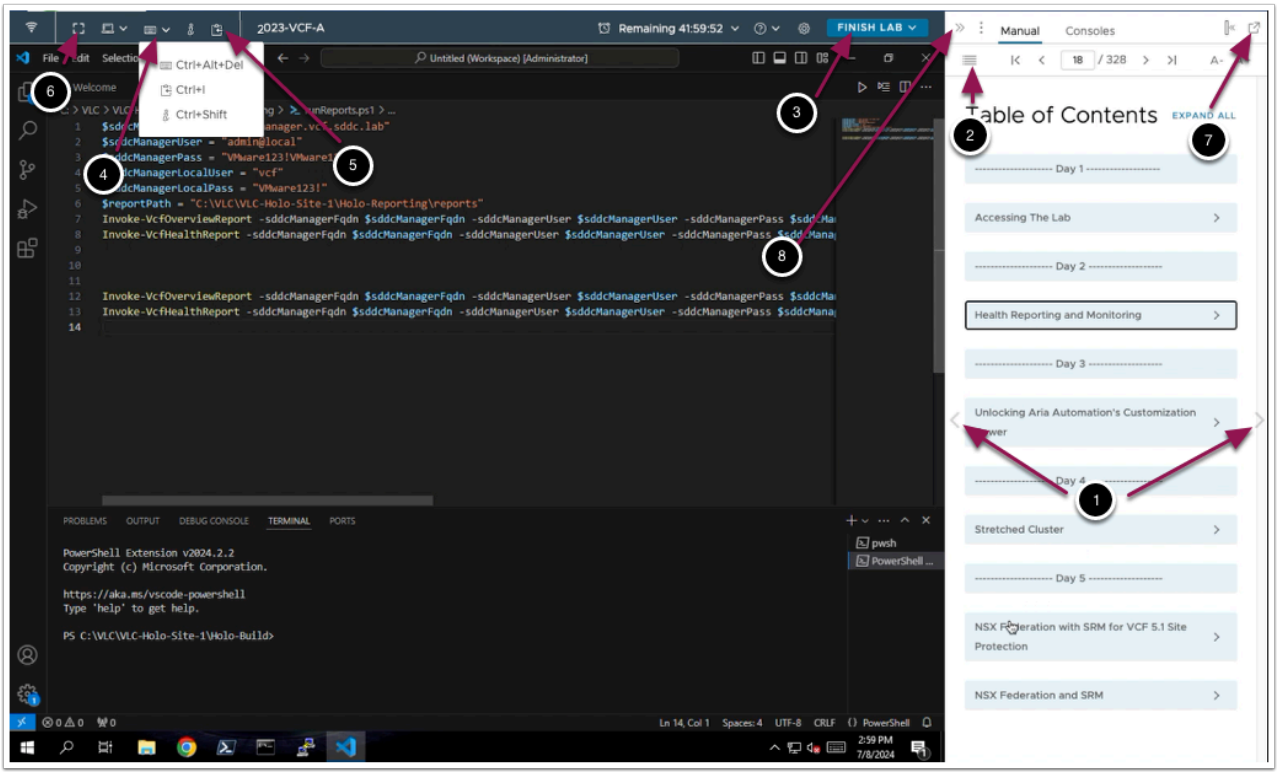
VCF DevReady with Tanzu

Table of Contents

DevReady with Tanzu - TDD - Lab Manual

Lab Guidance

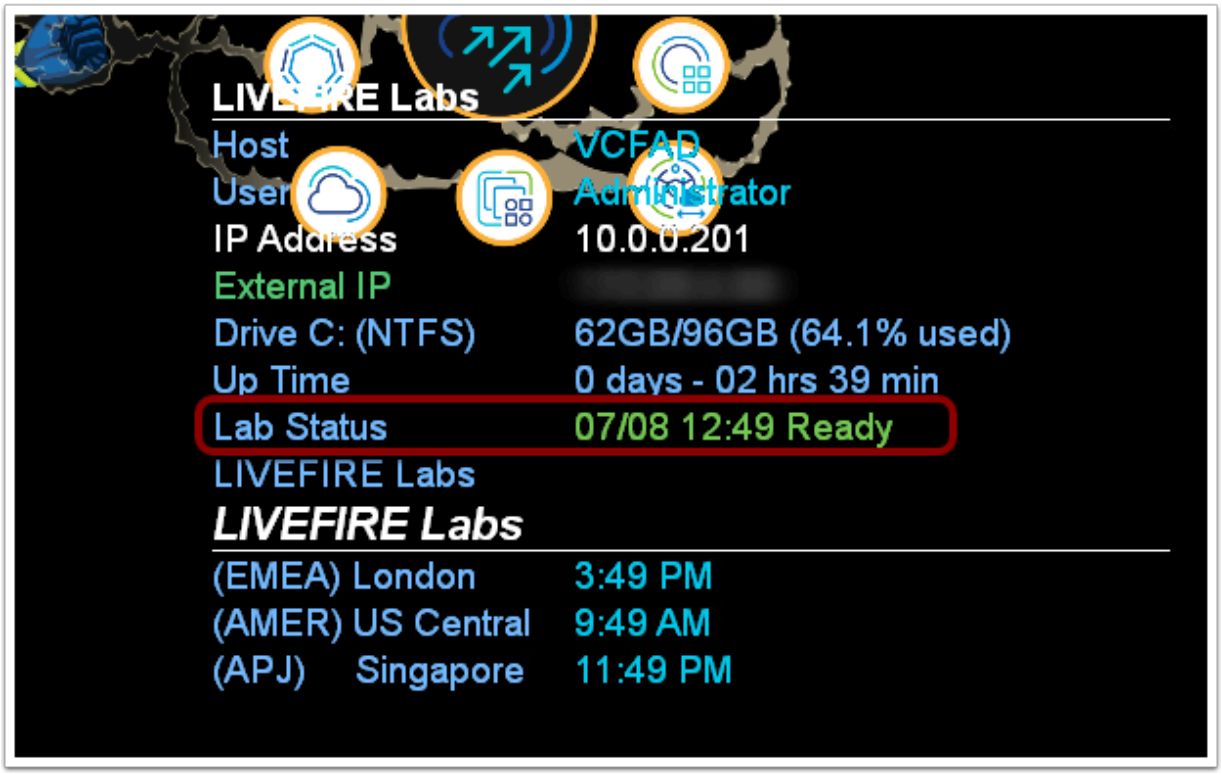
Welcome! To start somewhere other than the beginning, use the Table of Contents in the upper right-hand corner of the Lab Manual or click on one of the modules below.



From here you can:

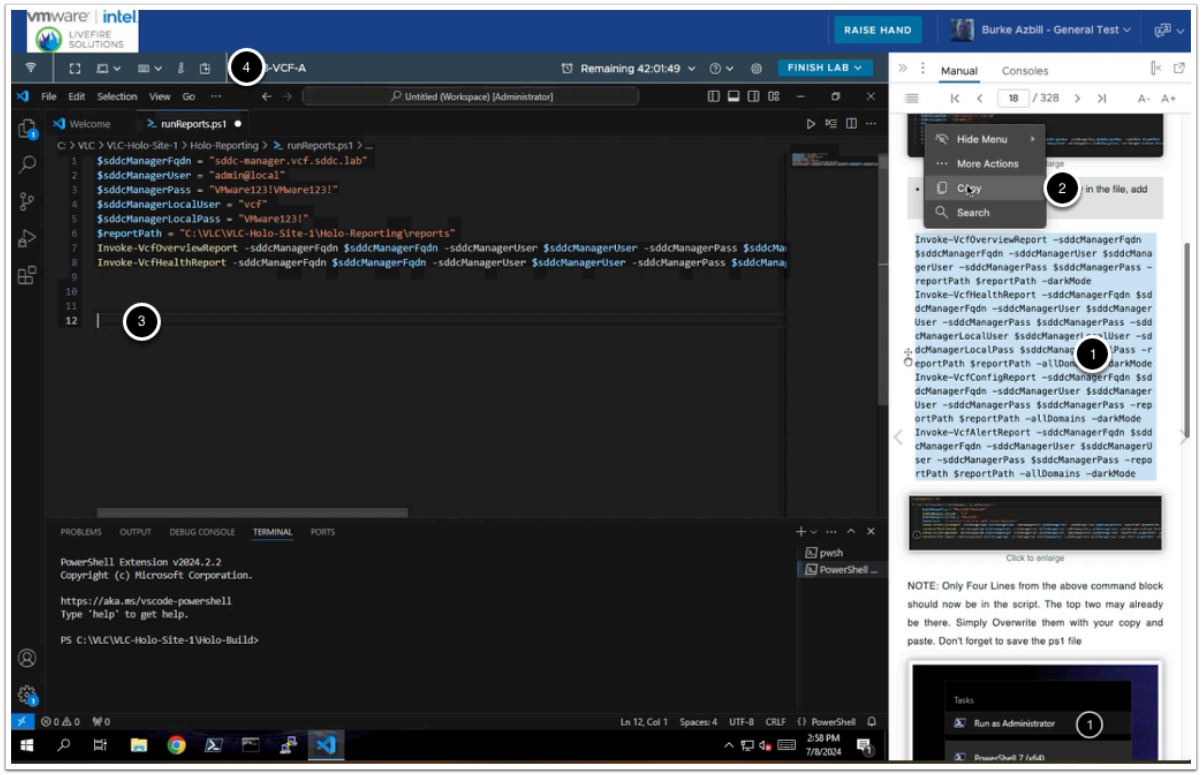
1. Click to advance to the next page and continue with the next lab module
2. Open the **TABLE OF CONTENTS** to jump to any module or lesson in this lab manual
3. Exit your lab and come back and resume it again in the future
4. Send Ctrl+Alt+Del to the console
5. Send Text into (Paste) the console
6. Enter Full-Screen
7. Detach the Lab Manual
8. Collapse the Lab Manual

You are ready....is your lab?



