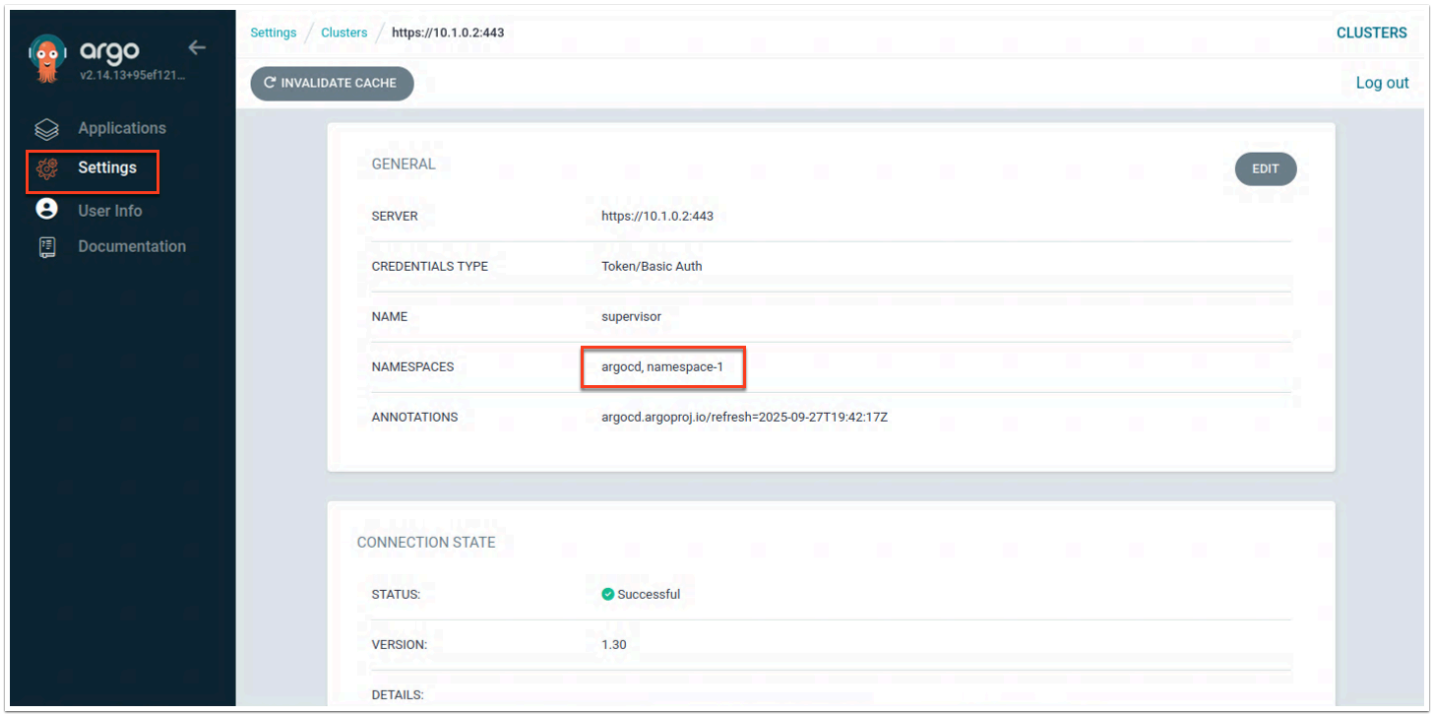
Click to copy

Then you run the command that will add, or in this case add the new **vSphere Namespace** to the **Cluster configuration** in the ArgoCD engine. The switch at the - `-upsert` is what tells ArgoCD to add not replace.

```
argocd cluster add supervisor --namespace argocd --namespace namespace-1 --upsert
```

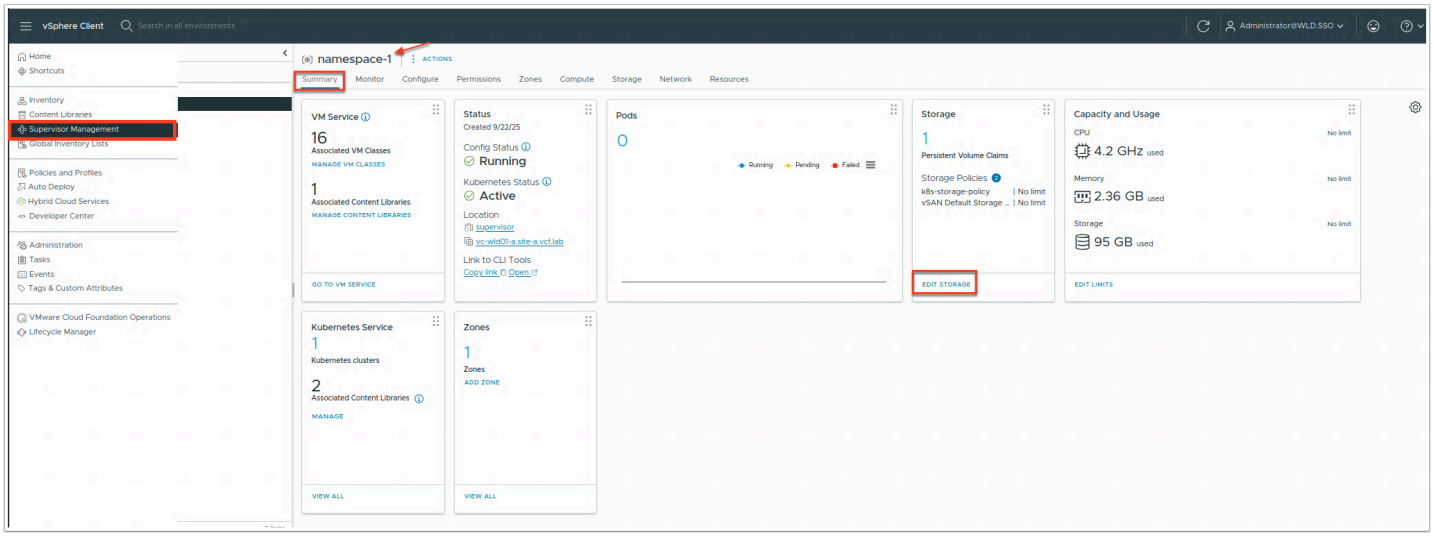
Click to copy

Answer **y** twice when you get asked. Once this is done, you can verfiy the change back in the ArgoCD UI --> Settings --> Clusters



Add a new Storage policy to vSphere Namespace

OK, very sorry about the bit of house cleaning efforts, but it needed to be done. **Back to our regularly scheduled programming.**

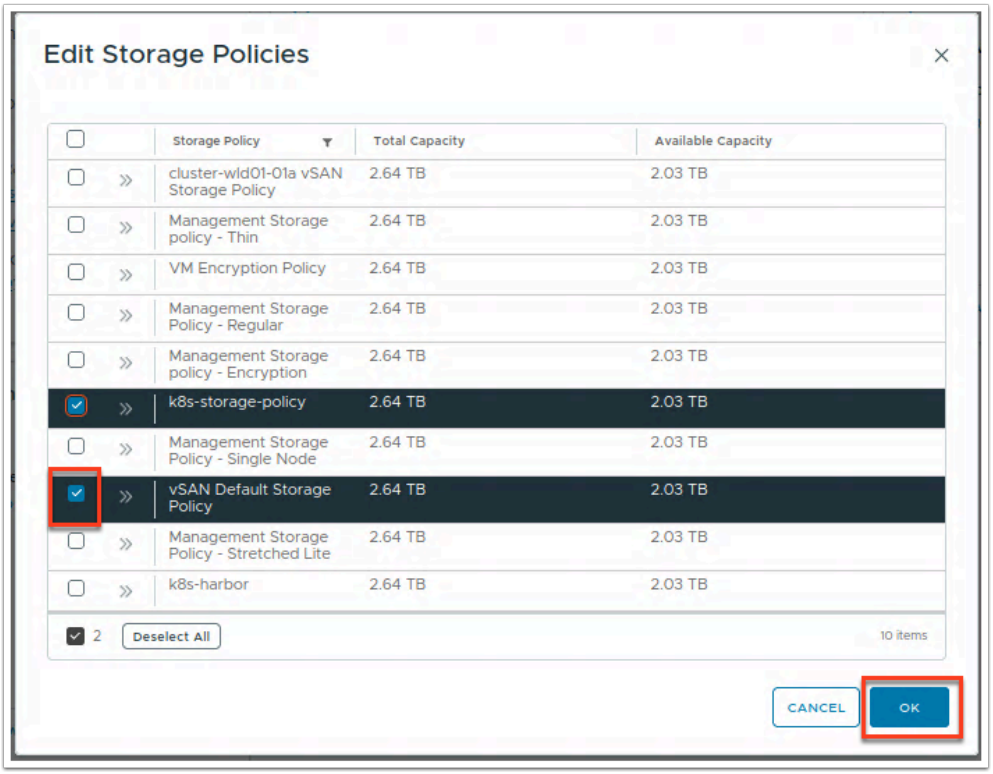


This module's focus is on how you can bridge the management of traditional **vSphere Virtual Machines** with the management techniques used by the dev-ops or git-ops teams as they work on new "cloud native" applications, like those that have been refactored to microservices and run in containers. Wth that in mind the following steps should be very familiar.

1. Go back to the **vCenter client** for ***wld01-a***
2. Click **Supervisor Management**
3. Click **namespace-1**

4. Click the **Summary** tab
5. Click **EDIT STORAGE** in the **Storage** tile

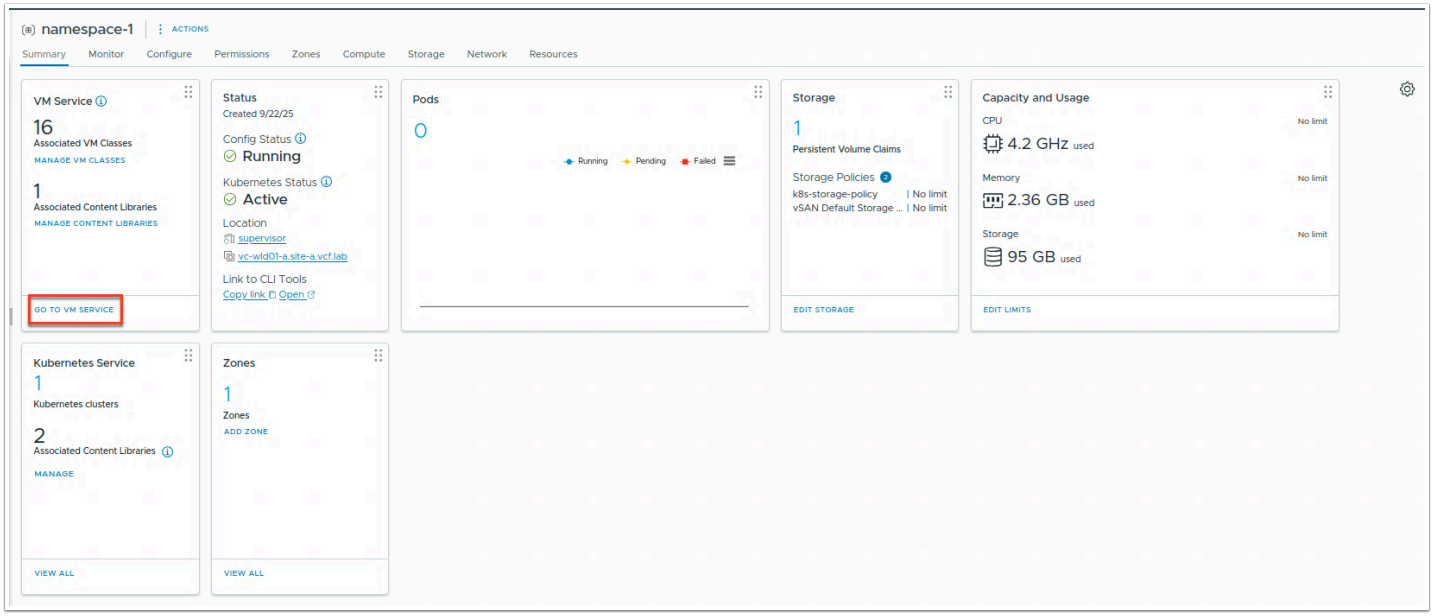
Add VM storage Policy



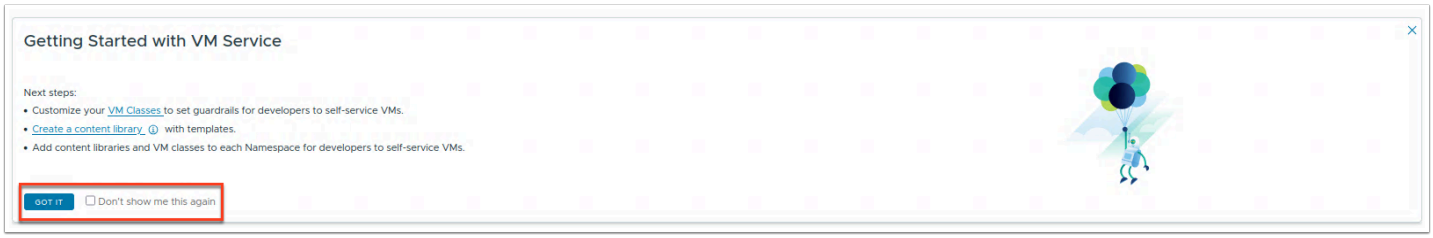
In this example we will use the existing ***vSan Default Storage Policy*** to represent the desired storage policy for the Virtual machine. Most likely the storage handling needs for a VM will be different from the storage needs of an application running in containers.

1. Click the **Radio Check Box** for ***vSan Deafault Storage Policy***
2. Leave the **Radio Check Box** for **k8s-storage-policy** checked
3. Click **OK**.

Close message in VM Service UI

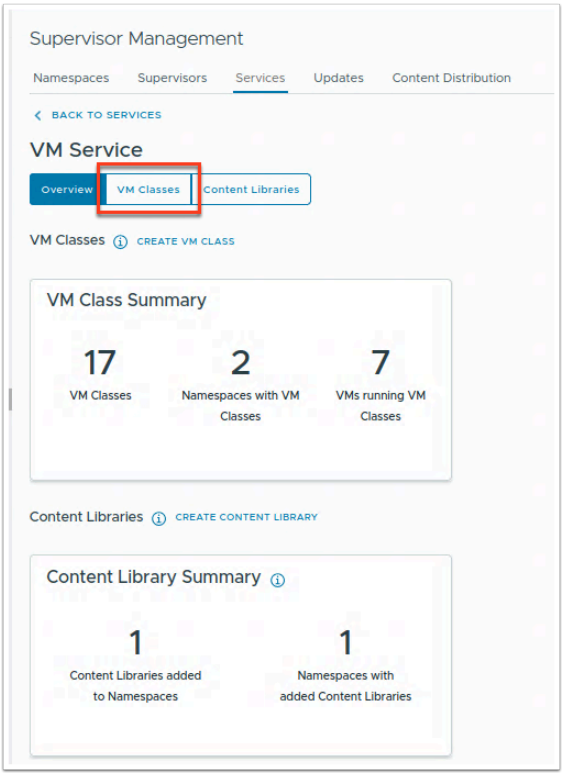


1. Click **GO TO VM SERVICE** in the **VM SERVICE** tile



1. Close the **Getting Started with VM Service** tile if is open by check the box for **Don't show me this again** and clicking **GOT IT**

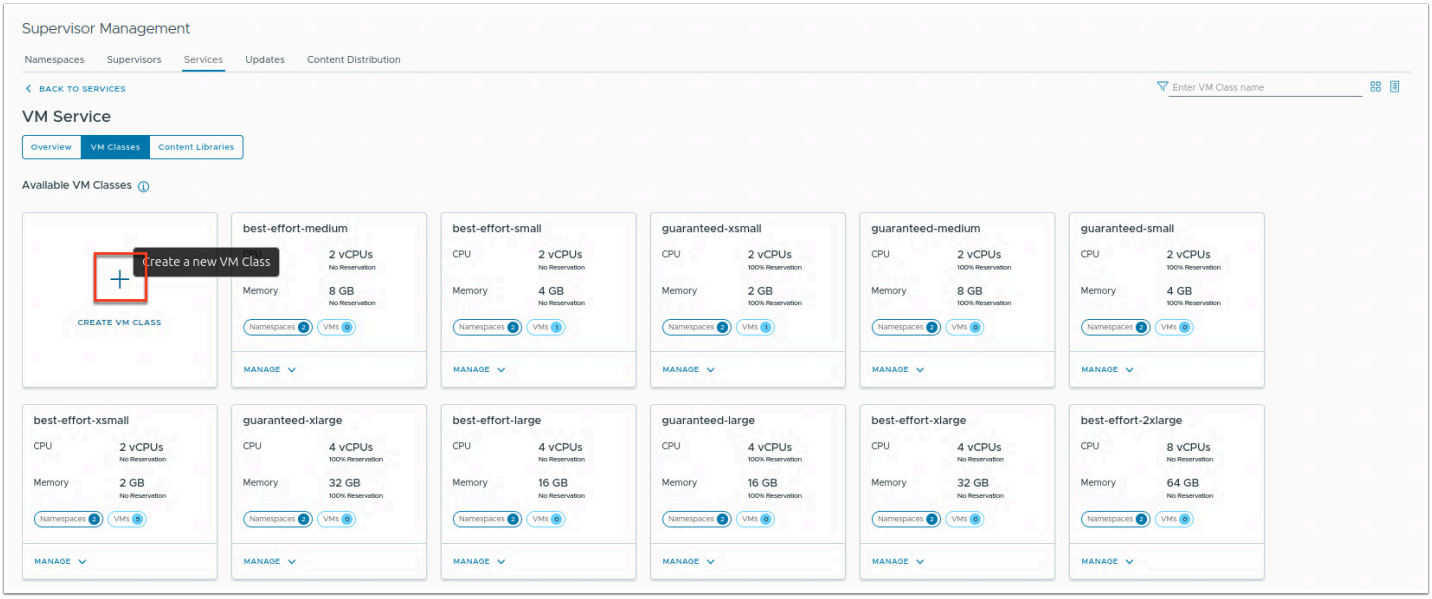
Review VM Service Requirements



We can see here we already have 16 VM Classes defined the VM Services. This was done in earlier efforts when you added them to the VKS Service.

1. Click **VM Classes** tab

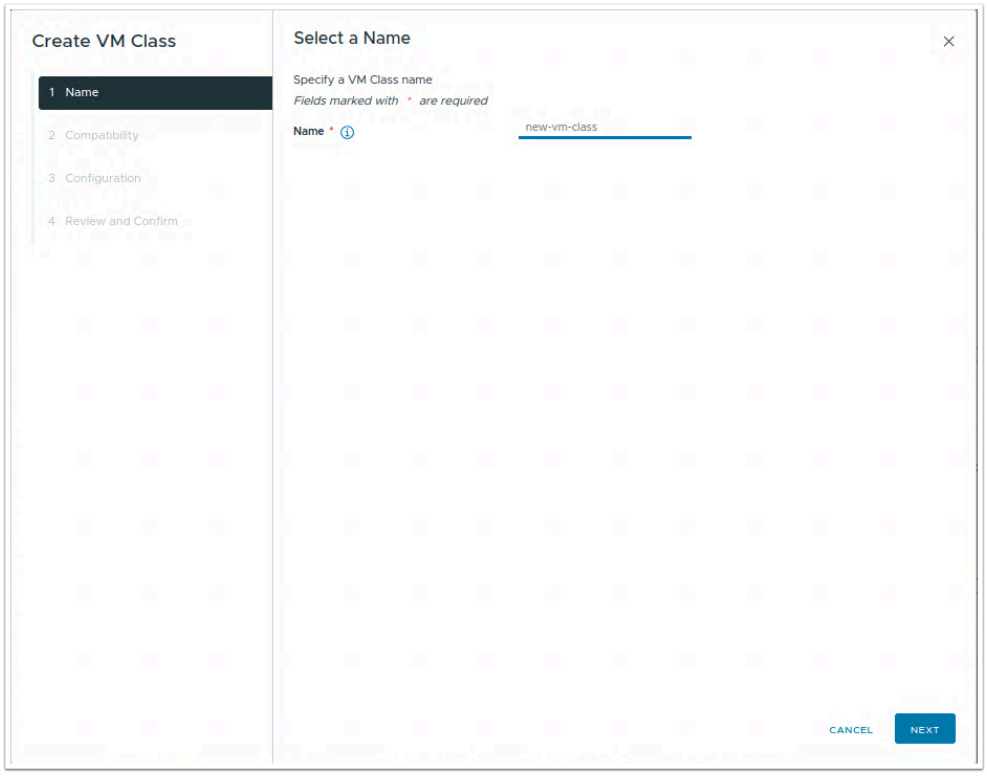
Continue Review



Here you can see the **VM Classes** that were added earlier. You can also create new classes from here.