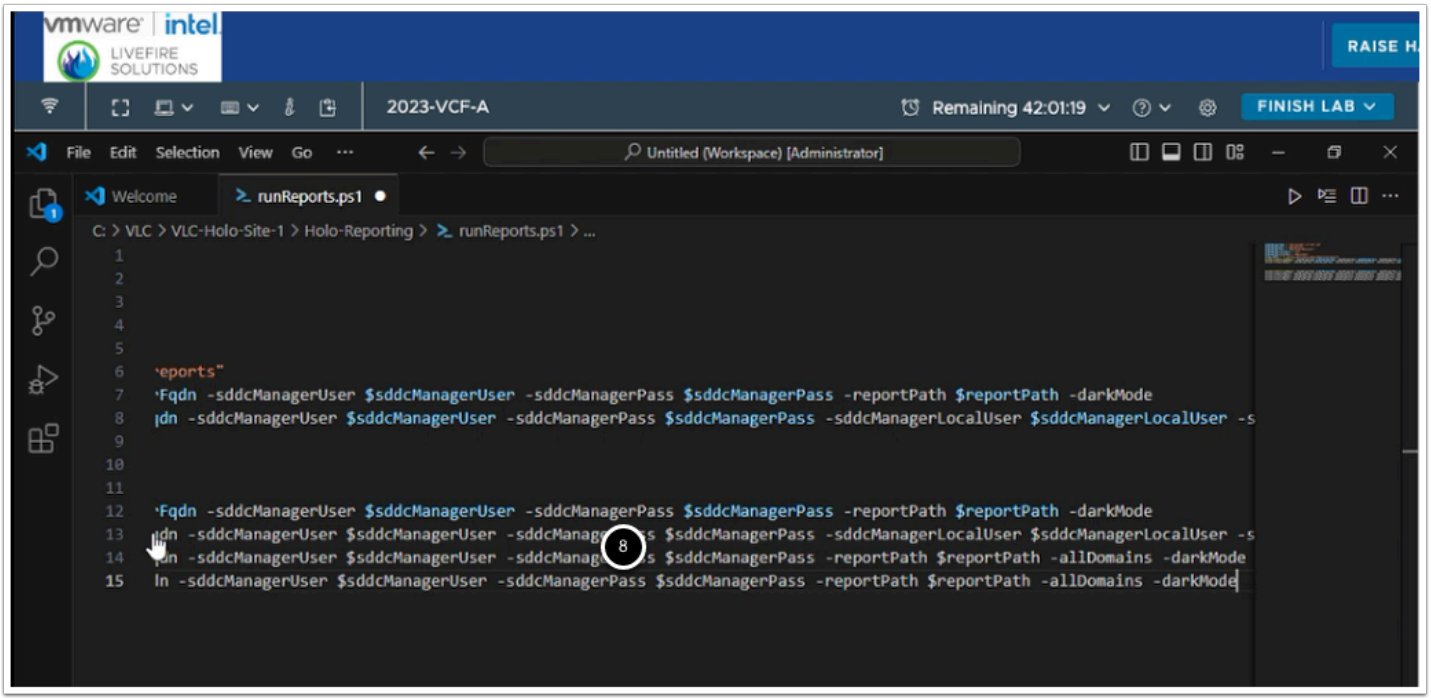
The lab console will indicate when your lab has finished all the startup routines and is ready for you to start. If you see anything other than "Ready", please wait for the status to update. If after 5 minutes your lab has not changed to "Ready", please ask for assistance.

How to Copy from manual and Paste into Console



1. Use your mouse to select the text you wish to copy
2. When you release the mouse button to complete your selection, a context menu shows up - Click Copy
   NOTE: Right click on selected text DOES NOT bring up a copy option and CTRL+C/CMD+C does not work
3. Place your cursor/prompt at the location you wish to paste inside the console
4. Click the Clipboard icon to bring up the "Send text to console" window

5. Paste the contents of your clipboard into the window using Right-Click -> Paste, or your OS Keyboard combination of either CTRL+V or CMD+V
6. Click the "SEND" button to send the text to the console
7. Click the "CANCEL" button when you are done sending text to the console. This will close the popup window