1. Click the **+** above **CREATE VM CLASS**

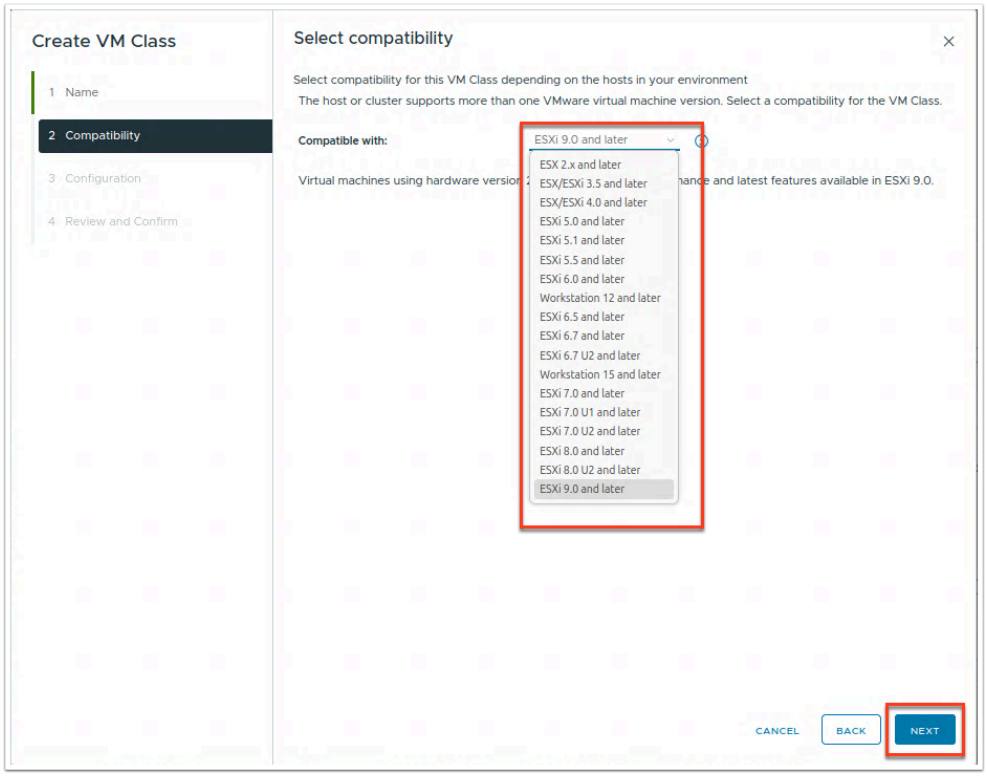
Create New VM Class - Name



You give your new **VM Class** a name

1. Click **NEXT**

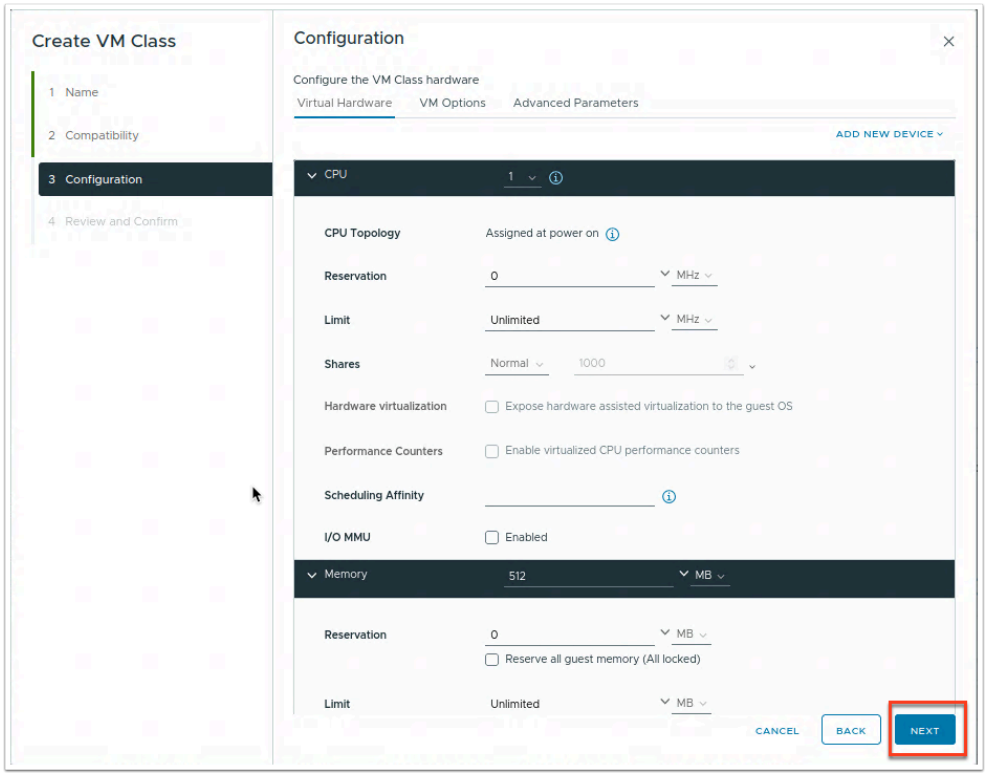
Create New VM Class - Compatibility



Choose the Hardware Version to use

1. Click **NEXT**

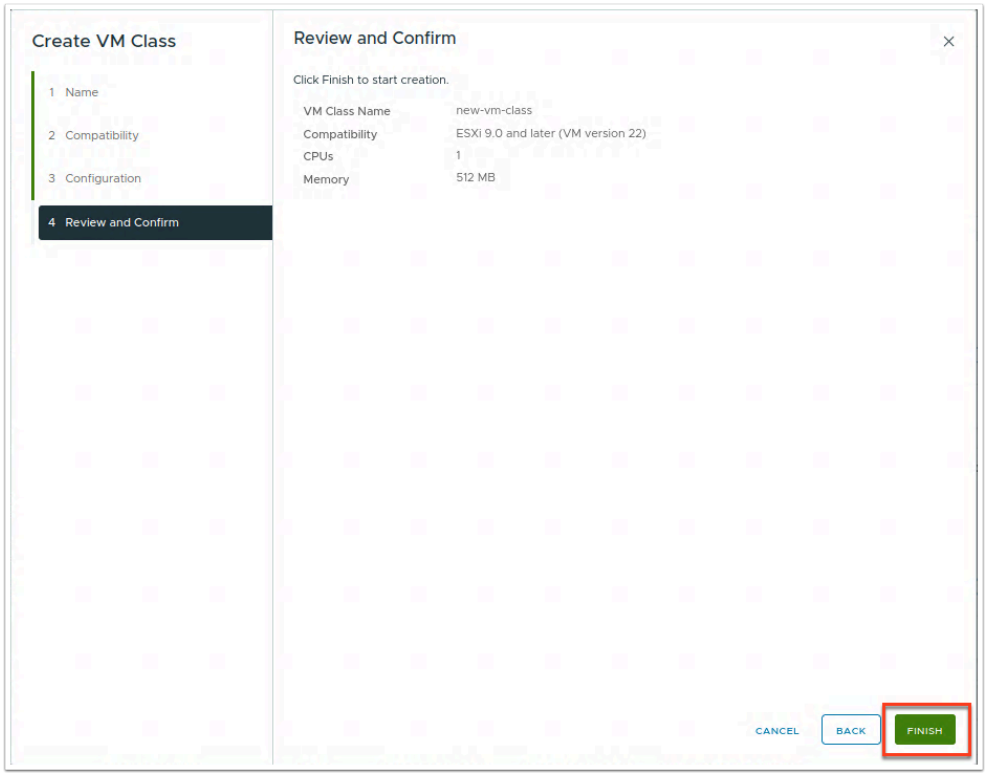
Create New VM Class - Configuration



Essentially here you have all that you expect to see under **VM Settings** in **vCenter**. You can click through the tabs to verify what you expect to see.

1. Click **NEXT**

Create New VM Class - Review and Confirm

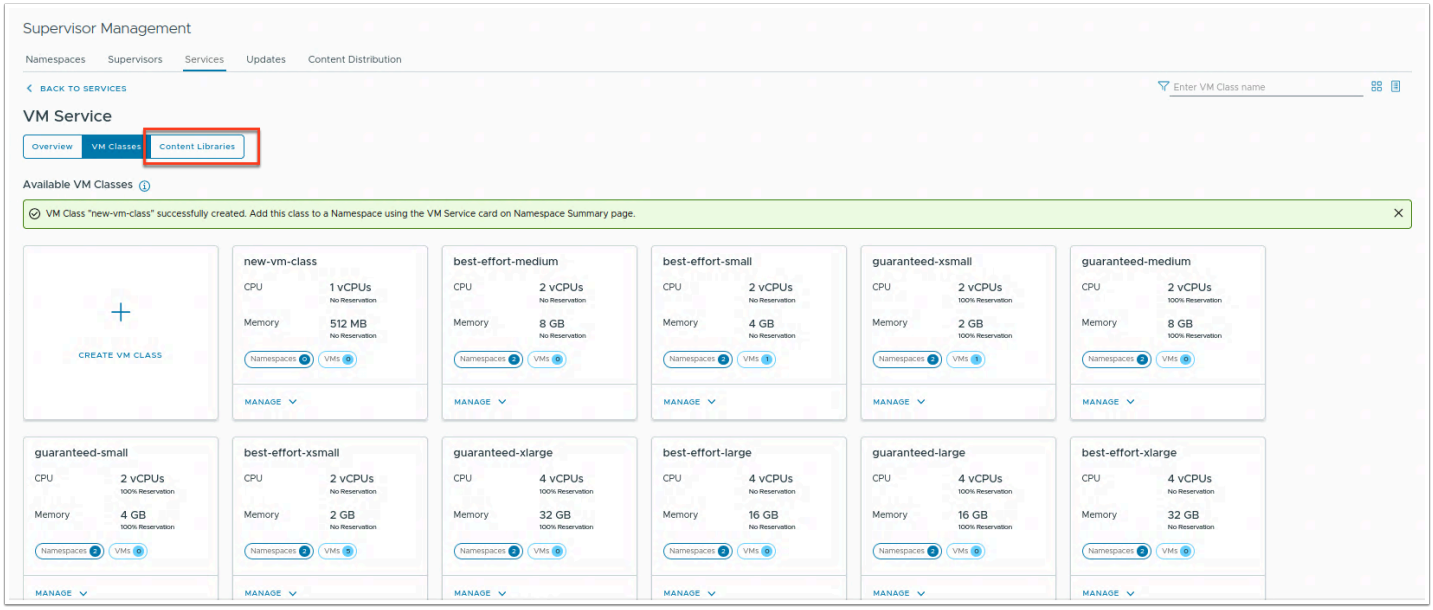


Review and confirm your settings

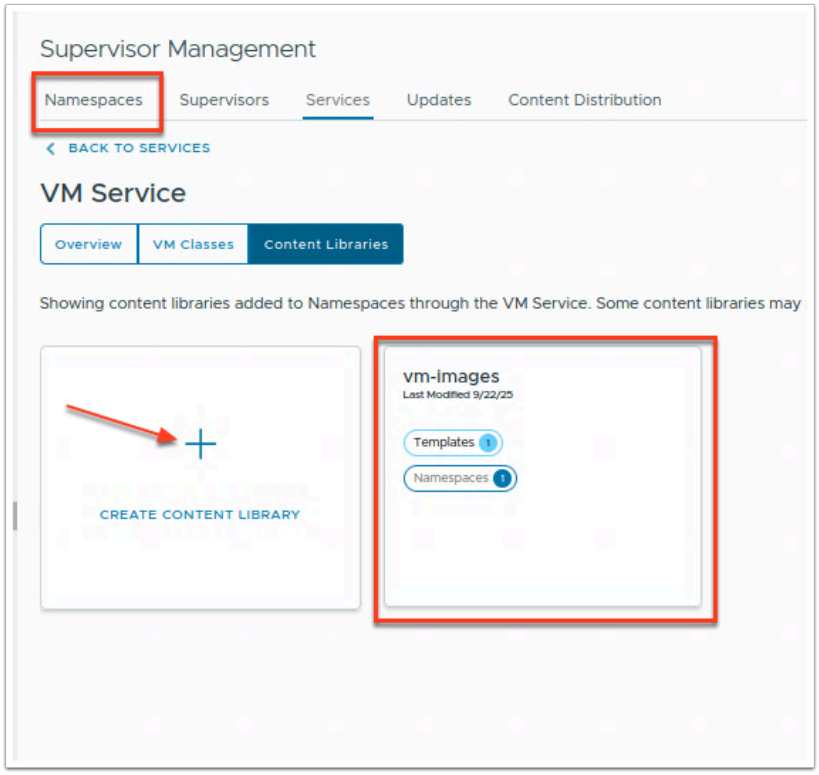
1. Click **FINISH**

> We are going to use an existing **VM Class** so it is entirely up to you as to whether save this definition or not. Maybe use it do learn what can be defined and consider the possibilities.

Content Libraries



1. Click **CONTENT LIBRARIES**

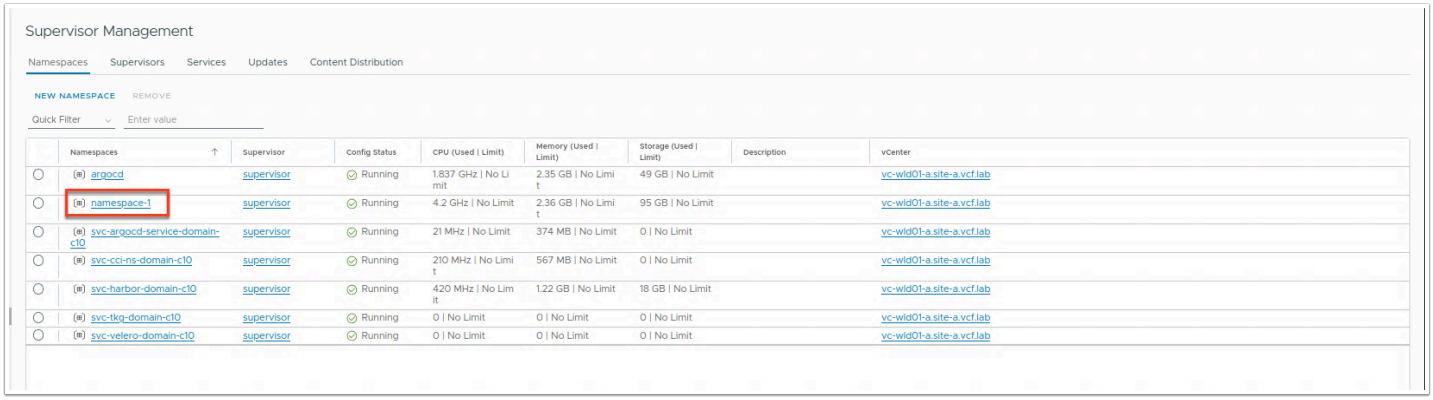


This **VM Service** already has a **Content Library** assigned to it from **Module 2** so you will not need to add one. However if you needed to create one you can start it from here by clicking the **+** sign in the **CREATE CONTENT LIBRARY** tile.

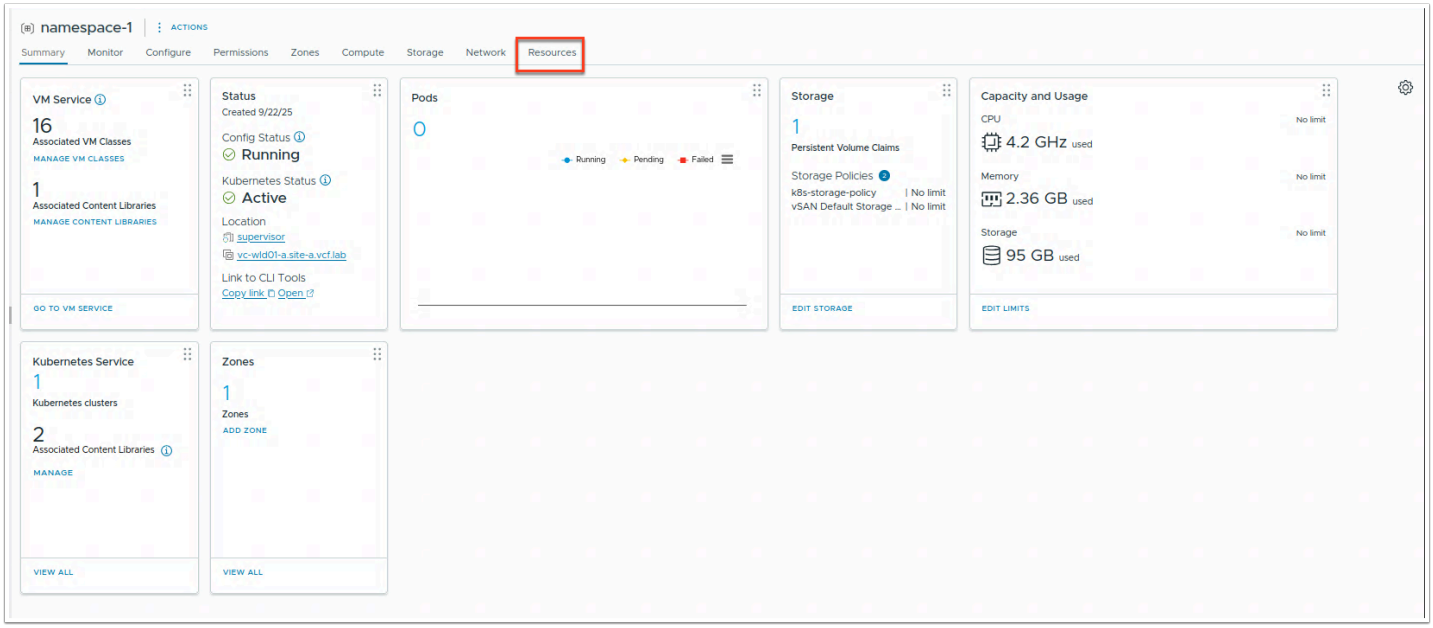
Since you have what you need:

2. Click the **Namespaces** tab

Back to namespace-1 - Resources

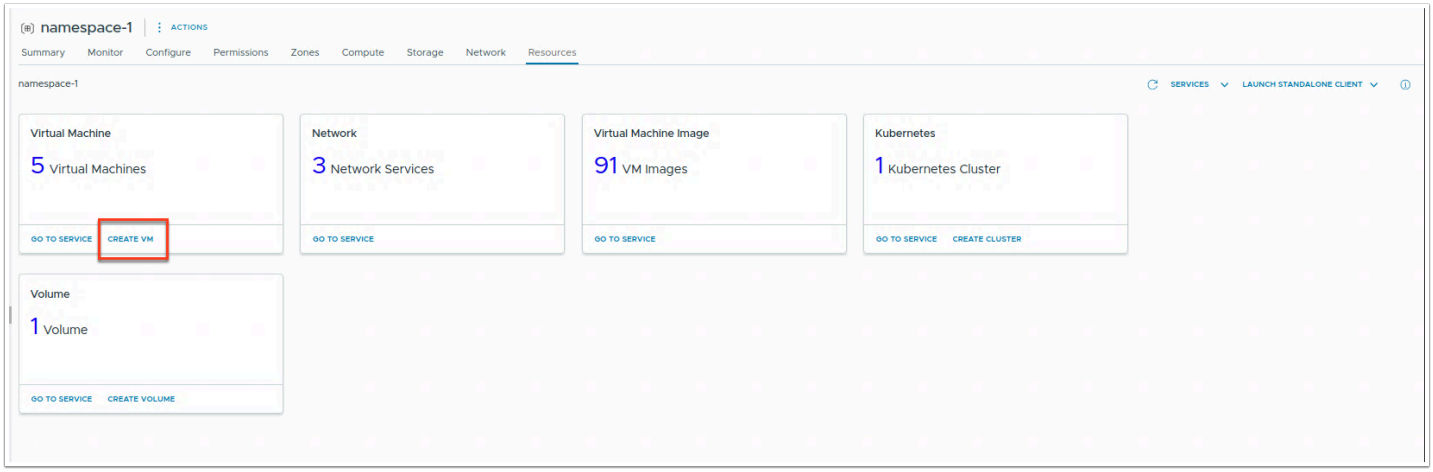


1. Click **namespace-1**



1. Click **Resources** tab

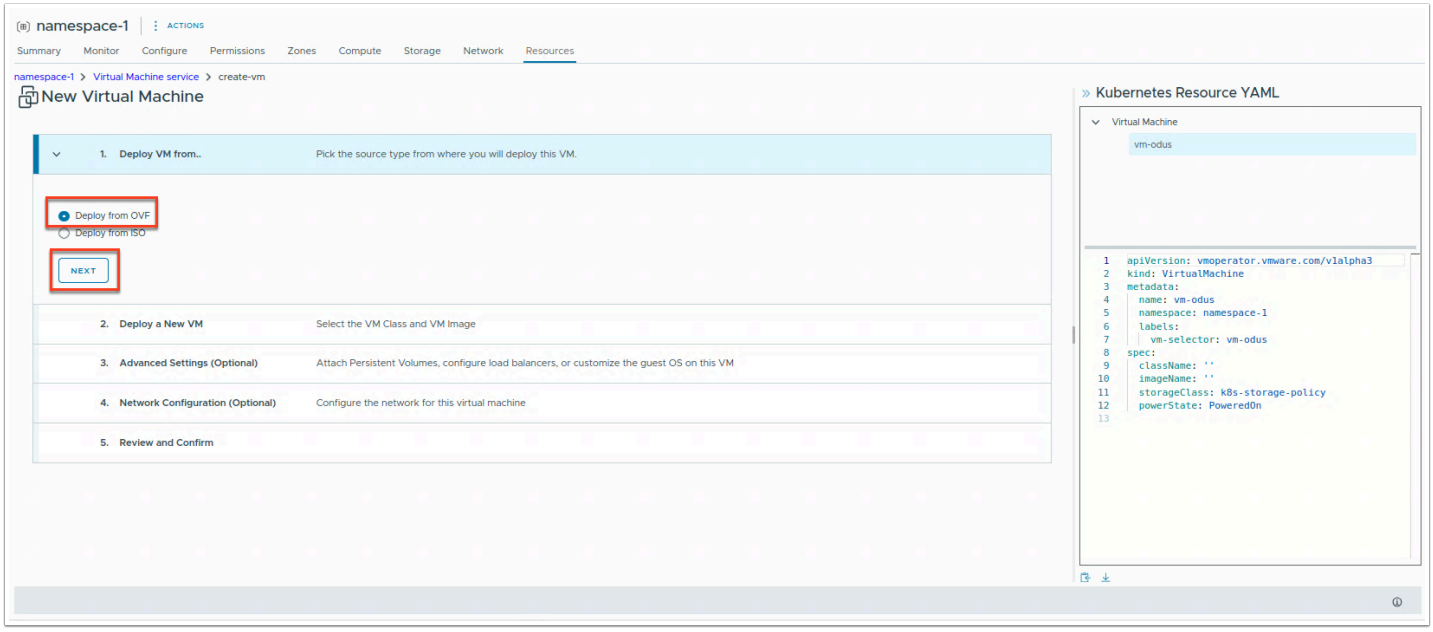
Create a basic Virtual Machine declaratively