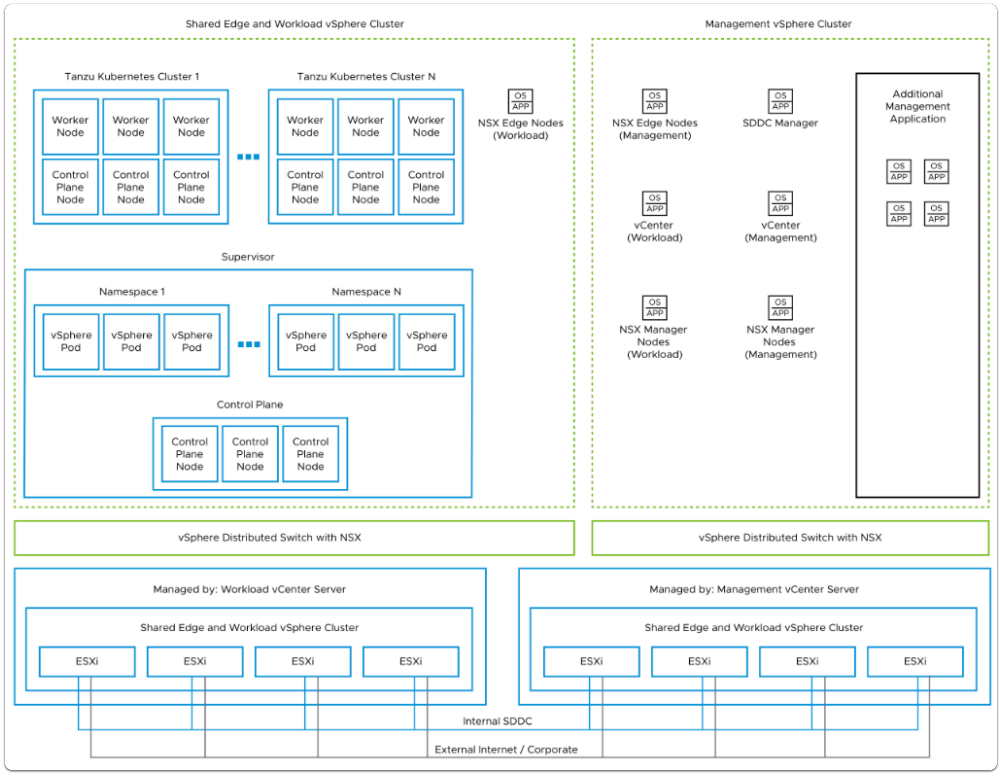
8. Confirm that your pasted text is at the desired location

Pre-Deployment considerations: Sites, Network, Storage

Pre-Deployment considerations: Sites, Network, Storage, Permissions

Taken Directly from the ***VMware Validated Solution:*** **Network Design for Developer Ready Infrastructure for VMware Cloud Foundation** found at

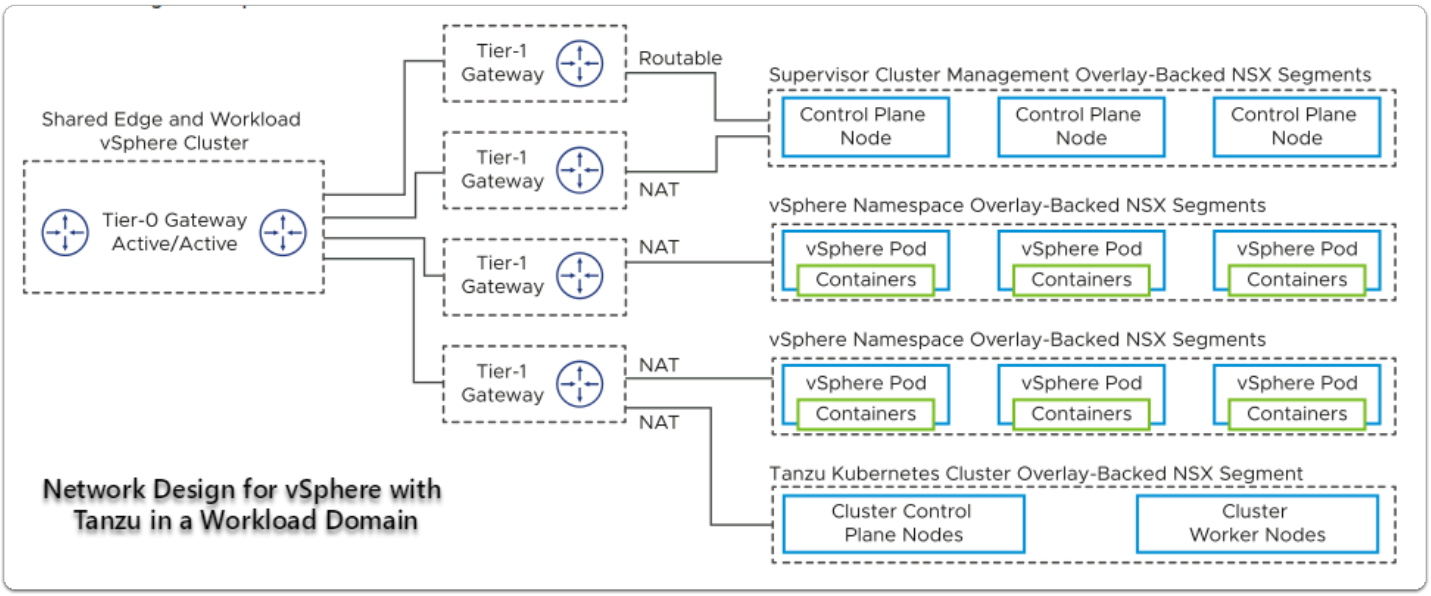
https://docs.vmware.com/en/VMware-Cloud-Foundation/services/vcf-developer-ready-infrastructure-v1/GUID-AF178A31-D09A-4265-89FD-5987D1B36757.html



NETWORK Considerations:

vSphere with Tanzu requires multiple networks. This section discusses networking design not covered in the NSX detailed design.

You deploy all vSphere with Tanzu workloads to overlay-backed NSX segments. NSX Edge nodes in the shared edge and workload vSphere cluster are deployed to VLAN-backed portgroups.



Networks Used by vSphere with Tanzu

Network	Routable/NAT	Usage
Supervisor Cluster Management Network	Routable	Used by the Supervisor control plane nodes.
Namespace Networks	NAT	When you create a namespace, a /28 overlay-backed NSX segment and corresponding IP pool is instantiated to service pods in that namespace. If that IP space runs out, an additional /28 overlay-backed NSX segment and IP pool are instantiated.
Service IP Pool Network	NAT	Used by Kubernetes applications that need a service IP address.
Ingress IP Pool Network	Routable	Used by NSX to create an IP pool for load balancing.
Egress IP Pool Network	Routable	Used by NSX to create an IP pool for NAT endpoint use.
Pod Networks	NAT	Used by Kubernetes pods that run in the cluster. Any

Network	Routable/NAT	Usage
		Tanzu Kubernetes Clusters instantiated in the Supervisor also use this pool.
Tanzu Kubernetes Cluster Service Pool Network	NAT	When you create a Tanzu Kubernetes cluster, an NSX Tier-1 Gateway is instantiated in NSX. On that NSX Tier-1 Gateway, a /28 overlay-backed NSX segment and IP pool is also instantiated.

Design Decisions on Networking for Developer Ready Infrastructure for VMware Cloud Foundation

Design Decision	Design Justification	Design Implication
Add a /24 overlay-backed NSX segment for use by the Supervisor control plane nodes.	Supports the Supervisor control plane nodes.	You must create the overlay-backed NSX segment.
Use a dedicated /20 subnet for pod networking.	A single /20 subnet is sufficient to meet the design requirement of 2000 pods.	Private IP space behind a NAT that you can use in multiple Supervisors.
Use a dedicated /22 subnet for services.	A single /22 subnet is sufficient to meet the design requirement of 2000 pods	Private IP space behind a NAT that you can use in multiple Supervisors.
Use a dedicated /24 or larger subnet on your corporate network for ingress endpoints.	A /24 subnet is sufficient to meet the design requirement of 2000 pods in most cases.	This subnet must be routable to the rest of the corporate network. A /24 subnet will suffice for most use cases, but you should evaluate your ingress needs prior to deployment
Use a dedicated /24 or larger subnet on your corporate network for egress endpoints.	A /24 subnet is sufficient to meet the design requirement of 2000 pods in most cases.	This subnet must be routable to the rest of the corporate network.

Design Decision	Design Justification	Design Implication
		A /24 subnet will suffice for most use cases, but you should evaluate your egress needs prior to deployment

STORAGE Considerations

You must configure a datastore with the activation requirements before activating a Supervisor. The Supervisor configuration requires the use of vSphere Storage Policy Based Management (SPBM) policies for control plane nodes, ephemeral disks, and image cache. These policies correlate to Kubernetes storage policies that can be assigned to vSphere Namespaces. These policies are consumed in a Supervisor or a Tanzu Kubernetes cluster, deployed by using Tanzu Kubernetes Grid Service in the Supervisor.

Design Decision	Design Justification	Design Implication
Create a vSphere tag and tag category, and apply the vSphere tag to the vSAN datastore in the shared edge and workload vSphere cluster in the VI workload domain.	Supervisor activation requires the use of vSphere Storage Based Policy Management (SPBM). To assign the vSAN datastore to the Supervisor, you need to create a vSphere tag and tag category to create an SPBM rule.	This must be done manually or via PowerCLI
Create a vSphere Storage Policy Based Management (SPBM) policy that specifies the vSphere tag you created for the Supervisor	When you create the SPBM policy and define the vSphere tag for the Supervisor, you can then assign that SPBM policy during Supervisor activation.	This must be done manually or via PowerCLI

PERSONAS and PERMISSION Considerations:

Personas describe types of system users, aligned with real people and their functions within the organization. You build a persona set based on your organization requirements for role-based access control.