Back in Local Consumption Interface the goal is to create a Virtual Machine deployment yaml

1. Click **CREATE VM**

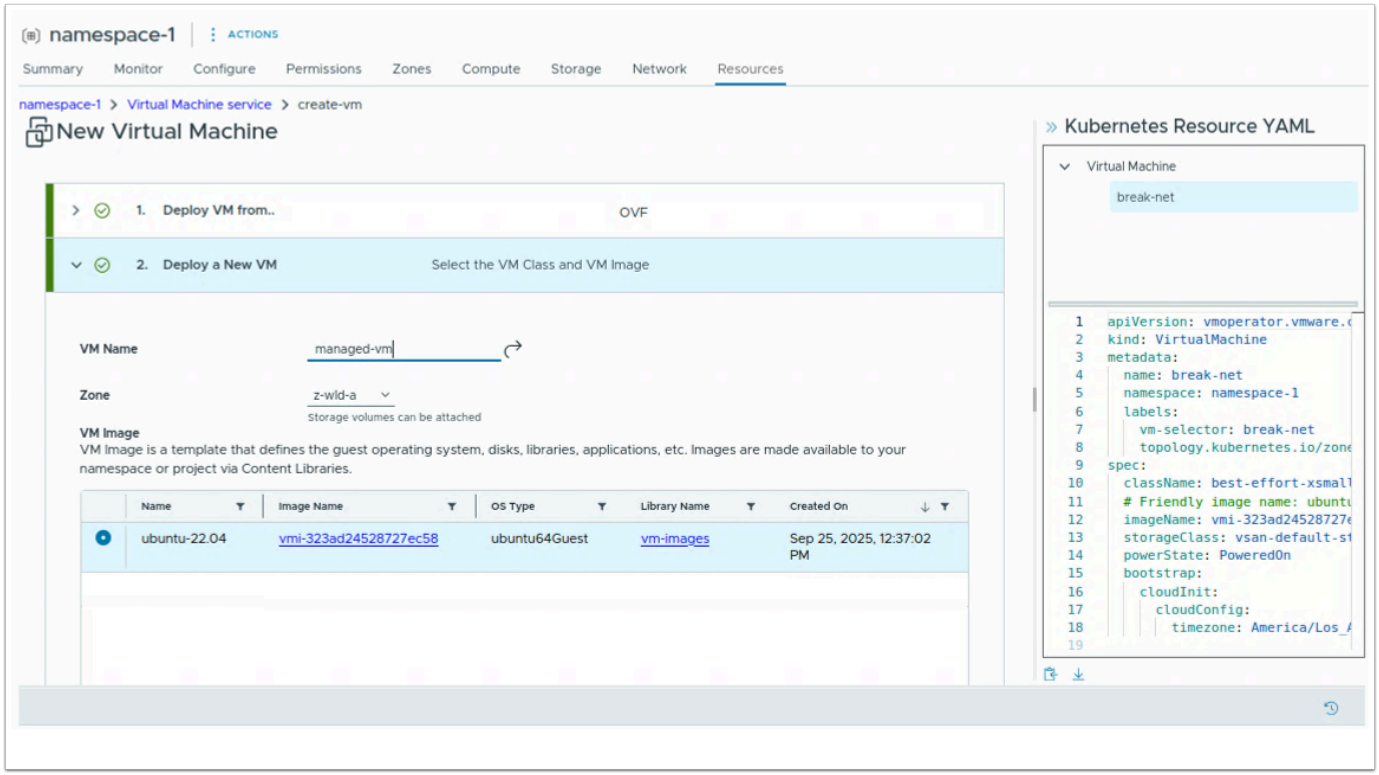
New Virtual Machine



Once again we see in the LCI the UI Workflow to creating a VM on the left two thirds and the yaml being created dynamically on the right.

1. Leave **Deploy VM from OVF** selected
2. Click **NEXT**

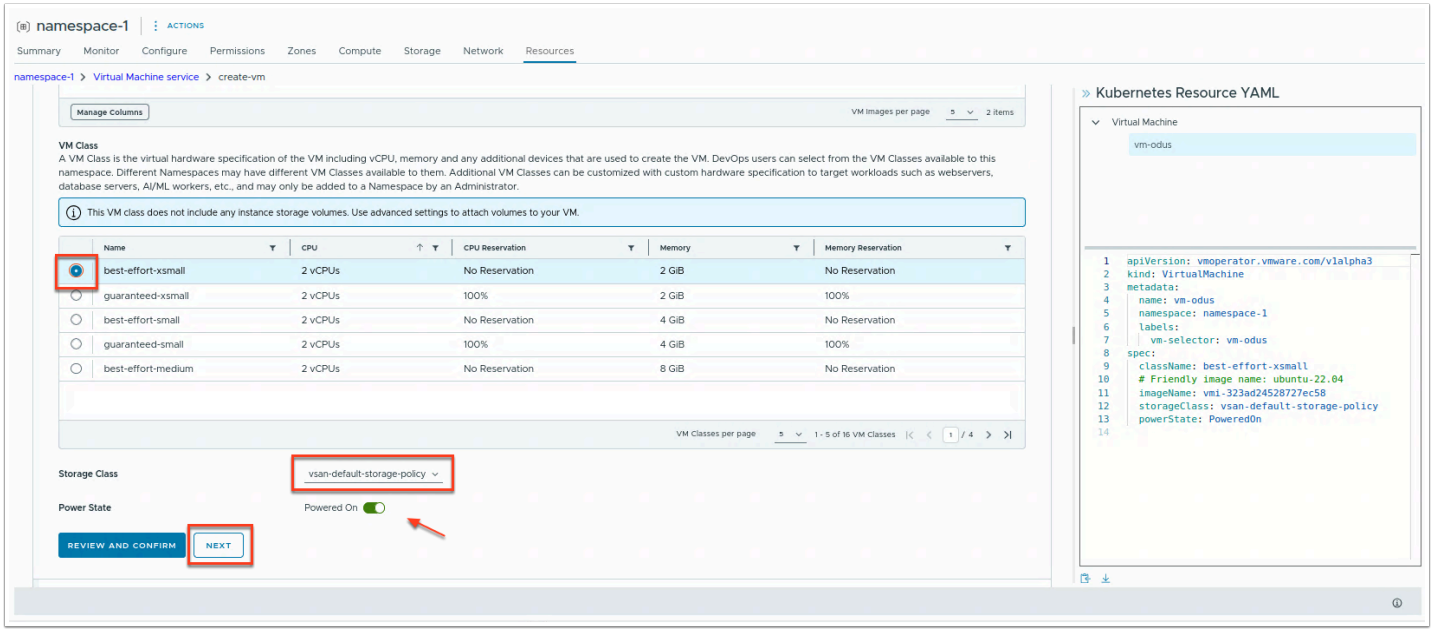
Deploy a VM



1. Set the **VM Name** as `managed-vm`
2. Click the dropdown for Z**one:** select `z-wld-a`
3. Select the `ubuntu-22.04` VM Image

> All the settings, and more, that are being set now can be made into variables by the pipeline project at deploy time. We are simply creating the basic yaml deployment file.

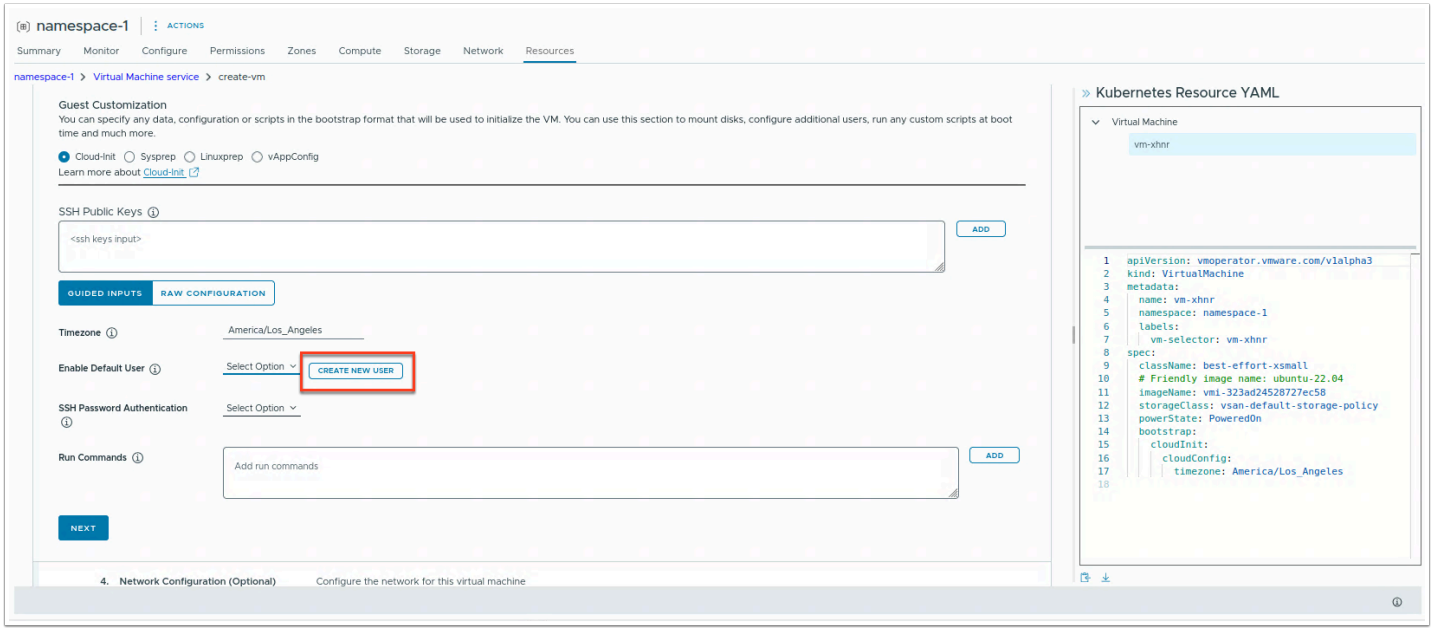
Deploy a VM (contd)



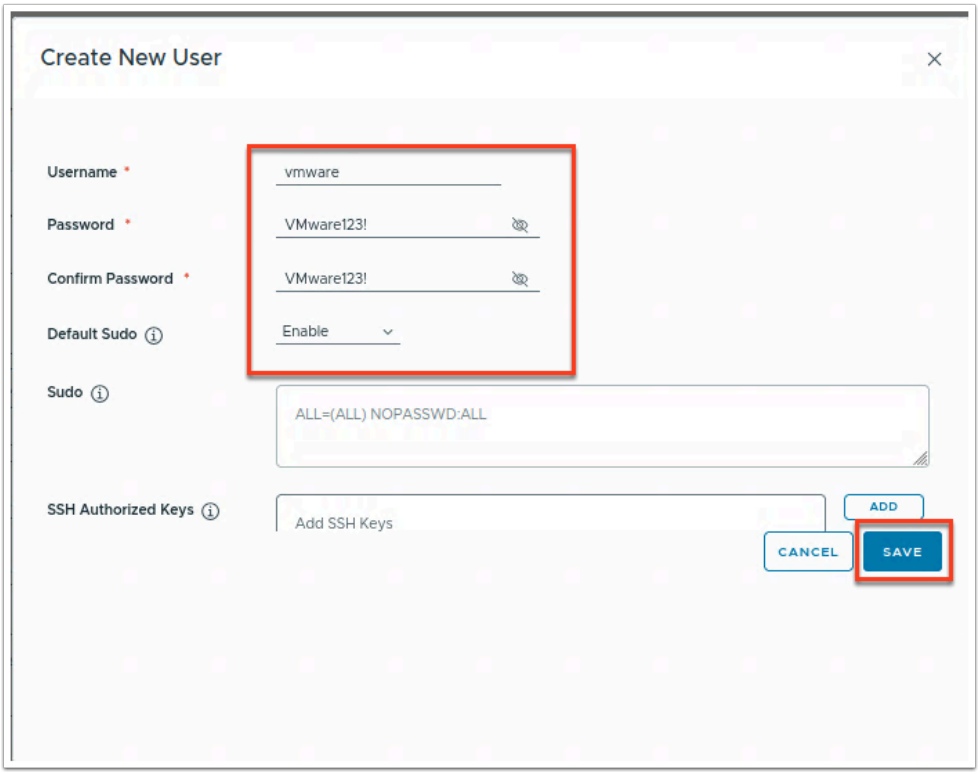
Scroll down in the workflow window to **VM Class**

1. Select the `best-effort-xsmall` **VM Class.**
2. Select the `vsan-default-storage-policy` from **Storage Class**
3. Leave the slider as `Powered On` for **Power State**
4. Click **NEXT**

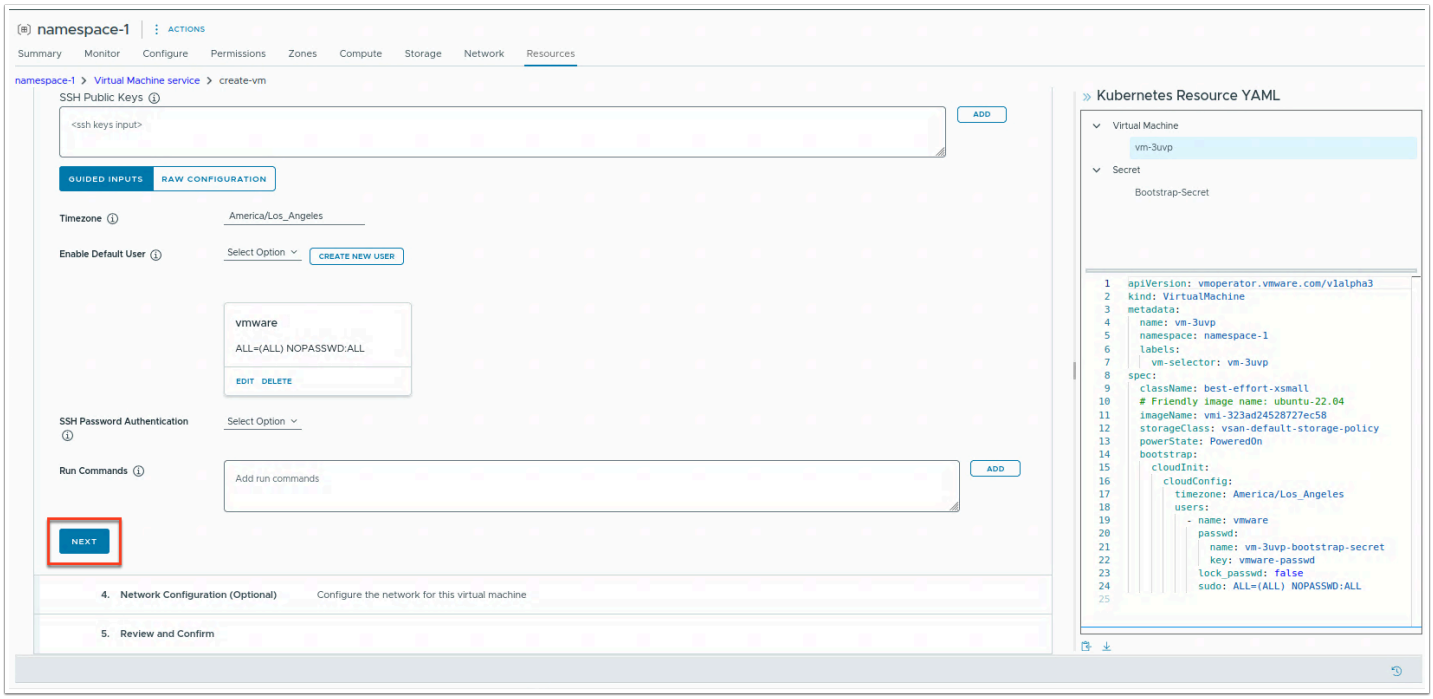
Advanced Setting (optional)



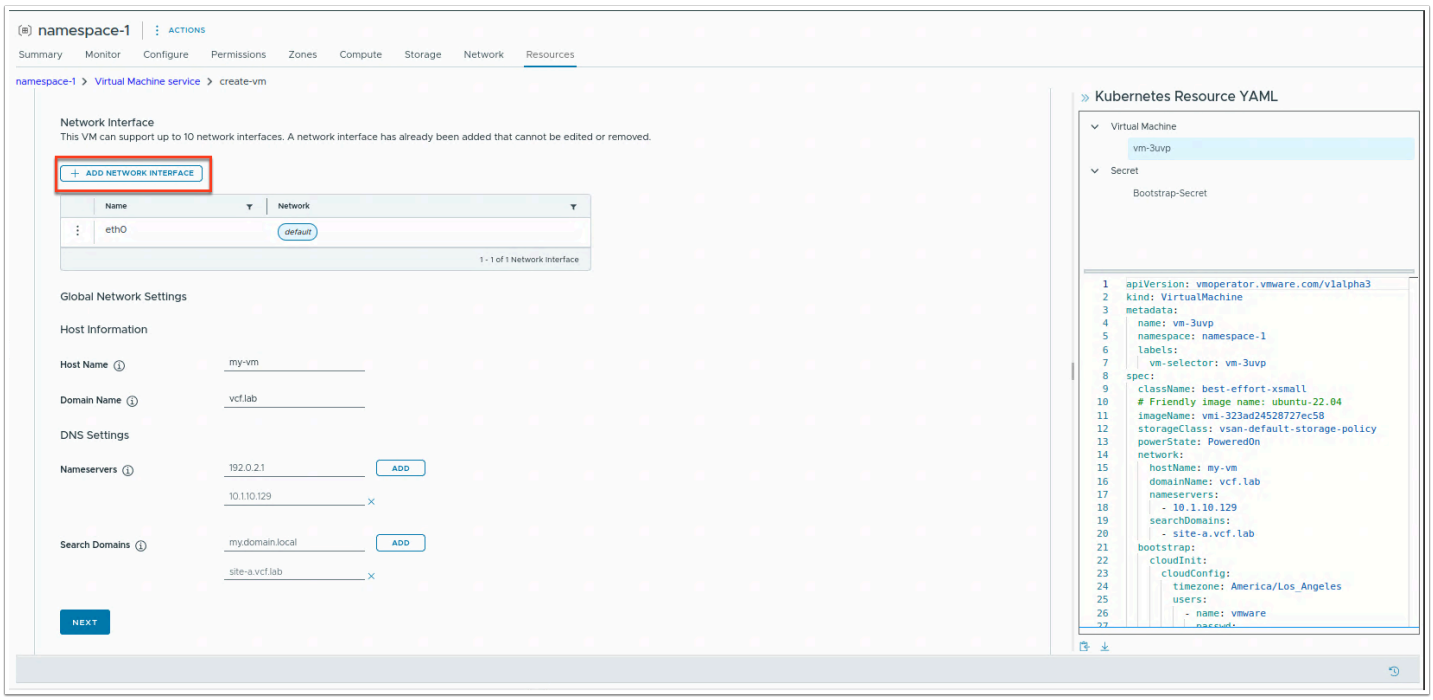
1. Make sure the **Cloud-Init** Radio button is selected
2. Enter **America/Los_Angeles**
3. Leave **Enable Default User** and **SSH Password Authentication** as they are
4. Click **CREATE NEW USER**

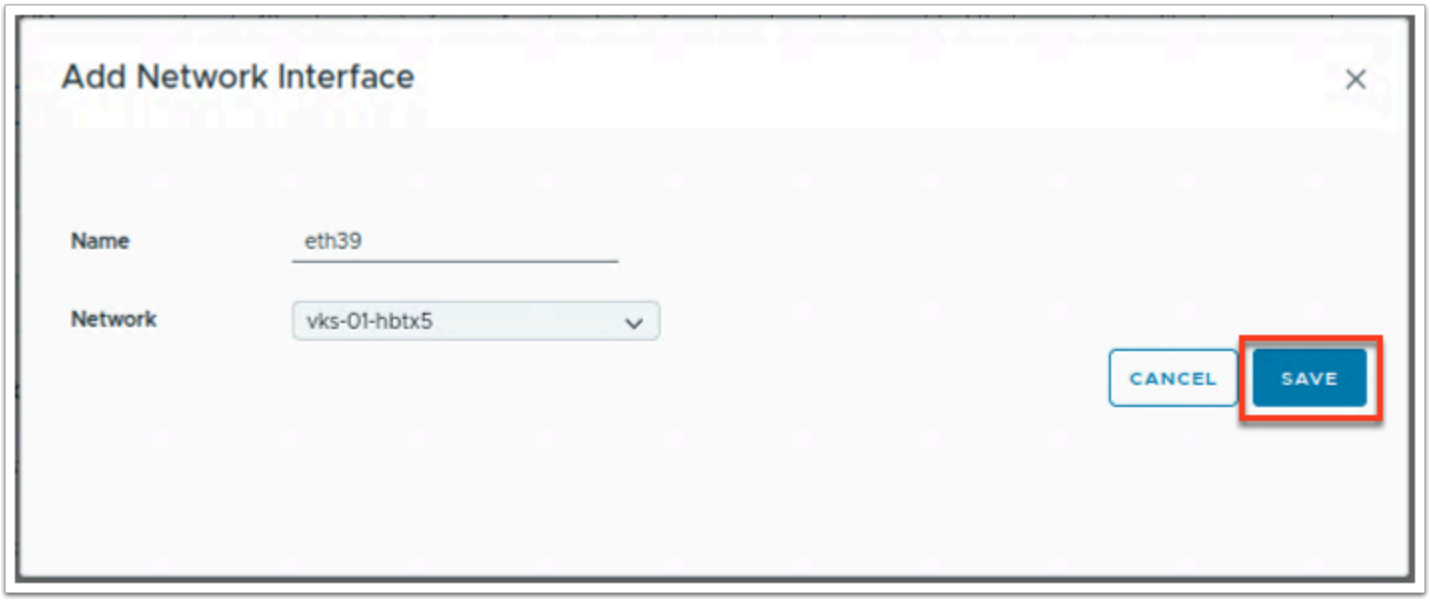


1. Set **Username:** `vmware`
2. Set **Password:** `VMware123!`
3. Set **Deafult Sudo:** `Enable`
4. Click **SAVE**



Network Configuration (Optional)

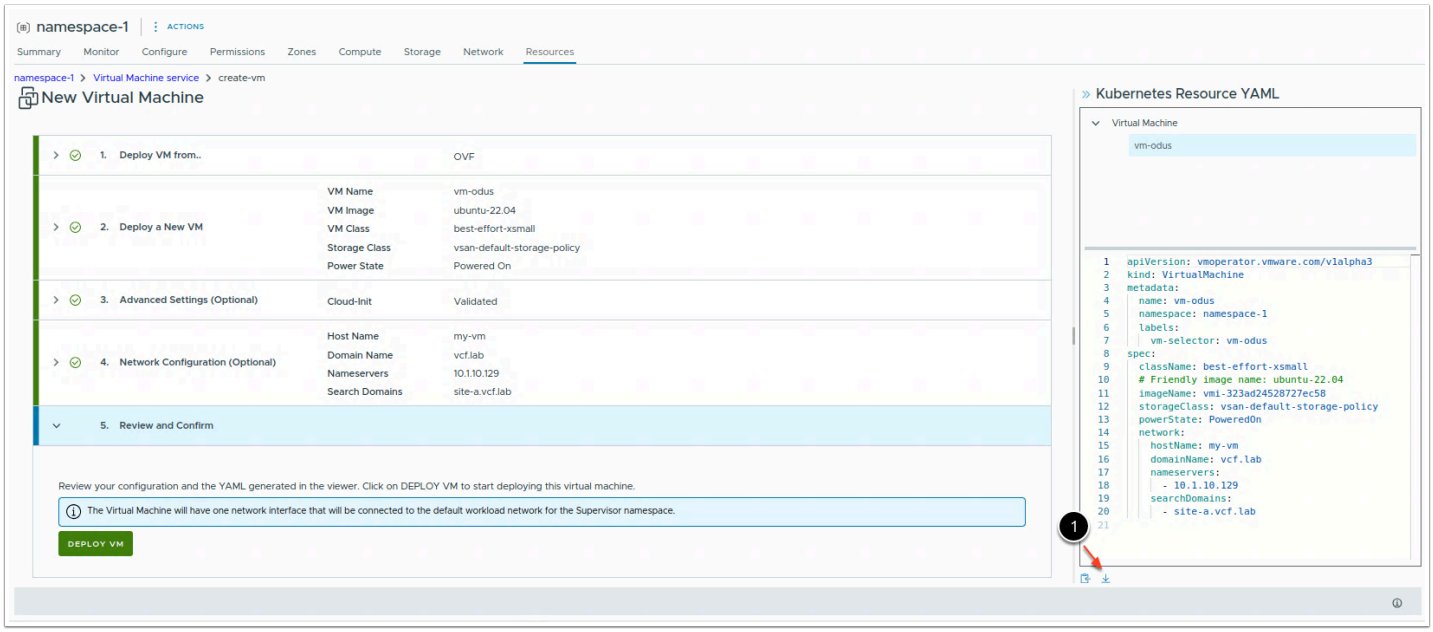




This lab does not have external internet access but we can do some simple network tests. Make the following configuration changes

1. Click **+ ADD NETWORK INTERFACE**
2. Take the defaults and click **SAVE**
3. For **HostName:** - you can name it anything you want.
4. For **Domain name:** `vcf.lab`
5. For **DNS Settings - > Nameservers:** enter `10.1.10.129` - click **ADD**
6. For **Search Domains:** enter `site-a.vcf.lab` - click **ADD**
7. Scroll down and click **NEXT**

Download Deployment yaml

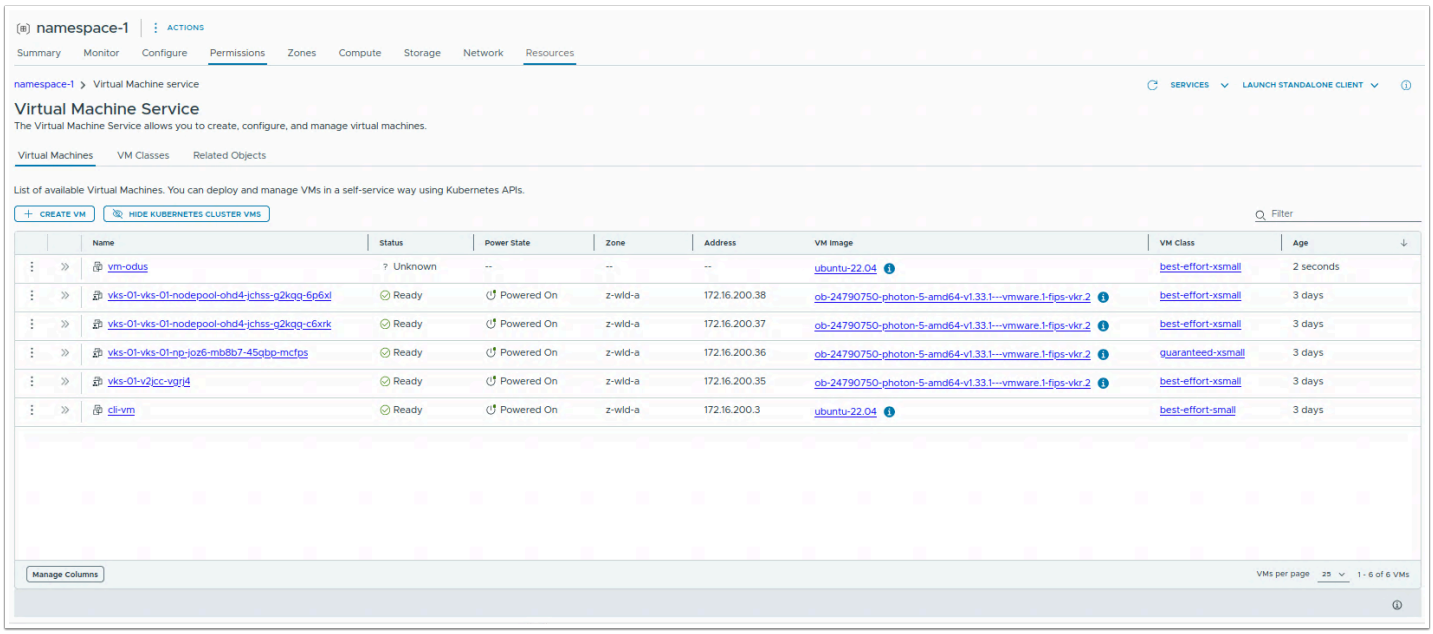


At this point you could click **DEPLOY VM**. First though click the **Download** botton for the YAML that has been created from the UI Workflow.

1. Click **Download All YAMls to zip**
2. Click **DEPLOY VM**

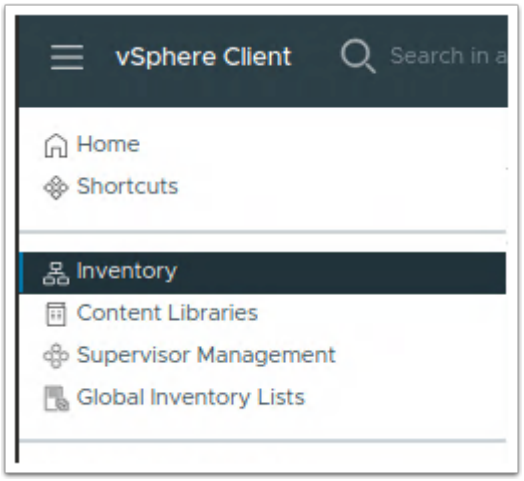
We will use this zip file later.

New VM Deploys



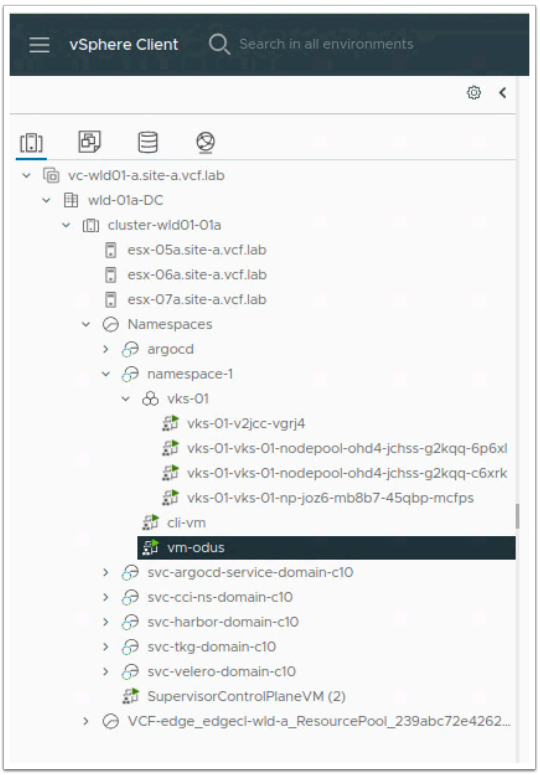
You should see your new VM show up in the Virtual Machines table.

vCenter Inventory



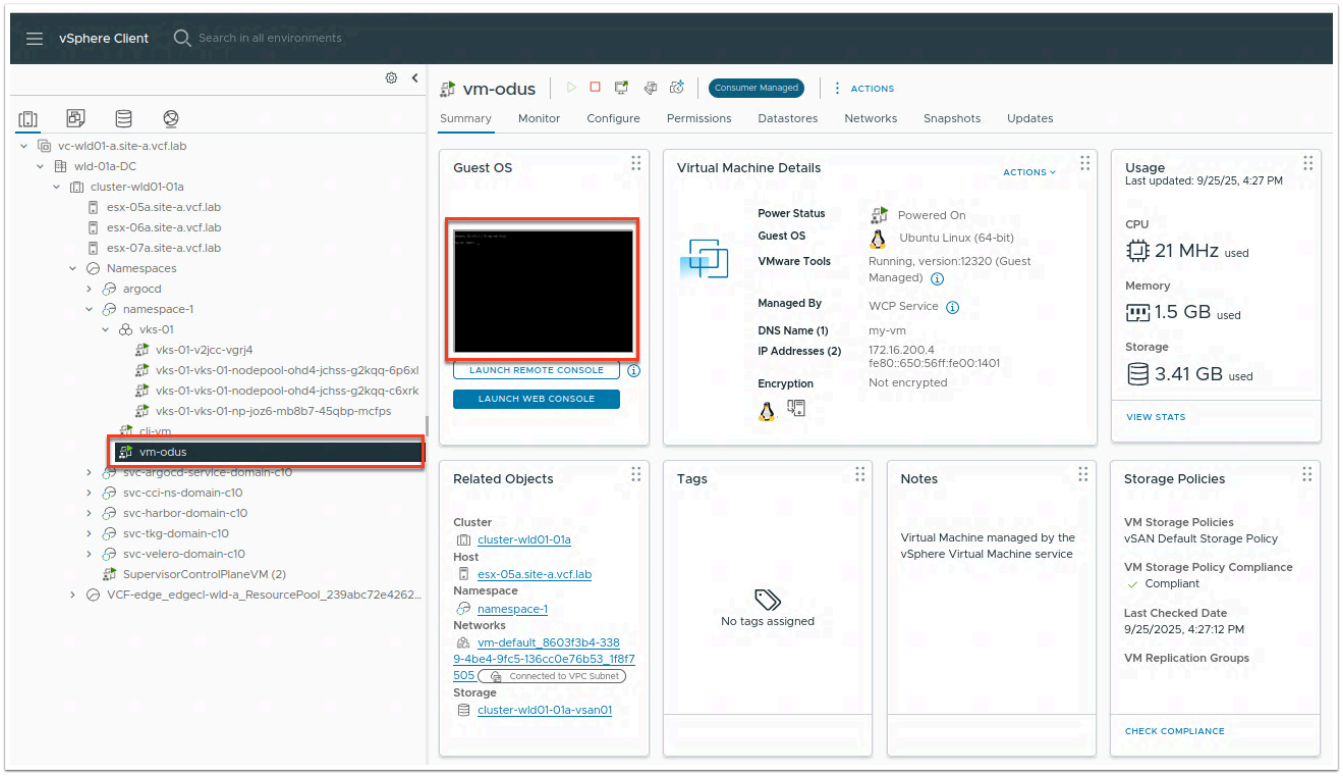
Open the console and log into the new machine.

1. Go to the **Menu** and open **Inventory**



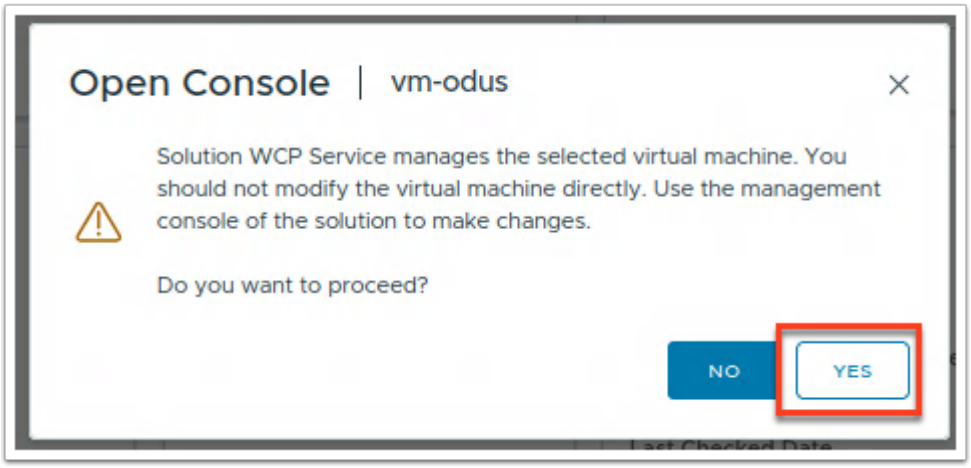
Expand the **vCenter --> vDC --> Cluster --> Namespaces --> namespace-1**. And you will see your new VM coming online.

Open VM Console

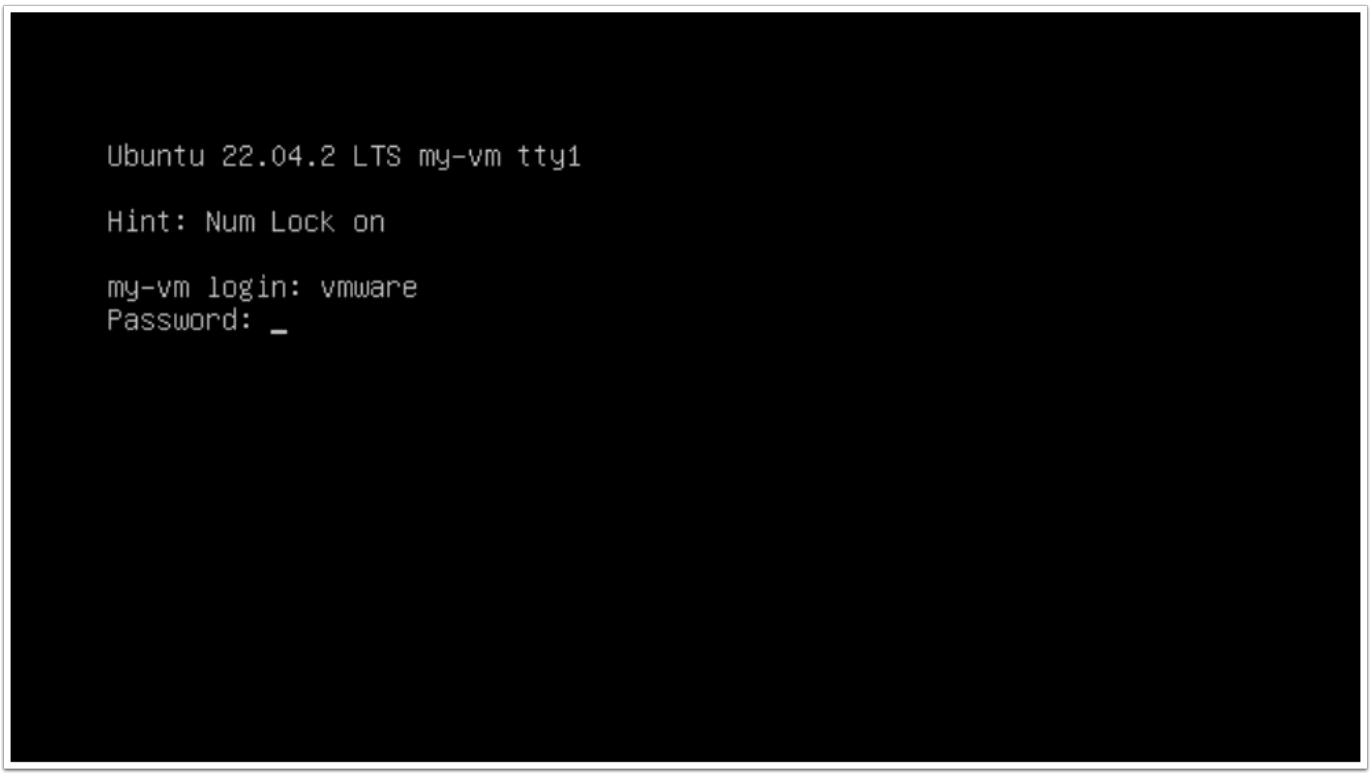


Once the VM is deployed and boots.