The following is an example of personas defined by the Developer Ready Infrastructure for VMware Cloud Foundation validated solution and their equivalent access. To delegate roles and define access based on roles and responsibilities within your organizational structure, you use these personas as a baseline for defining and building a set of personas.

Example Personas for Developer Ready Infrastructure

Persona	Responsibility	Solution Component	Component Role or Group
vSphere Administrator	Full administrative access to vSphere with Tanzu infrastructure - configuring and activating Supervisors and vSphere namespaces	vCenter Server	Administrator
DevOps Engineer	Deploying vSphere Pods, VMs, and Tanzu Kubernetes clusters on existing vSphere namespaces within a Supervisor	vSphere namespace	Can edit
Auditor	Read-only access for security and compliance review	vSphere namespace	Can view

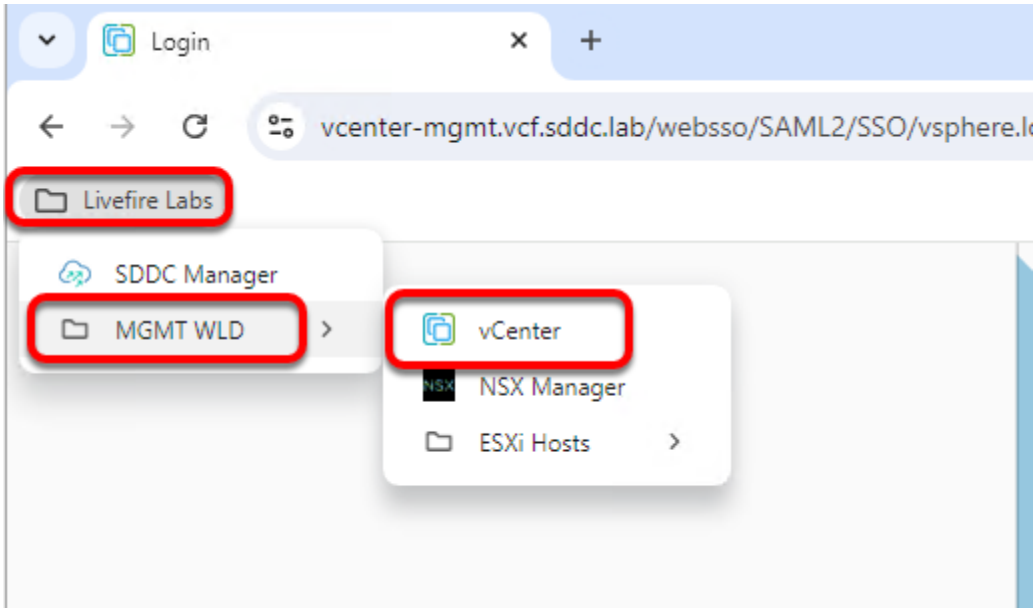
Storage: Create a Storage Policy

TASK DESCRIPTION AND OBJECTIVES

You need to create a storage policy before the Supervisor deployment. It is going to be used for Supervisor hosted Applications and Services.

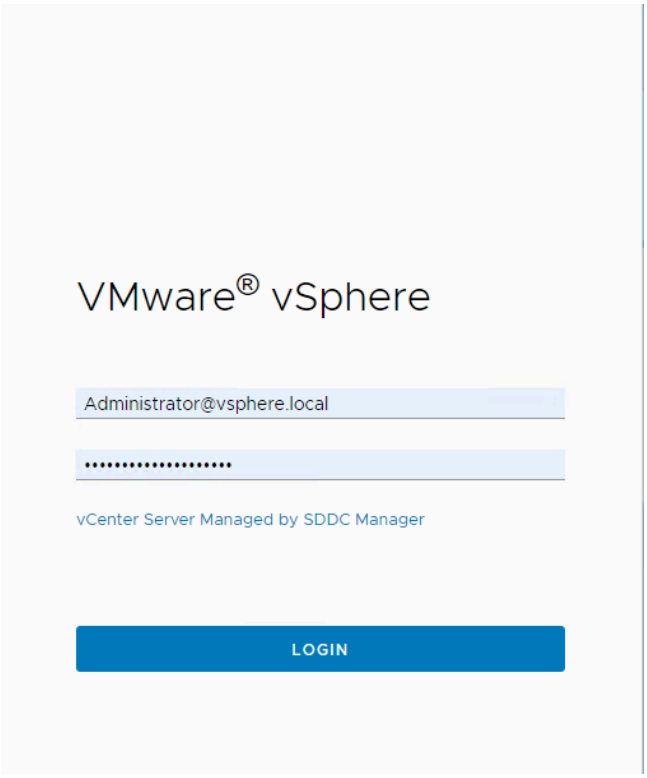
Before creating the **storage policy**, you need to create a custom **tag** and **category,** then tag your Datastore with the new tag. The policy then uses these tags as part of it's usage rule.