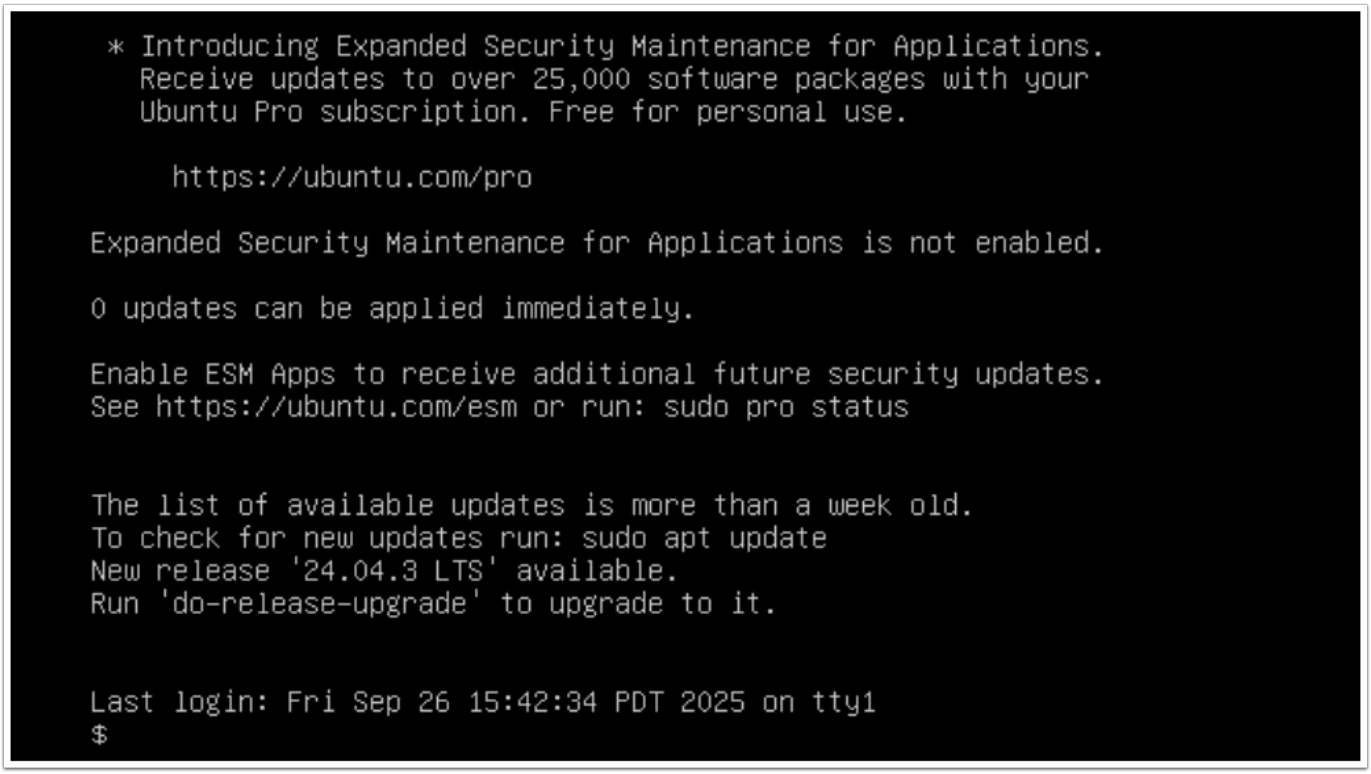
Click the **Console**



Answer Yes

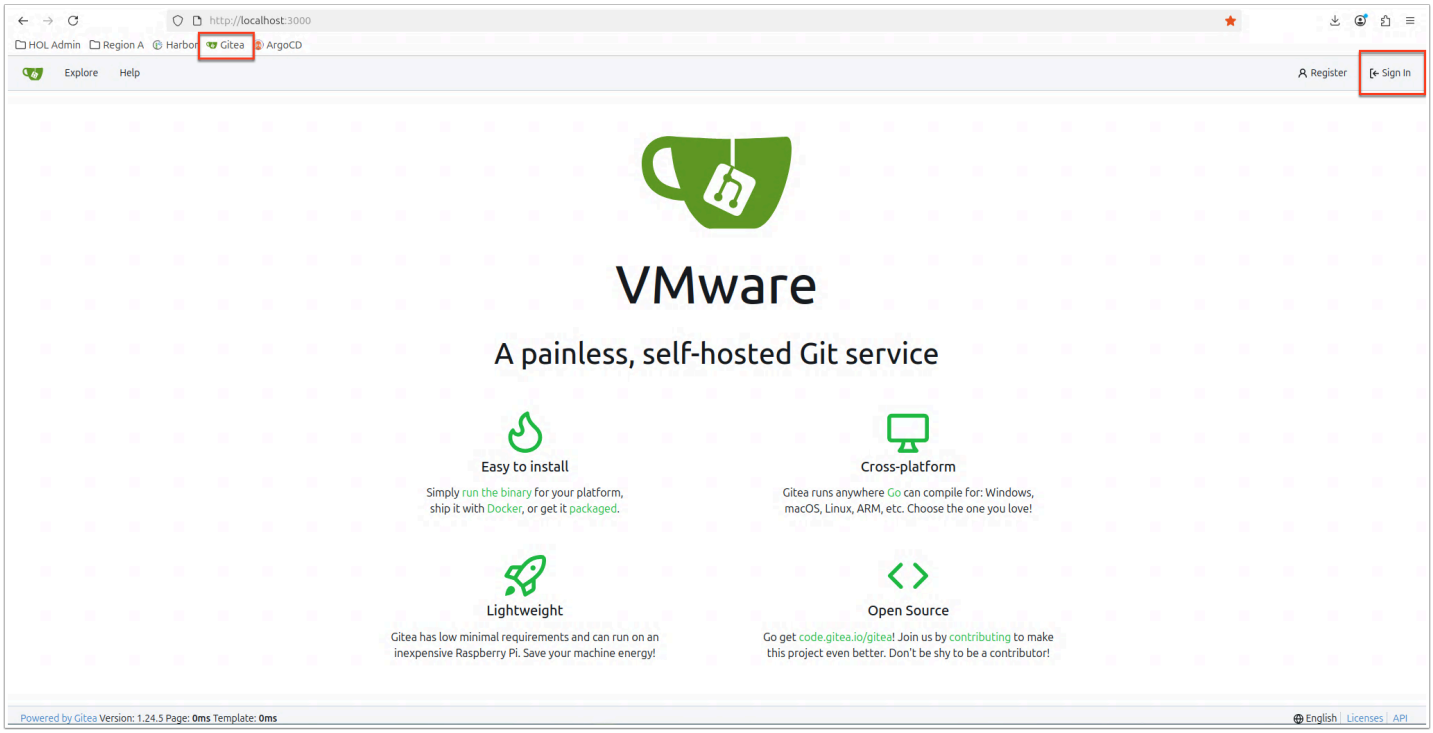


1. Login: `vmware`
2. Password: `VMware123!`



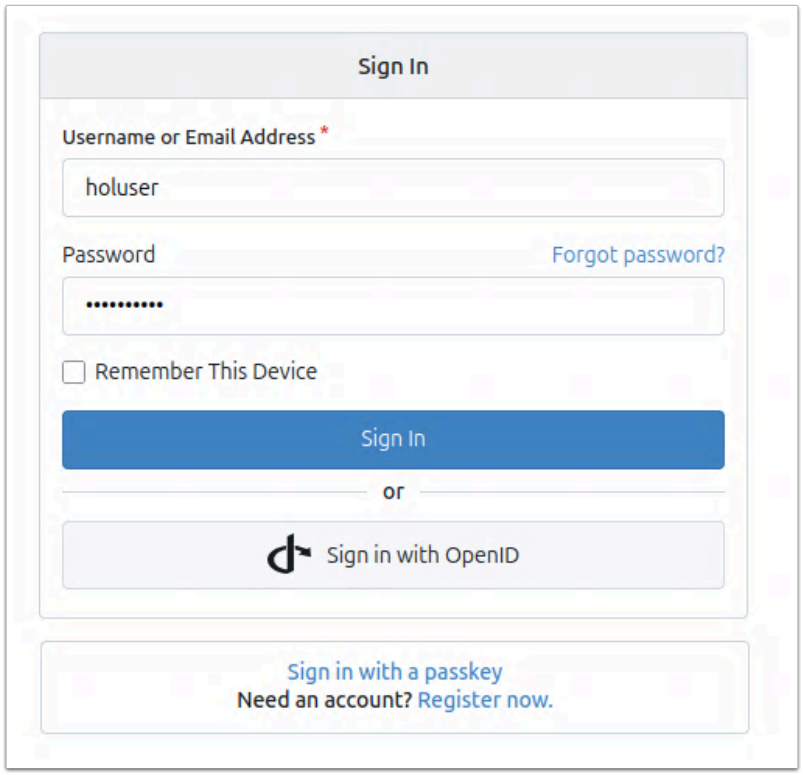
And you are in!

Open Gitea



You have deployed a **Virtual Machine** using the **vSphere Supervisor VM Service**. During the process of creating the **VM** you entered a series of values into the **UI** and the **LCI** created the **yaml** that represents your declared ***endstate configuration***. You are now going to take that yaml and put into a repository in GitEA, then connect that repo to **ArgCD** and direct **ArgoCD** to maintain your desired end state. In cooperation with the **vSphere Supervisor** you will be creating an example of a "**reconciliation engine**" we spoke about earlier.

1. In a new browser tab open the **Gitea** app and **Sign In**



As a reminder

1. Login with : Username

```
holuser
```

Click to copy

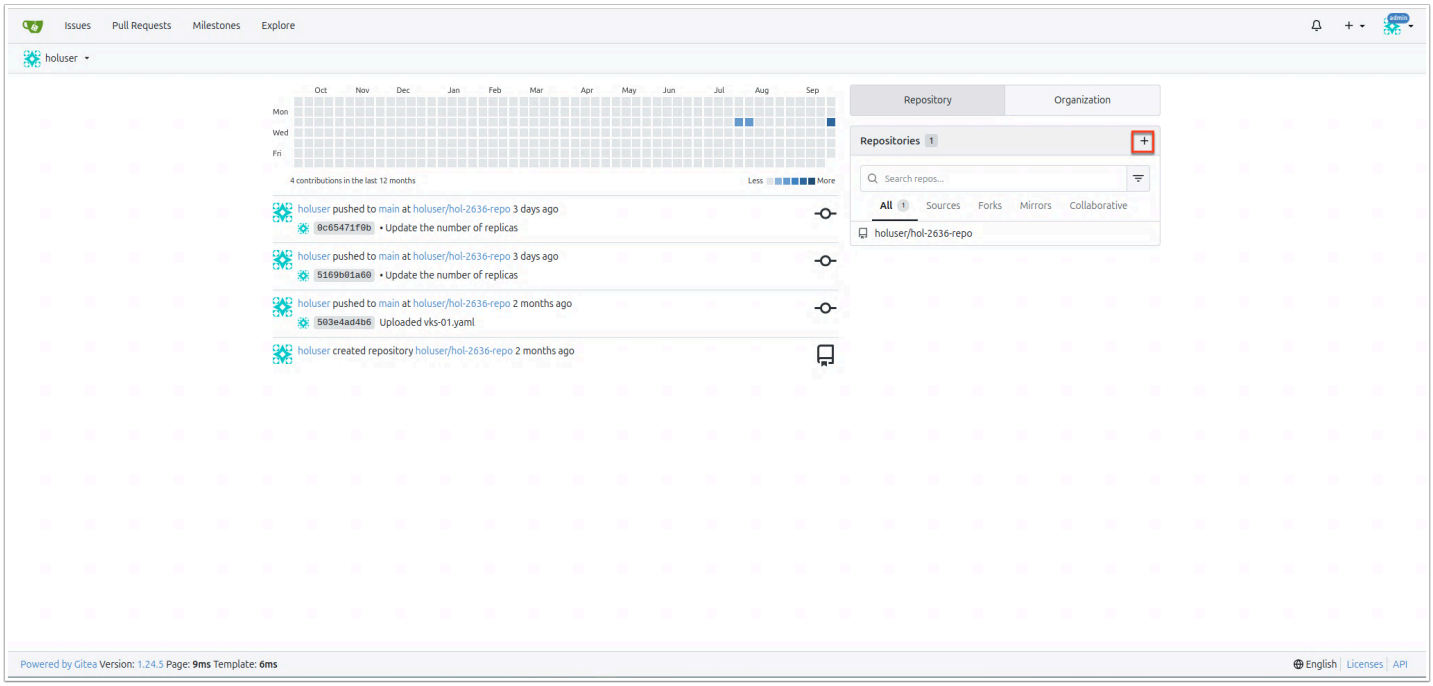
2. Password:

```
VMware123!
```

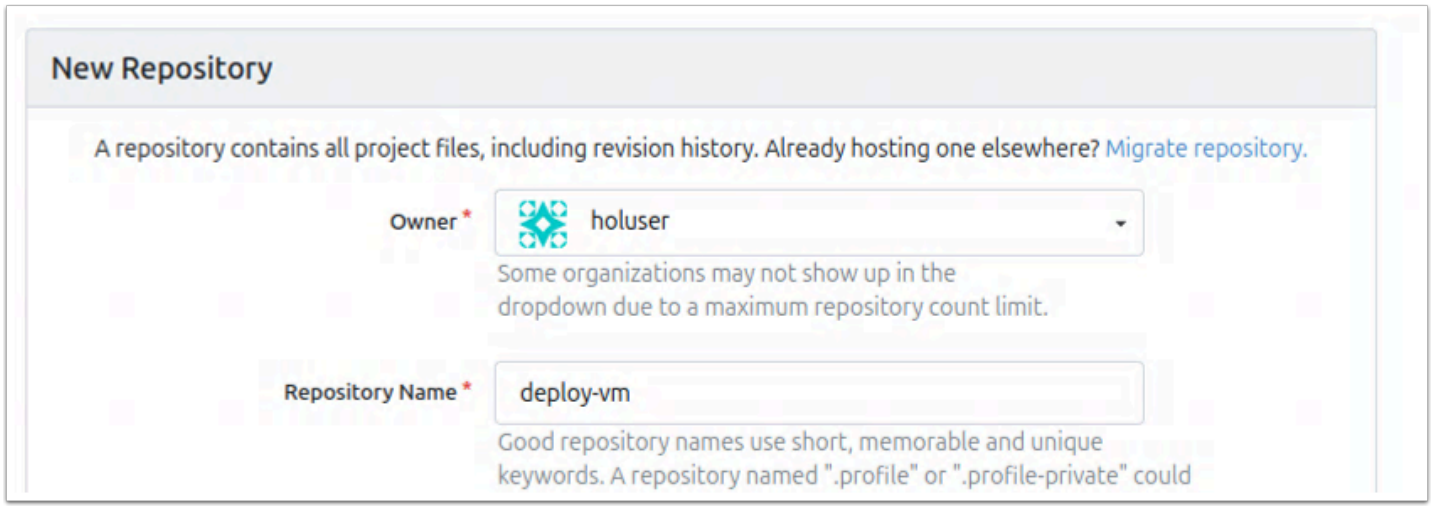
Click to copy

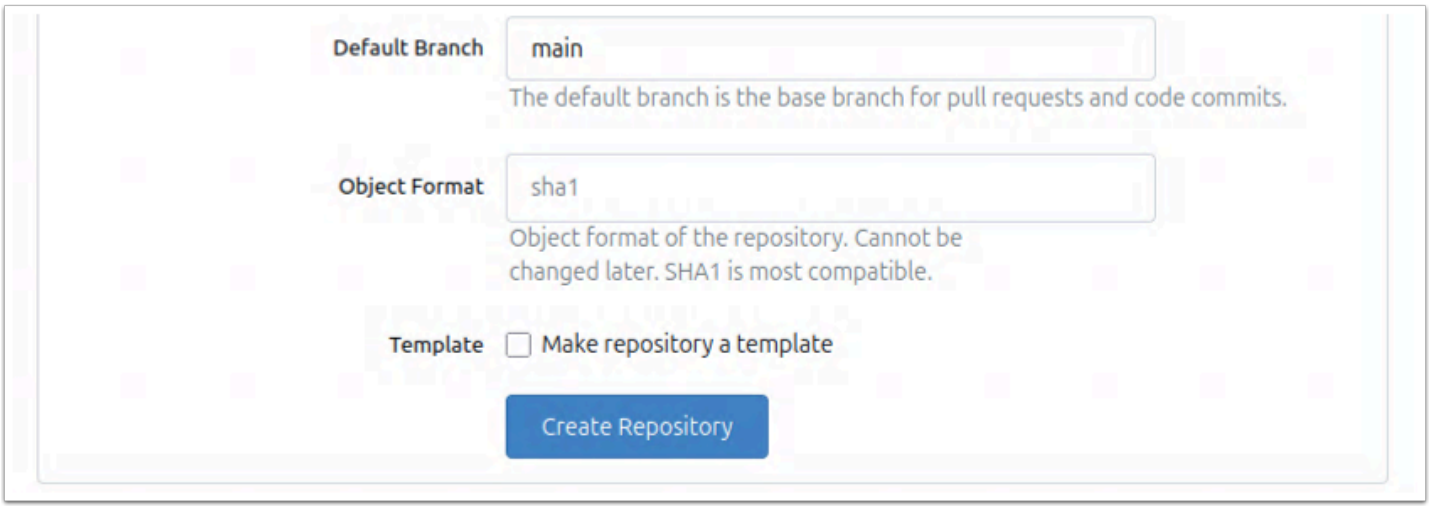
3. Click **Sign In**

Create a new Repo



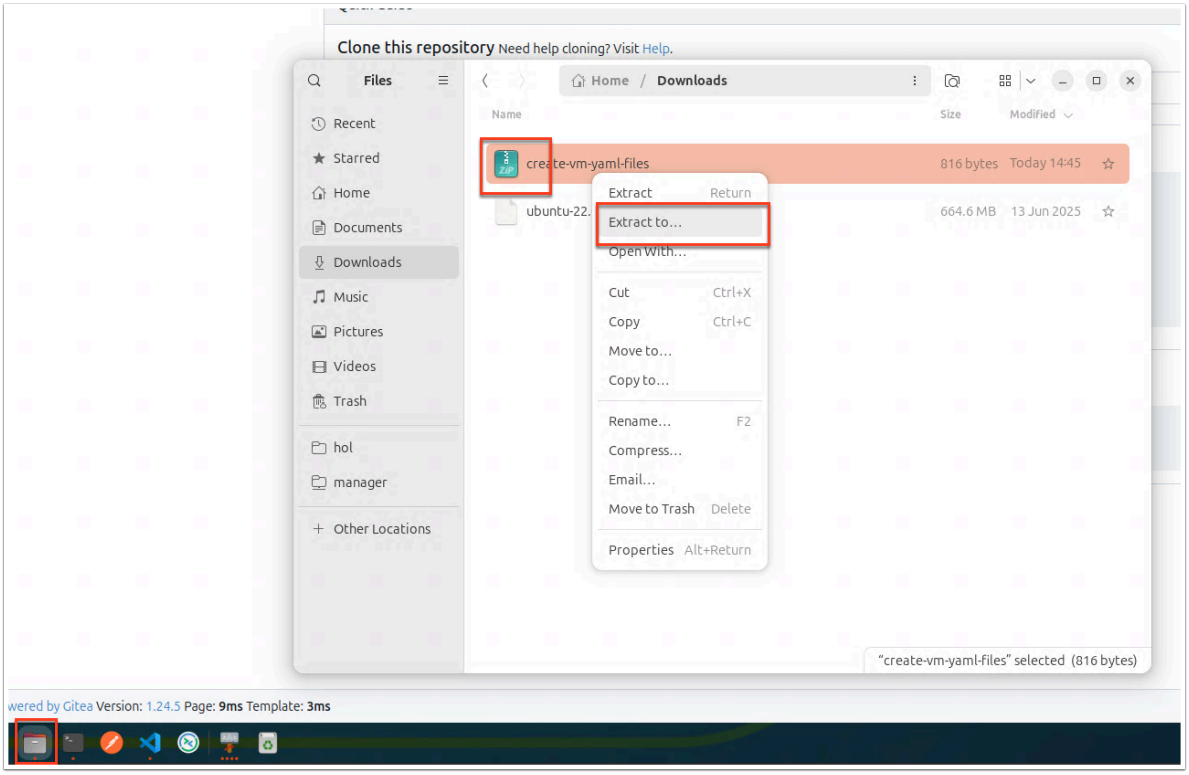
1. Click the **+** to create a new repo

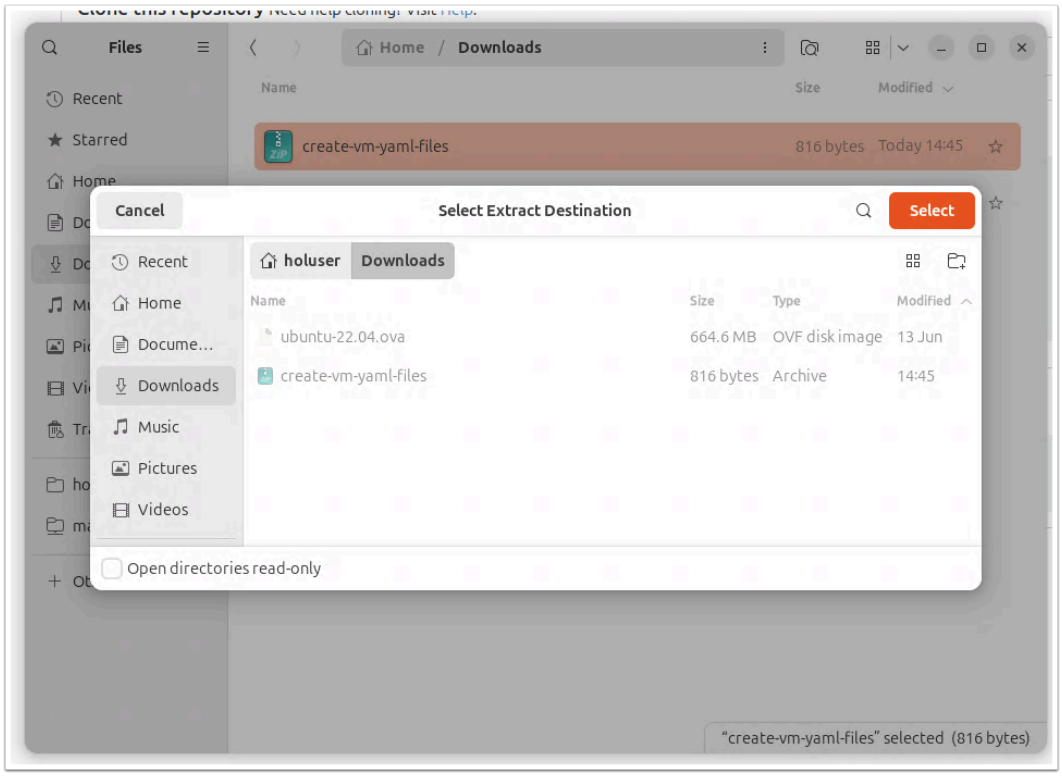


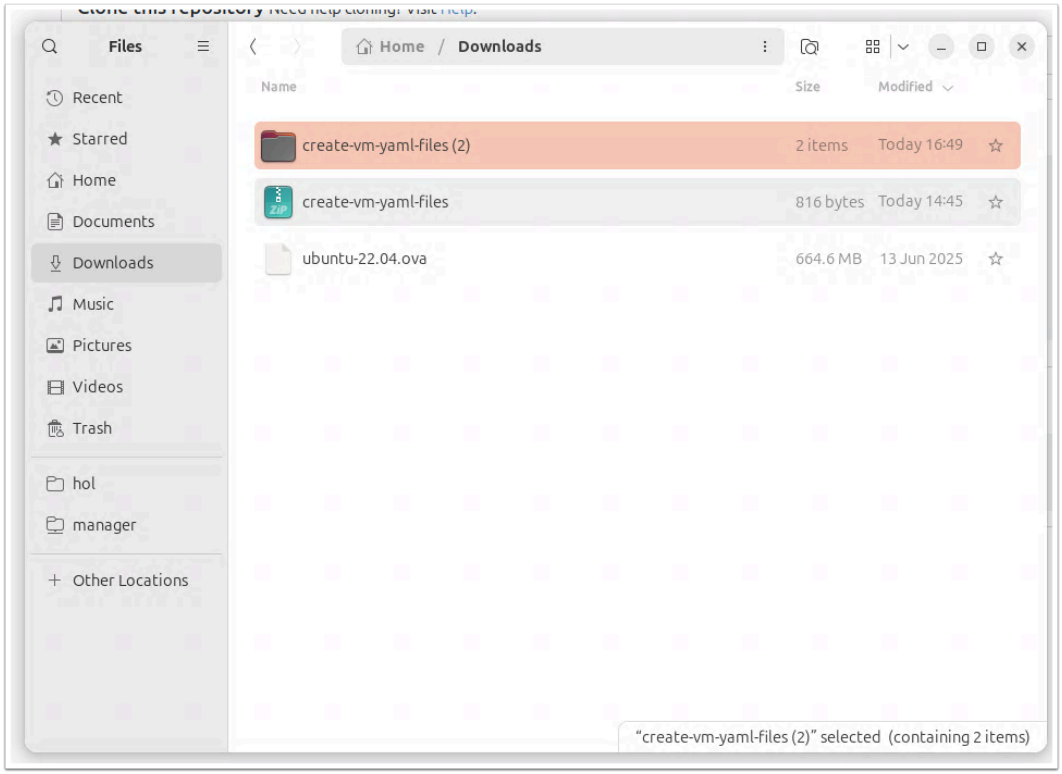


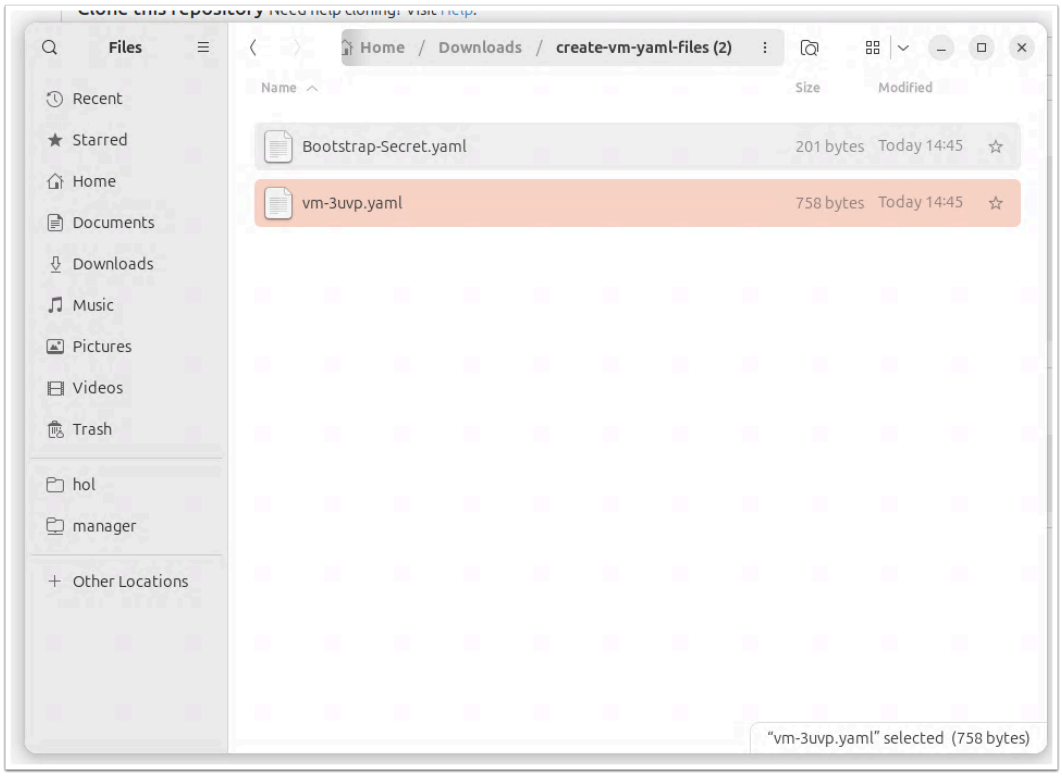
1. Name the repo `deploy-vm`
2. Click **Create Repository**

Unzip yaml so you can upload to your new repo

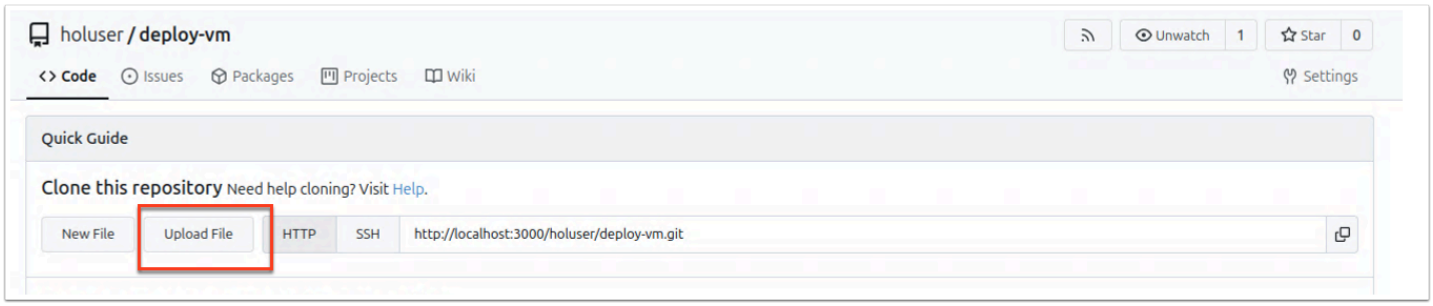



Files
Home / Downloads
Name
Size
Modified
Recent
Starred
Home
Documents
Downloads
Music
Pictures
Videos
Trash
create-vm-yaml-files
816 bytes
Today 14:45
Cancel
Select Extract Destination
Select
holuser
Downloads
Type
ubuntu-22.04.ova
664.6 MB
OVF disk image
13 Jun
Archive
14:45
Open directories read-only
"create-vm-yaml-files" selected (816 bytes)

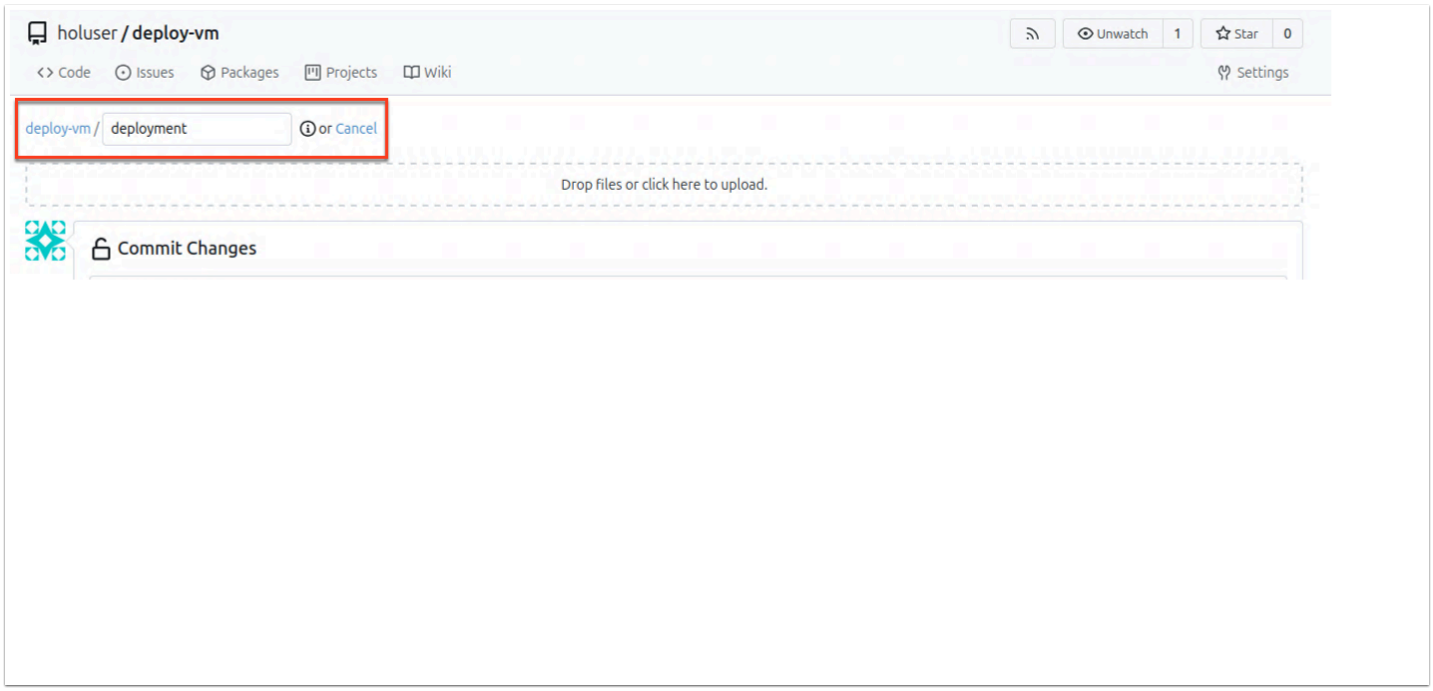

Files
Home / Downloads
Name
Size
Modified
Recent
Starred
Home
Documents
Downloads
Music
Pictures
Videos
Trash
hol
manager
Other Locations
create-vm-yaml-files (2)
2 items
Today 16:49
create-vm-yaml-files
816 bytes
Today 14:45
ubuntu-22.04.ova
664.6 MB
13 Jun 2025
"create-vm-yaml-files (2)" selected (containing 2 items)



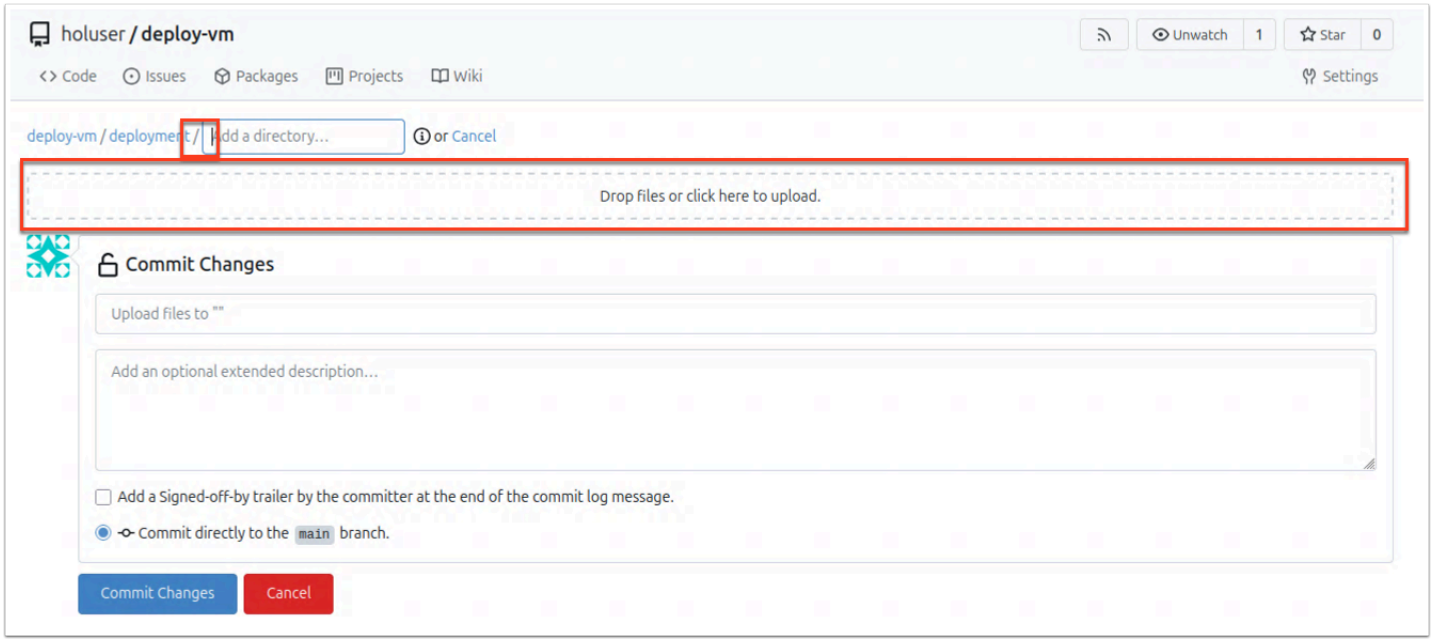
Upload yaml to new repo



Click **Upload File**

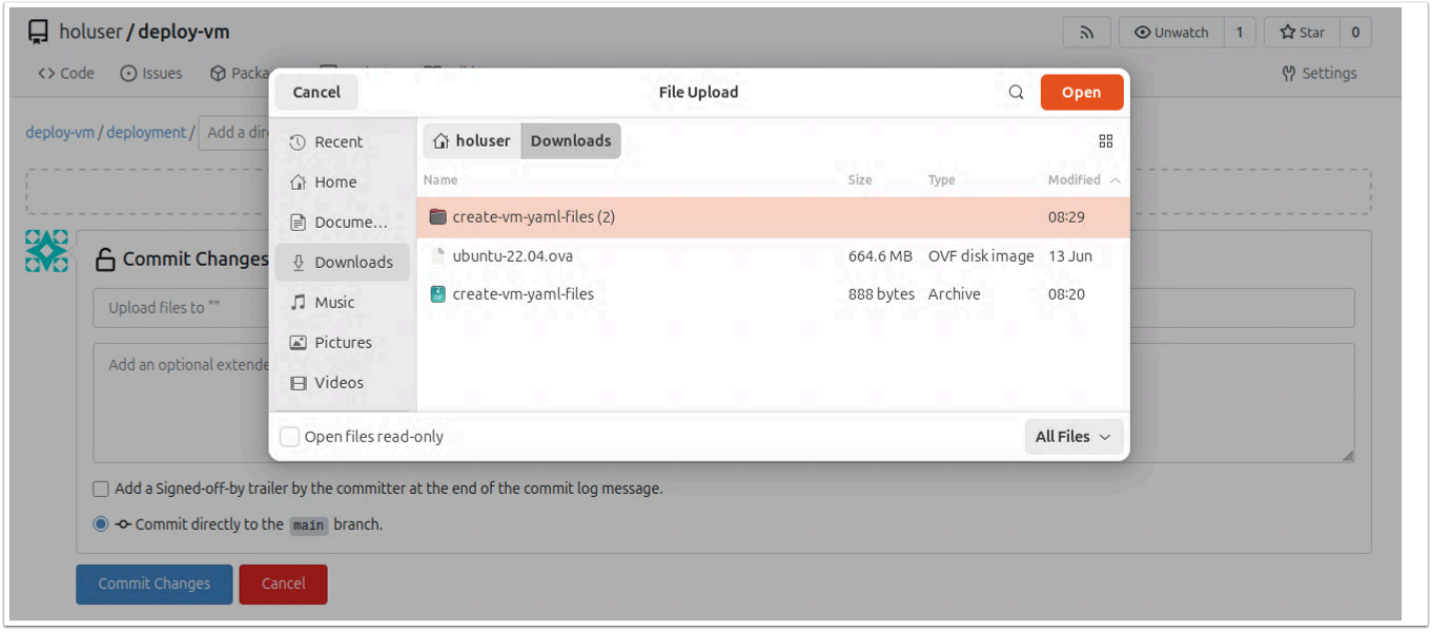


1. In the form at the top type `deployment` - this will be the folder name
2. Then add a `/` after the name



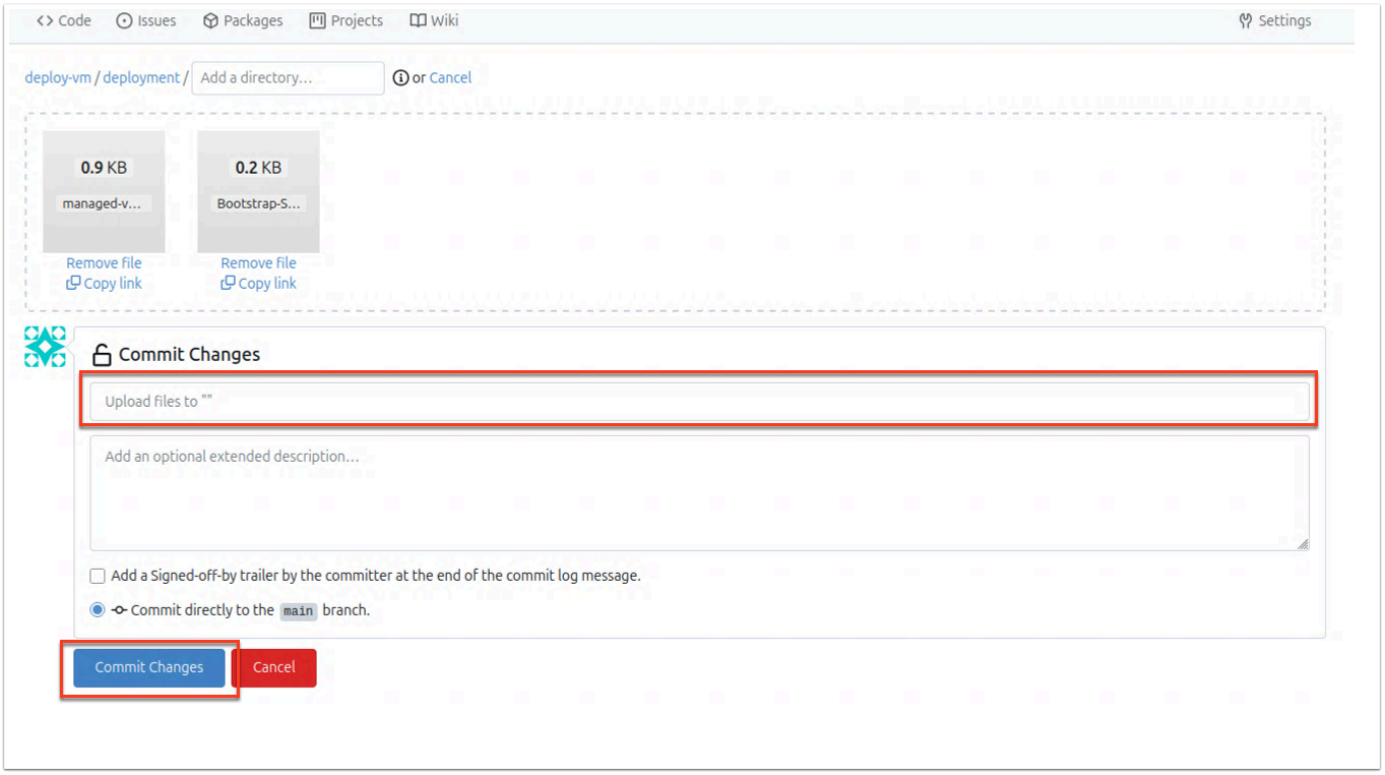
You will see that ***vm-deploy*** *has become a directory.*

1. Click into the **Drop files or click here to upload** form



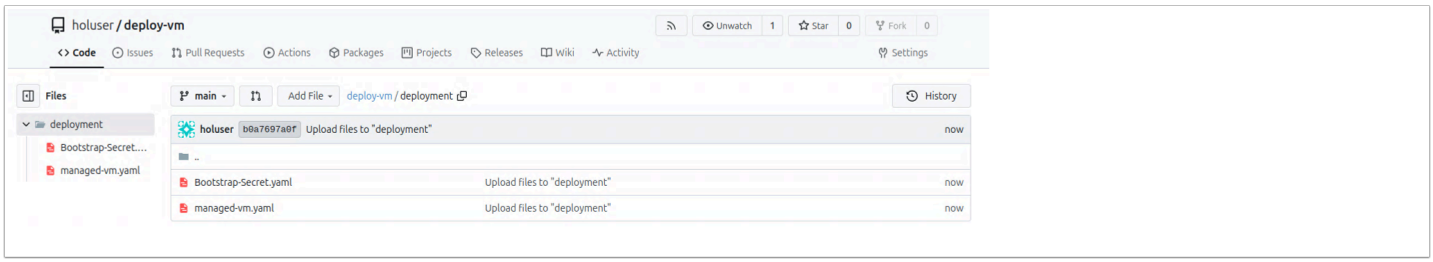
In the file Manager:

1. Click **Downloads**
2. Click the folder that extracting the downloaded zip create
3. Select both yaml files
4. Click **Open**

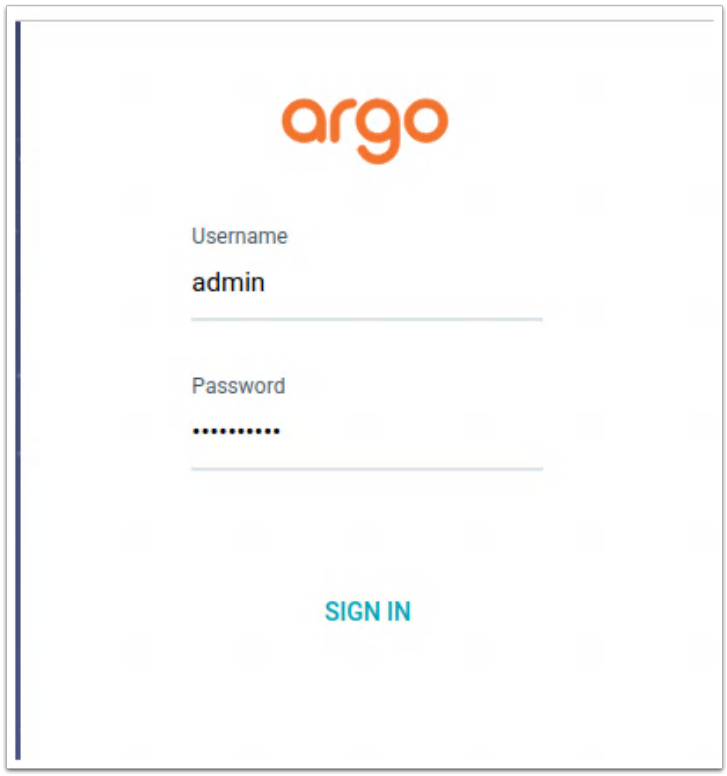


This part is standard git repo commit and sync behavior. Every time a change is committed to the repo the developer (you in this case) must document what is being done.

1. At a Minium you must add a single line to this top field, it can as simple as one word - **"Update"** or it can be much more. I leave this up to you.
2. The description can be left blank or you can be as verbose as you wish.
3. Click **Commit Changes**



Login To ArgoCD



Log Into ArgoCD

Username:

```
admin
```

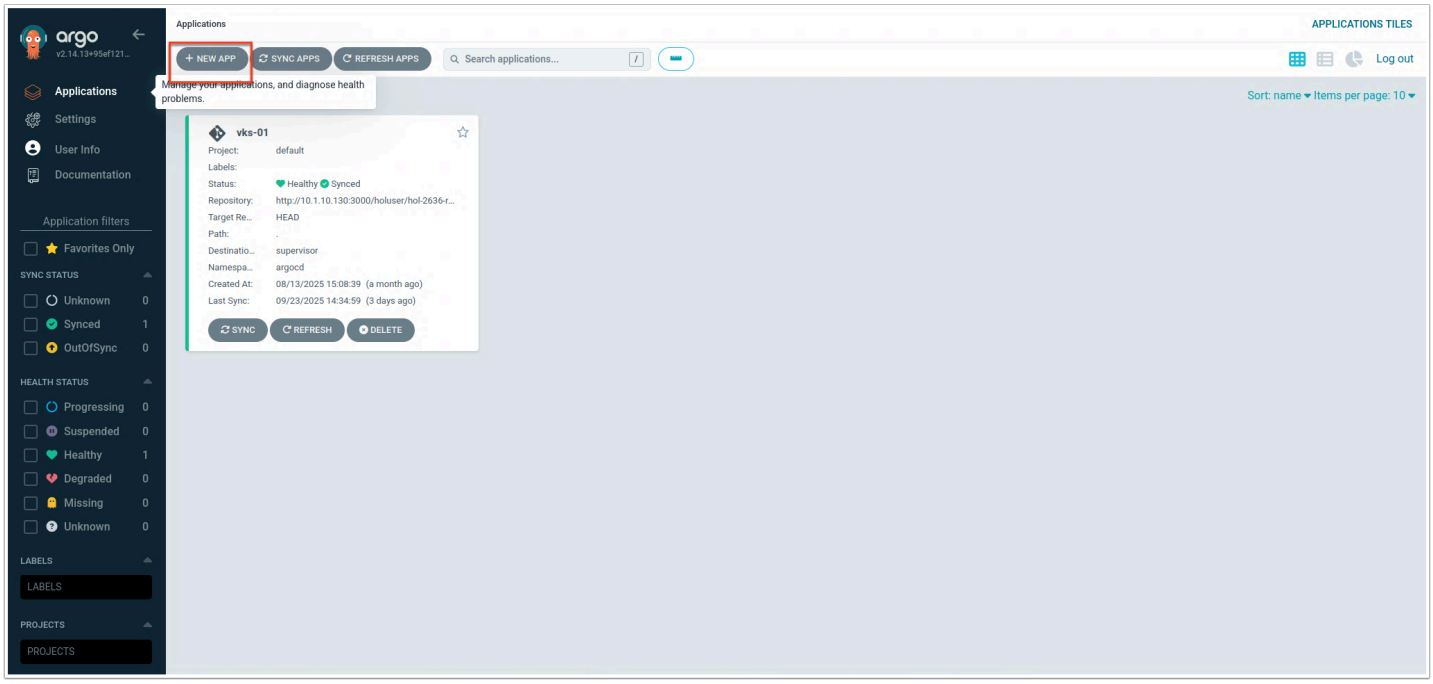
Click to copy

Password:

```
VMware123!
```

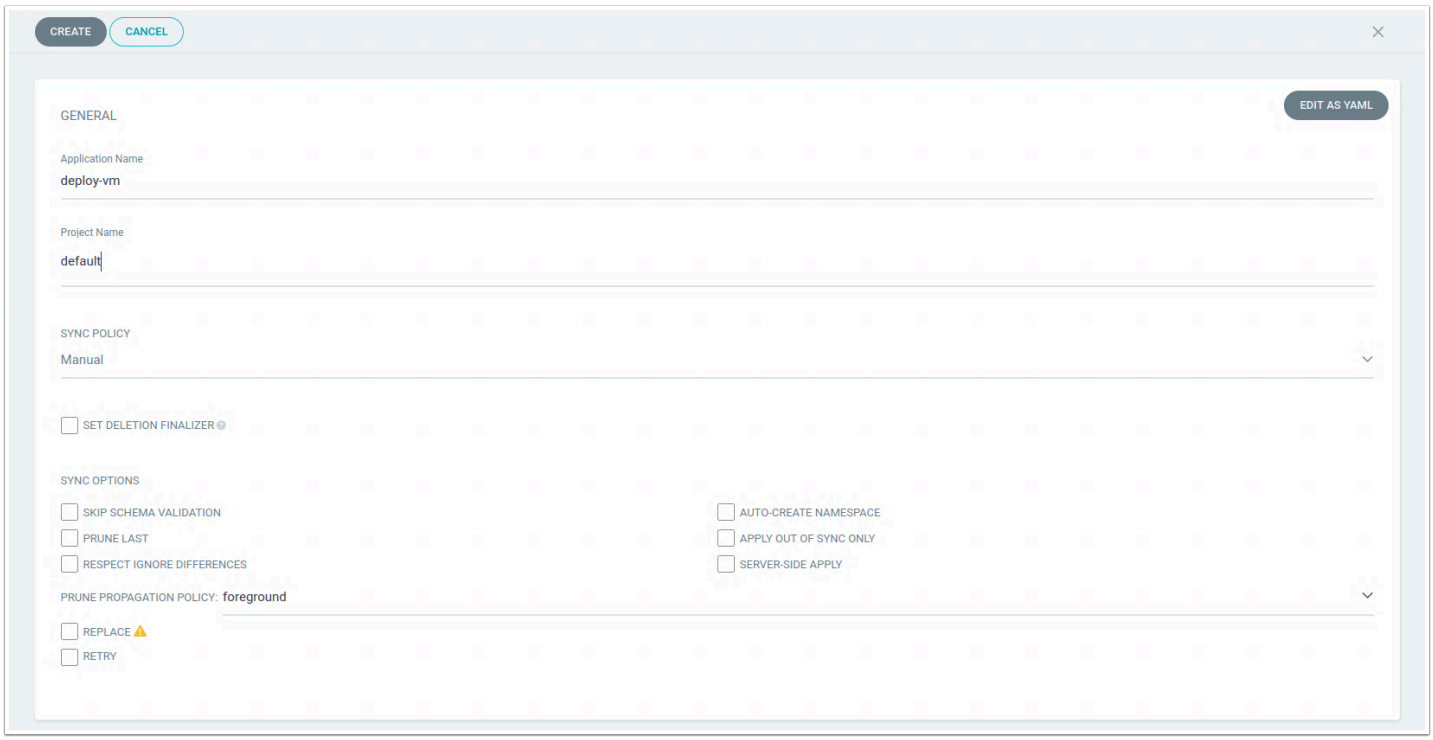
Click to copy

New ArgoCD App



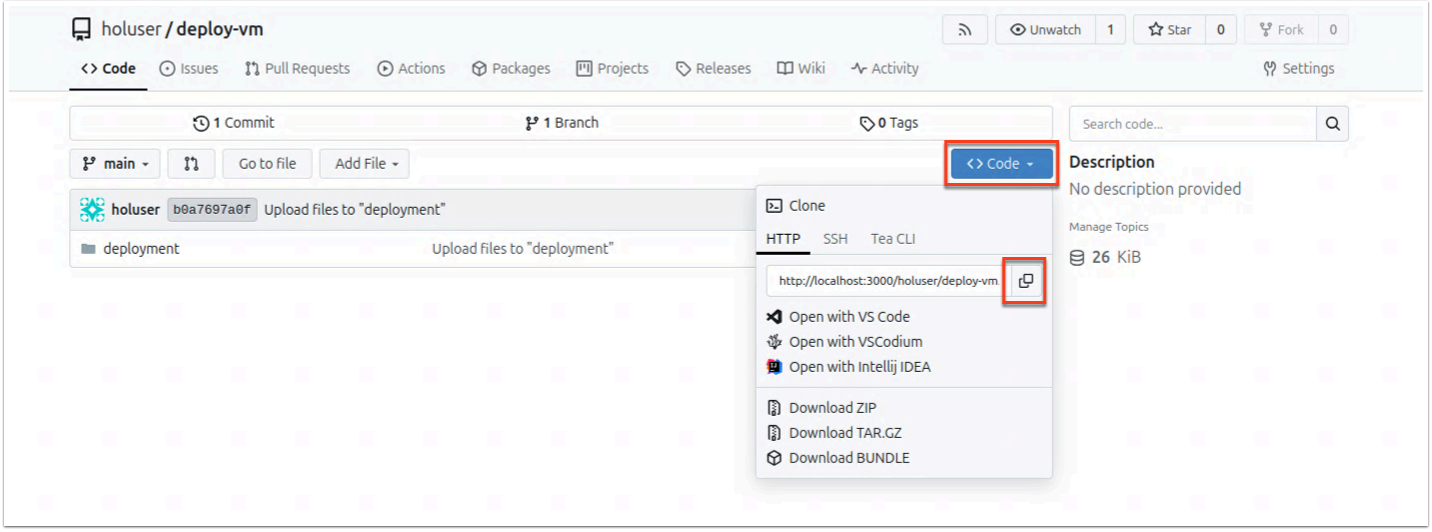
You are now going to create a new app. much like the existing App and see how ArgoCD keeps your VM in your **declared *endstate configuration.***

1. *Click* ***+ NEW APP***



1. For **Application Name:** `deploy-vm`

2. For **Project Name:** click into field and select `default`



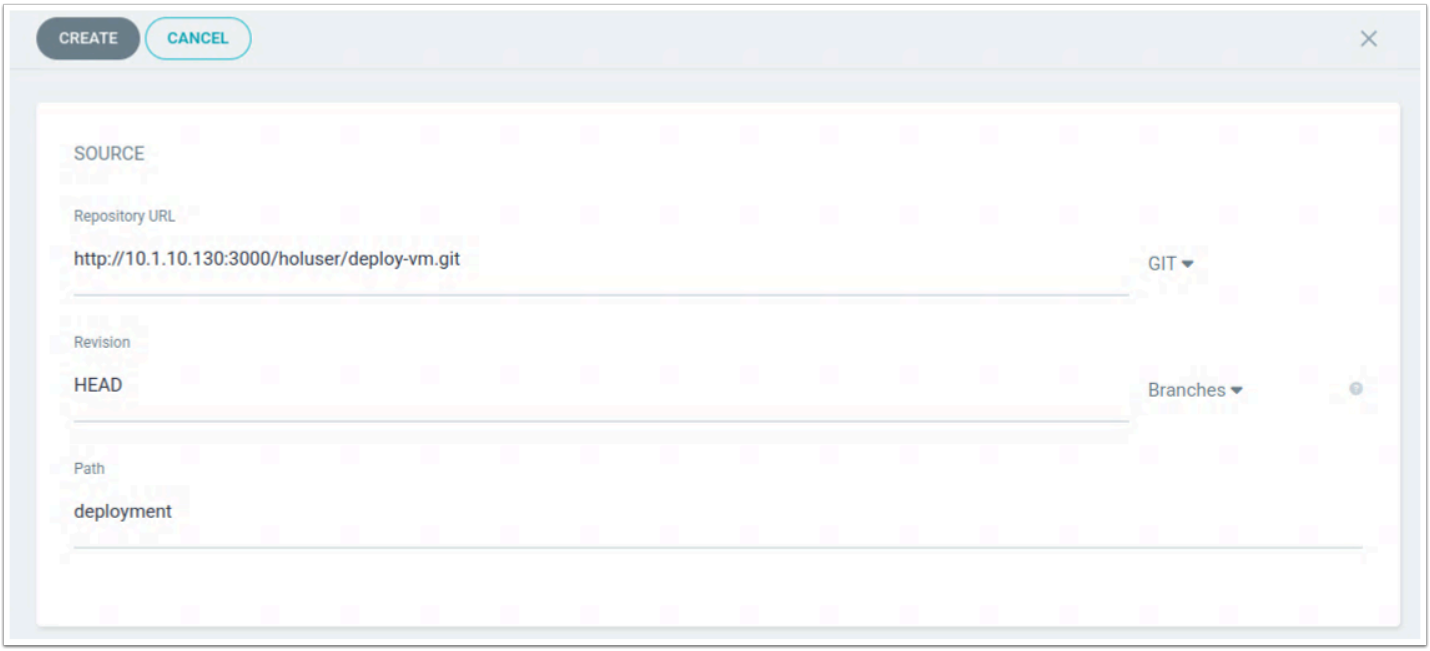
Go back to the **GitEA UI** and the **vm01** repo

1. Click the **CODE** Dropdown
2. Click the **Copy to Clipboard** for the **HTTP Clone action**

The **URL** as it is will not work, it is handed to you as http://localhost, but the ArgoCD app needs the hostname or IP. So change `localhost` to `10.1.10.130` so it looks like this

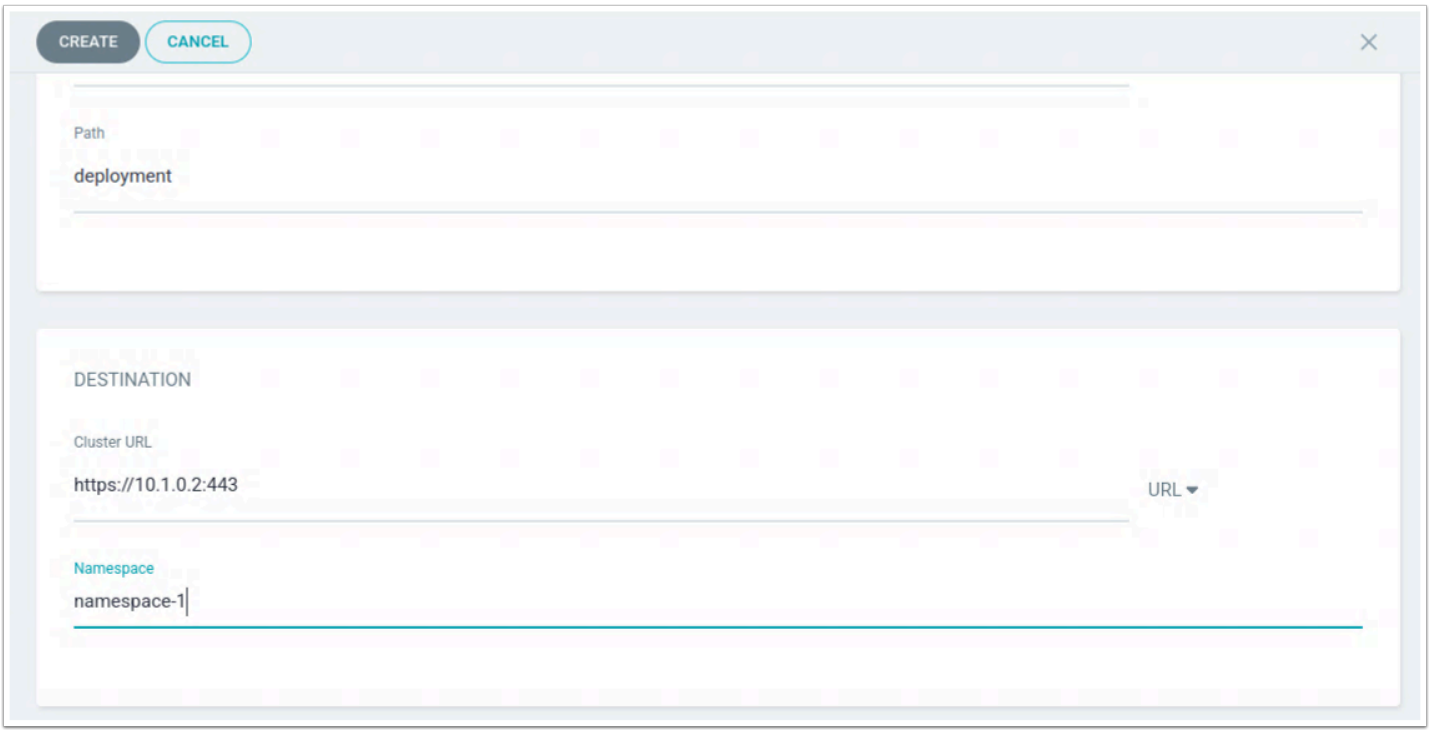
```
http://localhost:3000/holuser/deploy-vm.git
```

Click to copy
Copy the new **URL** into your clipboard



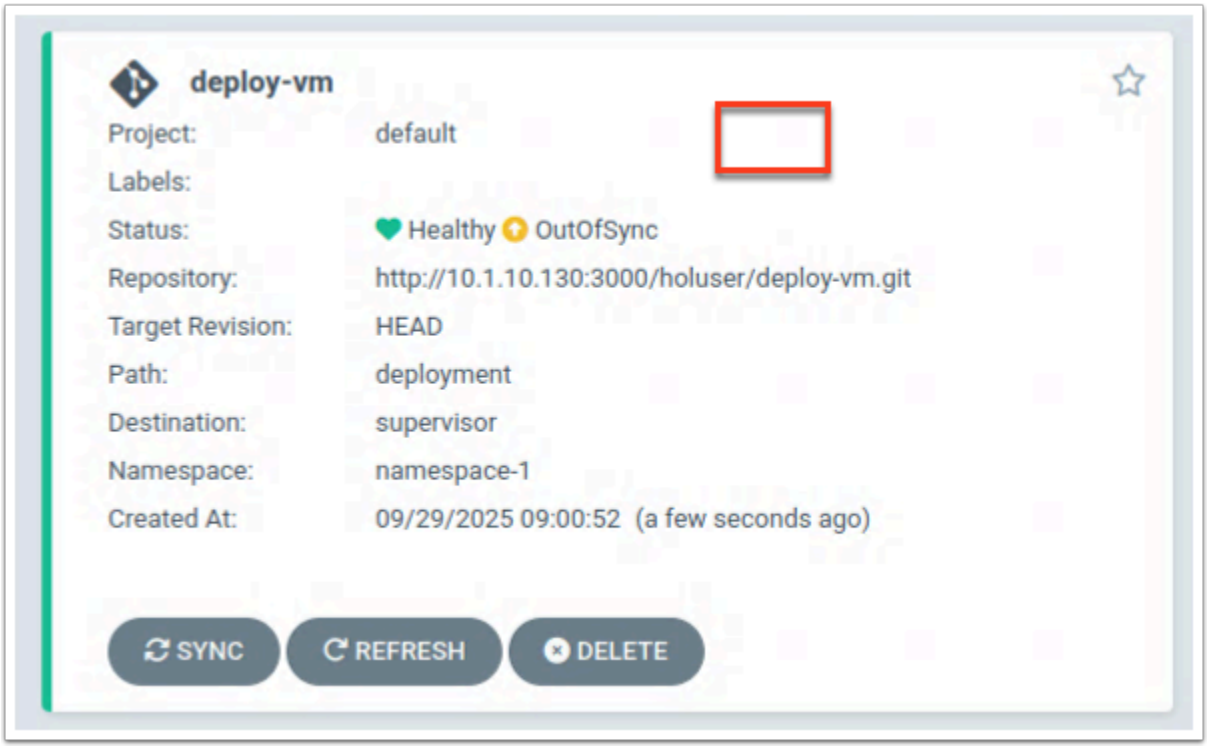
1. For **Repository URL:** paste your edited URL - Hit **ENTER -** this tells ArgoCD to check the URL

2. Click into **Revision:** select HEAD - ***Note: it may already say HEAD, but do this anyway***
3. For **Path:** type `deployment` - the folder we put the deploy yaml into.

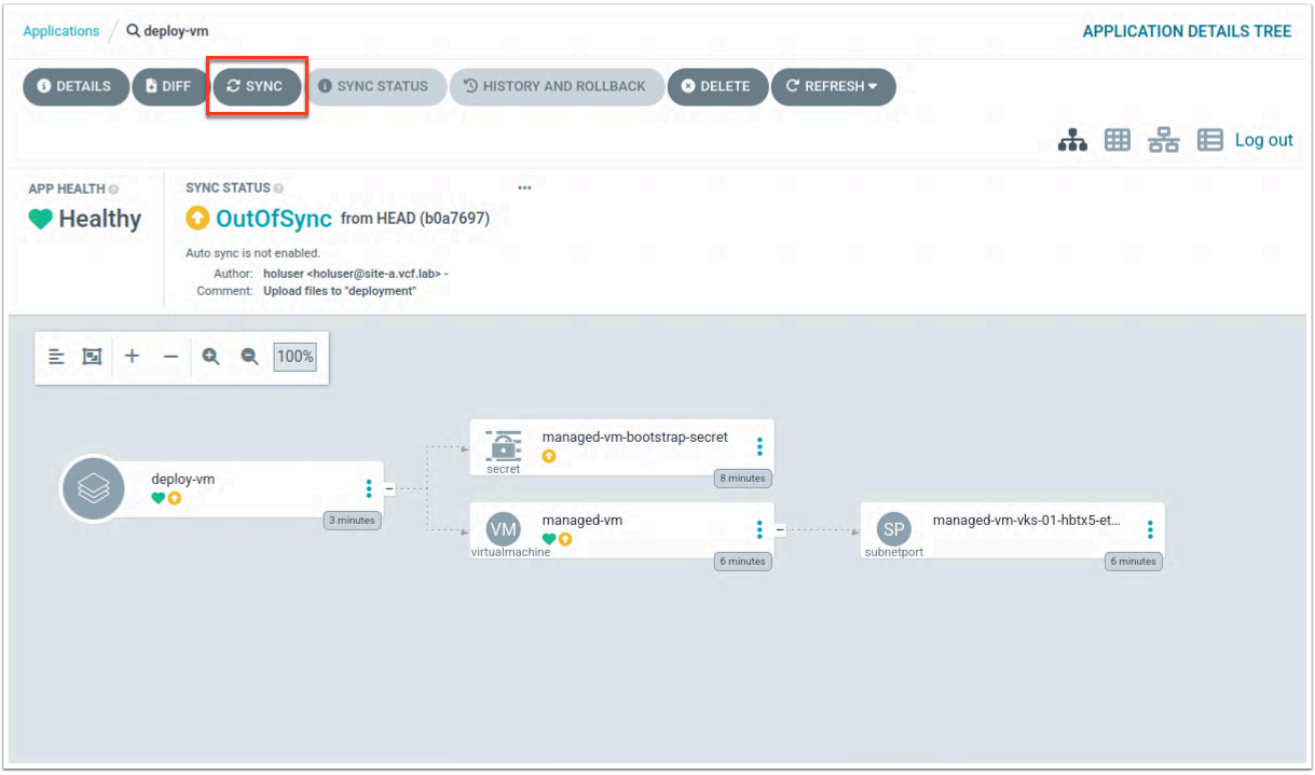


4. Click the **Cluster URL Dropdown** select the only entry - `https://10.1.0.2:443` - this the Supervisor IP. In a non lab environment this would be a most likley be the FQDN for a Load balanced, three node Supervisor
5. Enter `namespace-1` for **Namespace**
6. Click **CREATE**

Sync ArgoCD Application



1. Click White Space on the App tile

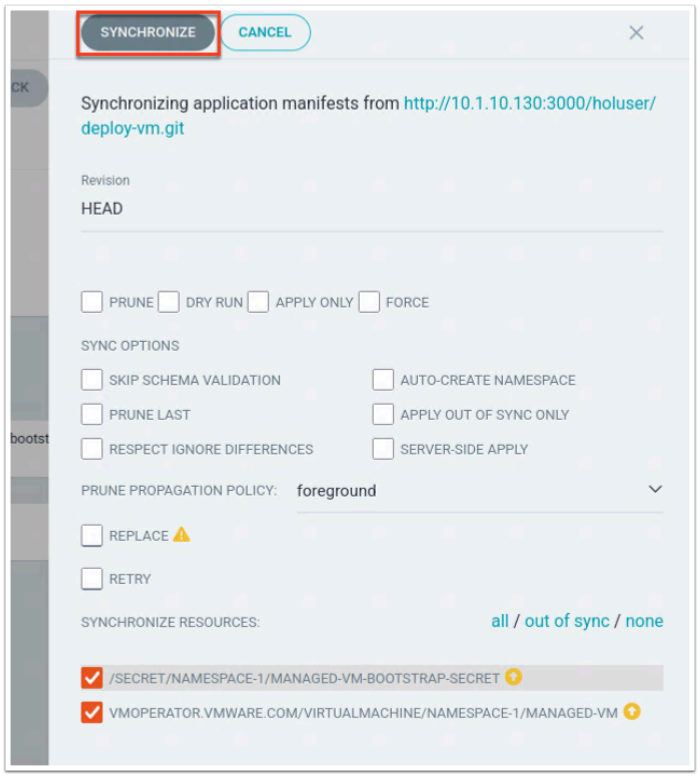


The ArgoCD UI is pretty simple. Each of the icons in this screen give you information on the state of the assets ArgoCD is managing. The heart, as you might have guessed references "HEALTH" and the Circle shows the status of Synchronization of the asset.

In this example the green heart is saying the VM has been found and seems to be healthy, BUT the yellow SYNC icon is saying it may be out of sync from the declared endstate.

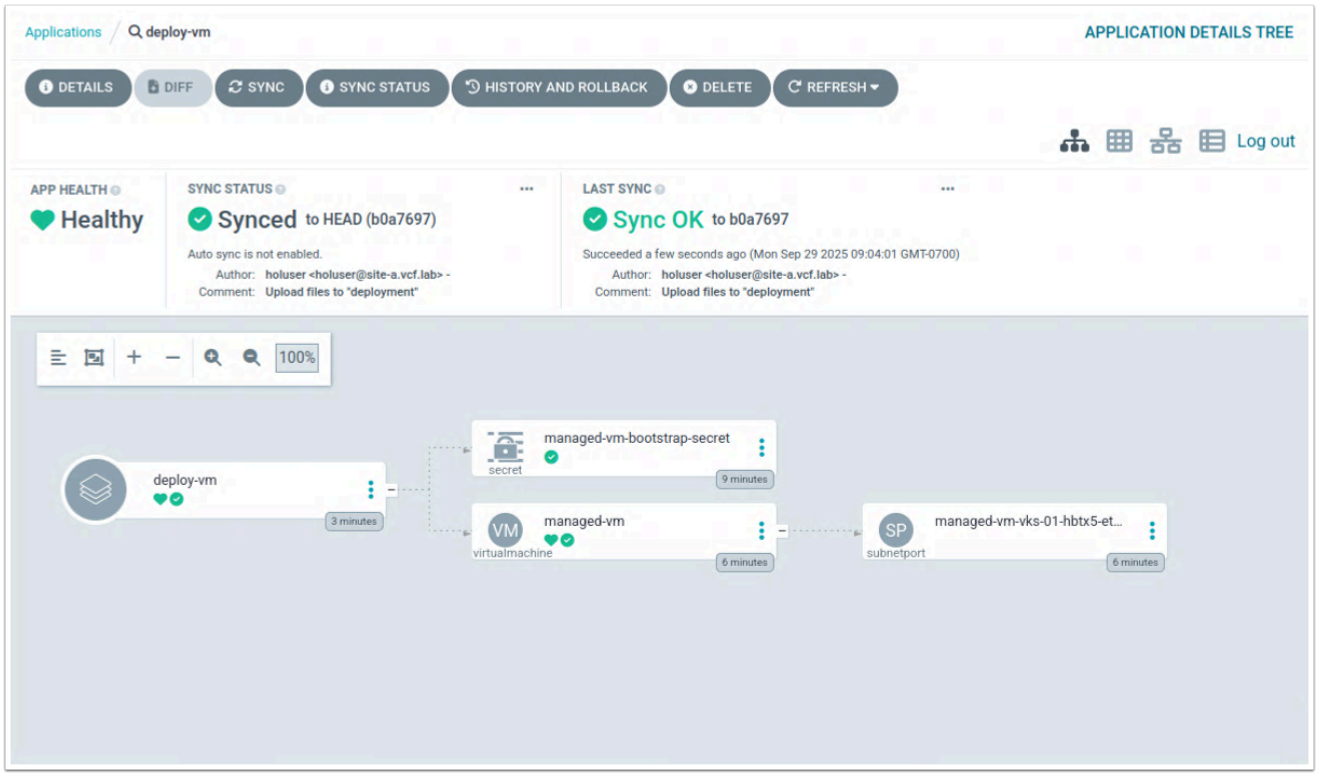
So...lets synchronize it

1. Click **SYNC**



In the new window you are going to leave all the settings as is. We want to leave the two assets it has recognized as needed to be synchronized. Leave the bootstrap secrets file and the vm deployment yaml checked.

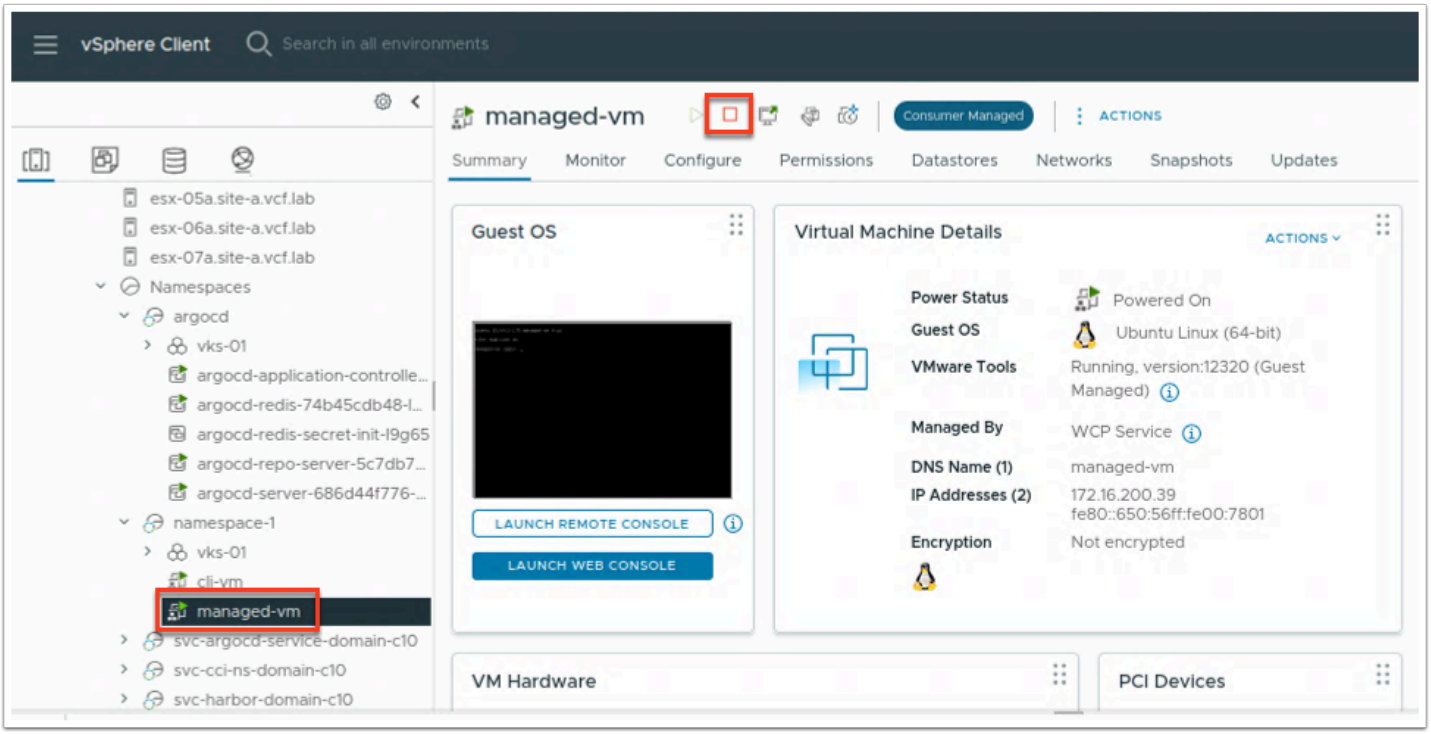
1. Click **SYNCRONIZE**



In a few seconds you should see the status change to all green, healthy and synchronized. This means the managed vm is healthy running, in existence as expected, and it is currently in the expected endstate as declared.

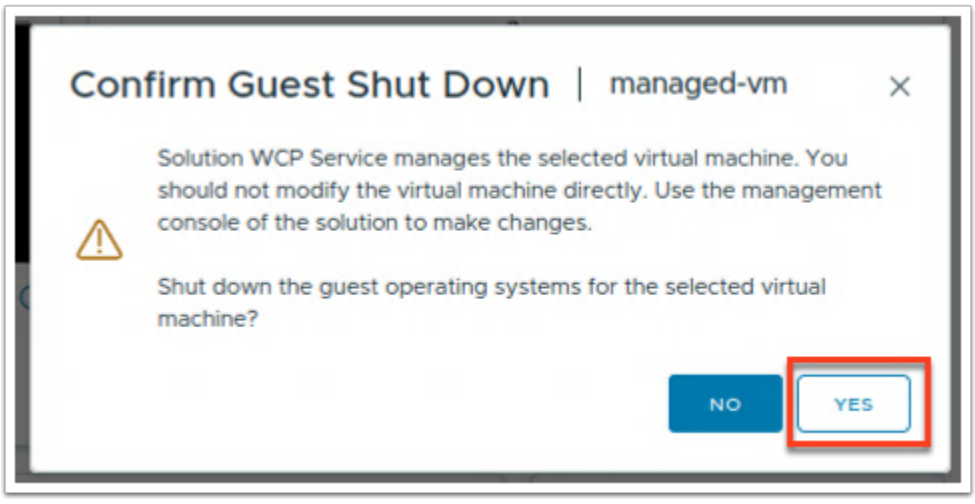
SYNCRONIZED!!

Now Break The Managed VM



Now we break it!

Put your pesky ***know it*** all **VI Admin** hat on. Log into the **wld01 vCenter**. Find your **VM** in **inventory** under **namespace-1** and click **power off**



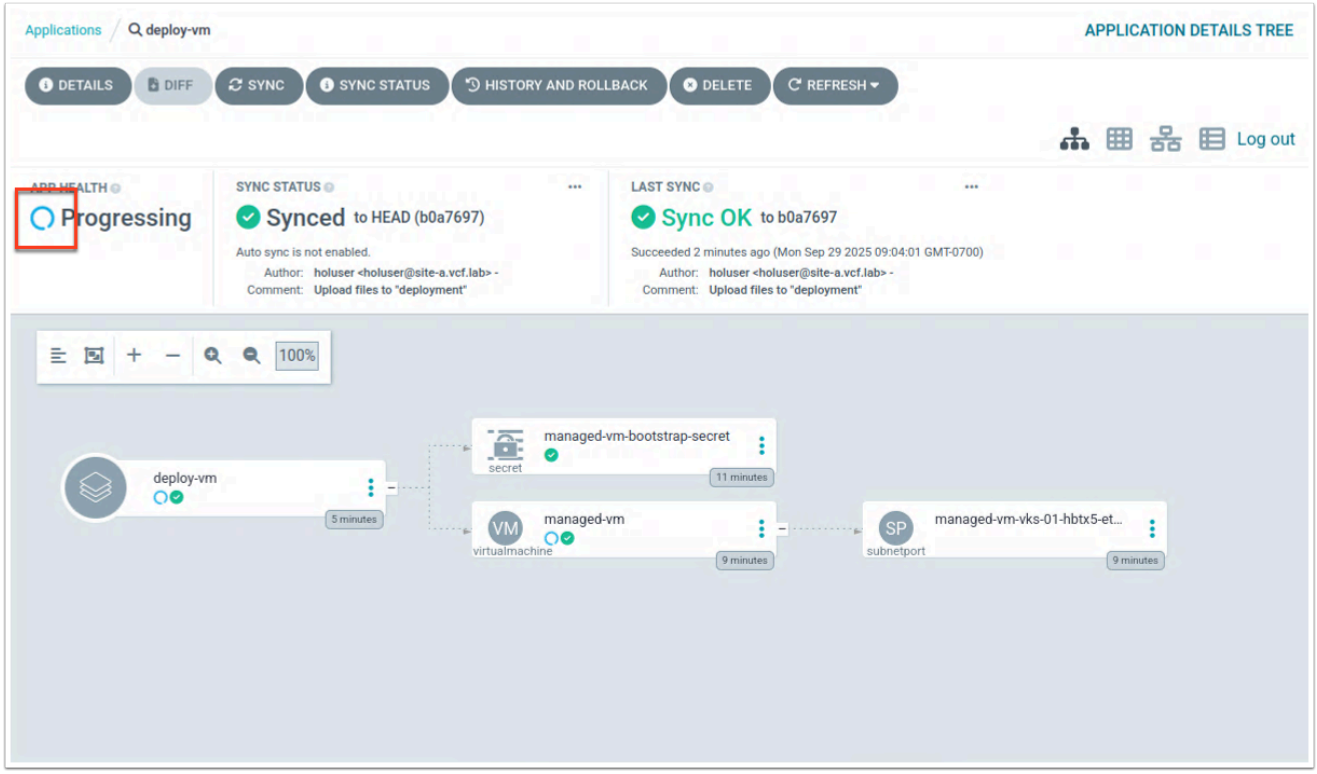
"WAIT HOLD ON! "

"This VM is managed by another system?!"

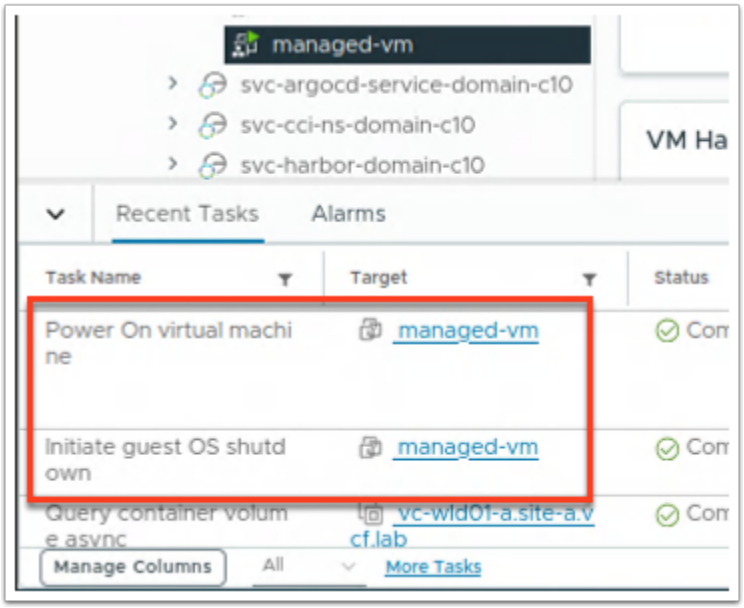
"What the heck is the WCP Service?"

"I know better!"

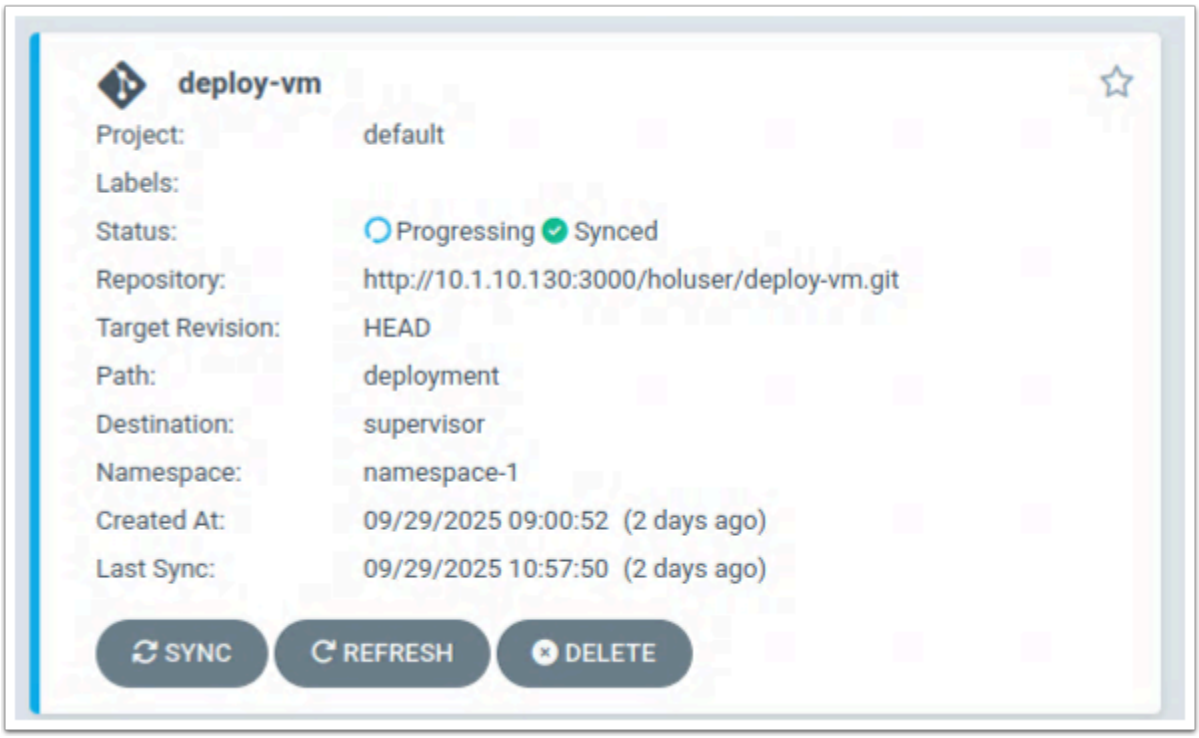
1. Click **YES**



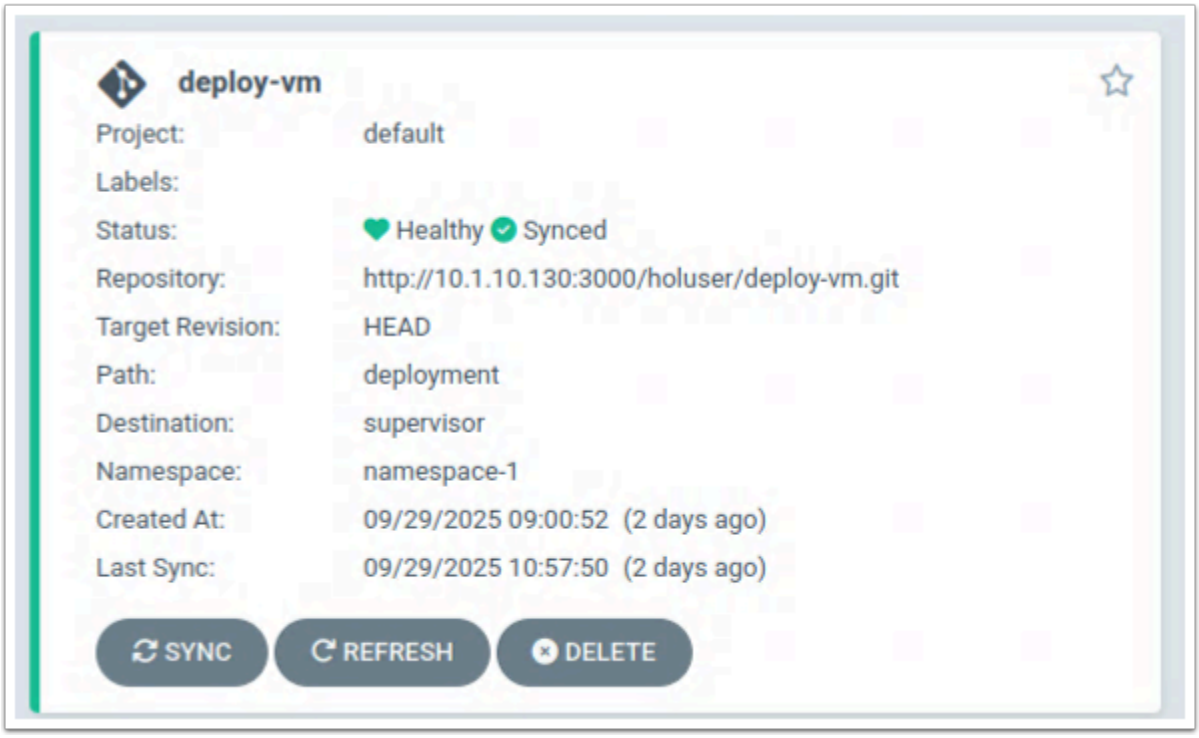
Swap back to the **ArgoCD UI**, and you will see the health gets put into **Progressing**.



Swap back to the **wld01 vCenter** and in tasks you will see the **VM Shutdown**, then ..**PowerOn**!!?



Back to ArgoCD



And as soon as the VM reports back in the expected state, it goes healthy.

From here there are a bunch of other things we can try, like change the encoded password in the secrets file. Change storage policies. Some of them will work as expected some will take a bit more pre-work to get the full workflow interrated and working. This is meant to be a technical intro. Explore you or your customers requirements, but a real use and go from there.

Additional Information

Additional Information

Thanks to all who took the time to take this lab and for your interest and support for the VCF/ VKS service. For more information on VKS and Supervisor Services:

https://blogs.vmware.com/cloud-foundation/?s=breakroom+chats

The following two videos walk thru the updates of the Supervisor and VKS which were covered in Module 4.

NOTE : We cannot guarantee that headsets will be available in the Hands On Labs area so as a courtesy to those around you, if headsets are not available please watch and listen on a personal device or use your own headset.

Supervisor update and VKS update videos:

https://youtu.be/KOeVCcr7aa0

https://youtu.be/bCUVPJ-SnMA

VKS Tech Talk:

https://www.youtube.com/watch?v=F3CdrmgjVN4

Conclusion

In this module, we demonstrated how to use gitea and ArgoCD to demonstrate continuous deployment using the ArgoCD supervisor which we deployed for you.

End Of Lab manual