Open Chrome



- Open the Livefire Labs Bookmarks Folder
- Choose --> **MGMT WLD** --> **vCenter**

User and password



Login Using
User:

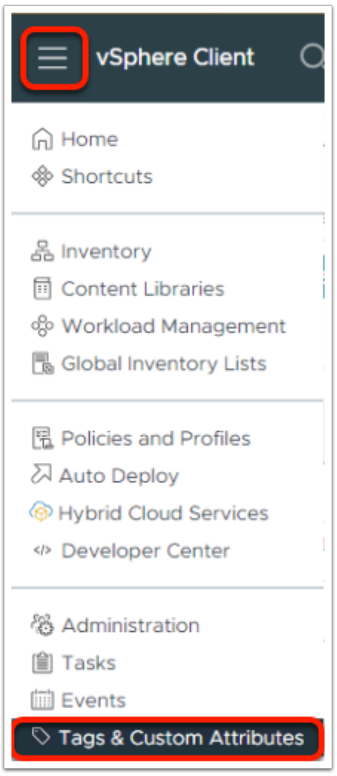
```
Administrator@vsphere.local
```

Password:

```
VMware123!VMware123!
```

Open Tags and Custom Attriburtes

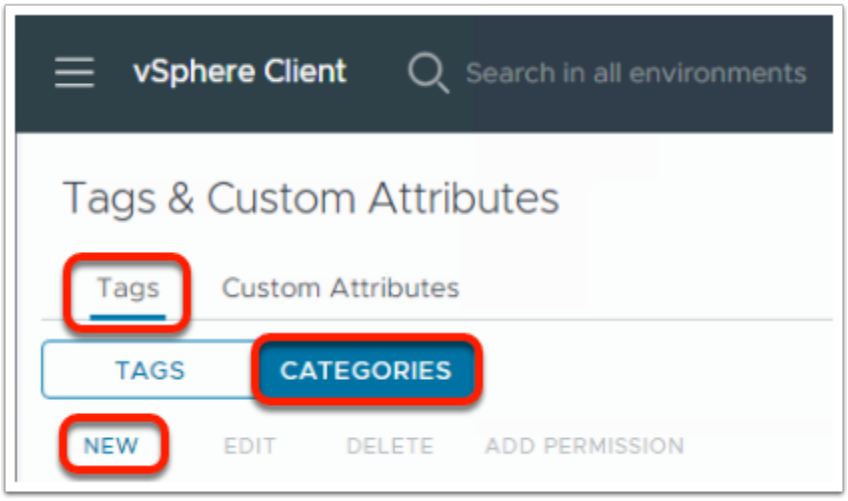
Once logged into the vSphere client UI:



Click on the hamburger (ellipsis) menu in the top left corner

Select **Tags & Custom Attributes**

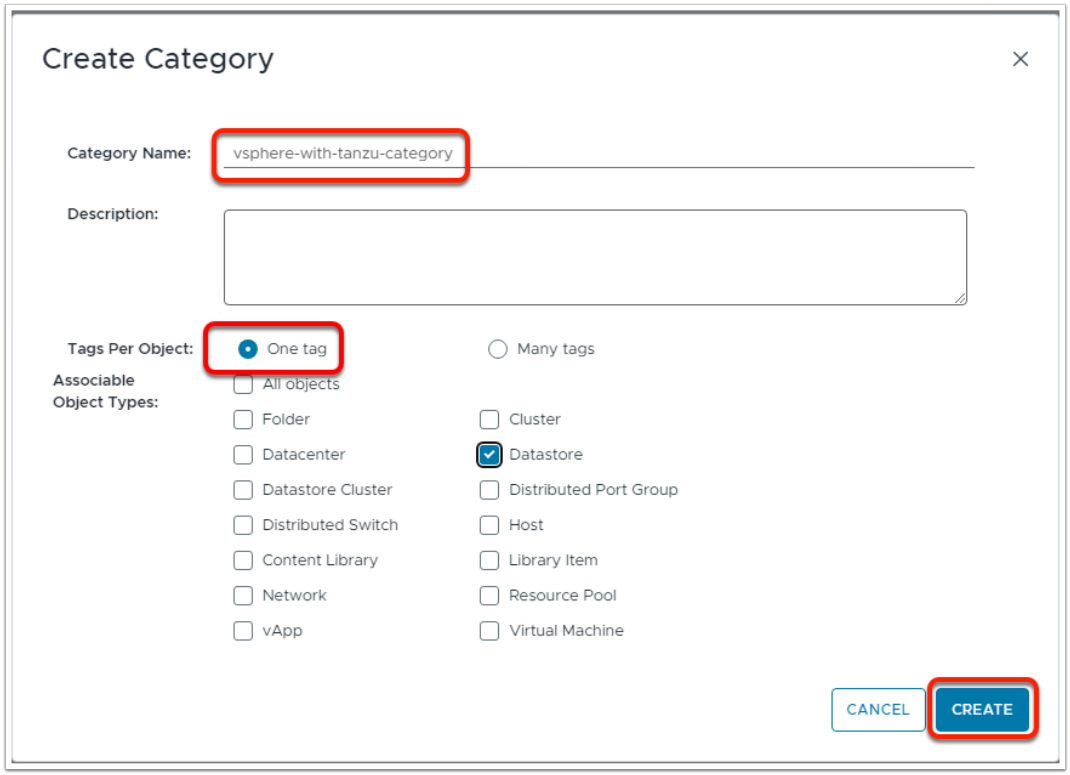
Create a new Category:



Under **Tags & Custom Attributes**,

Select **Tags**, then **CATEGORIES** --> **NEW**

Create Category



In the **Create Category** wizard enter as following:

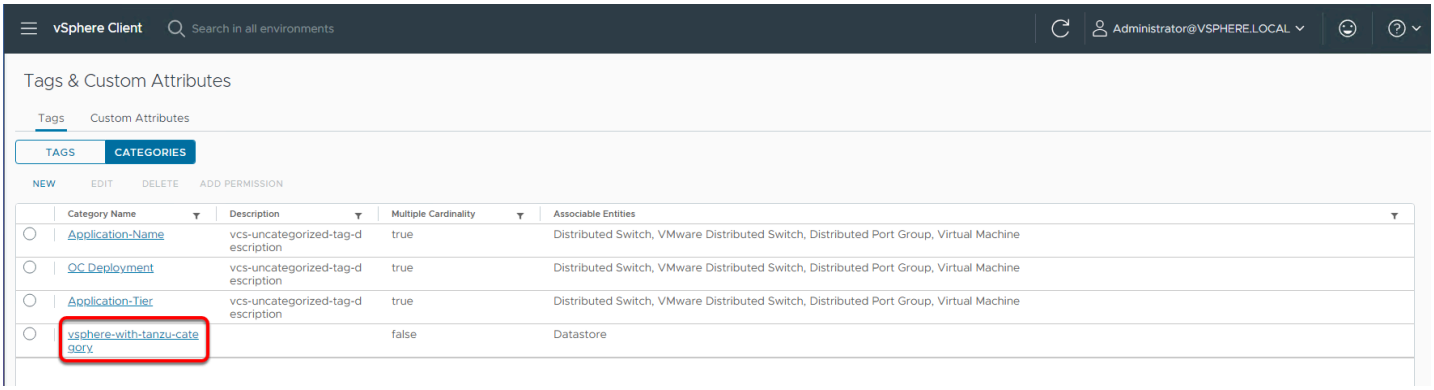
Category Name:

```
vsphere-with-tanzu-category
```

- **Tags Per Object:** One Tag
- **Associable Object Types:** Datastore

Confirm with **CREATE**

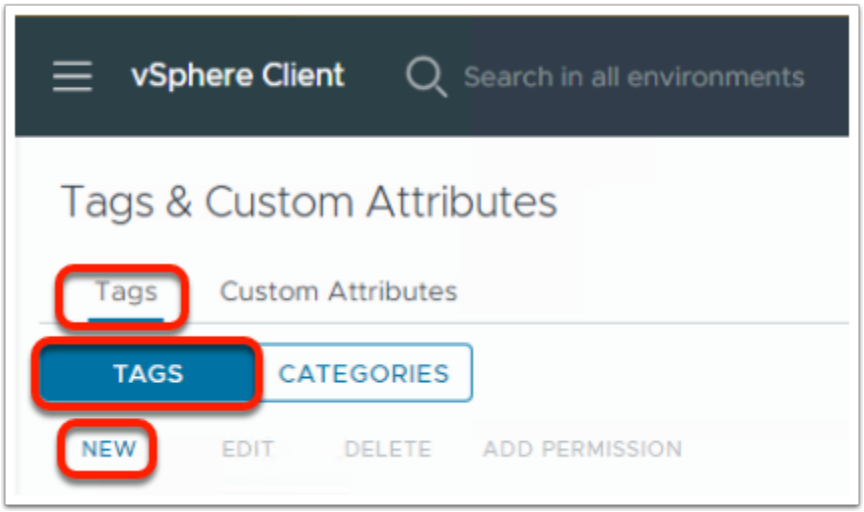
Confirm New Category



You will see your Category created.

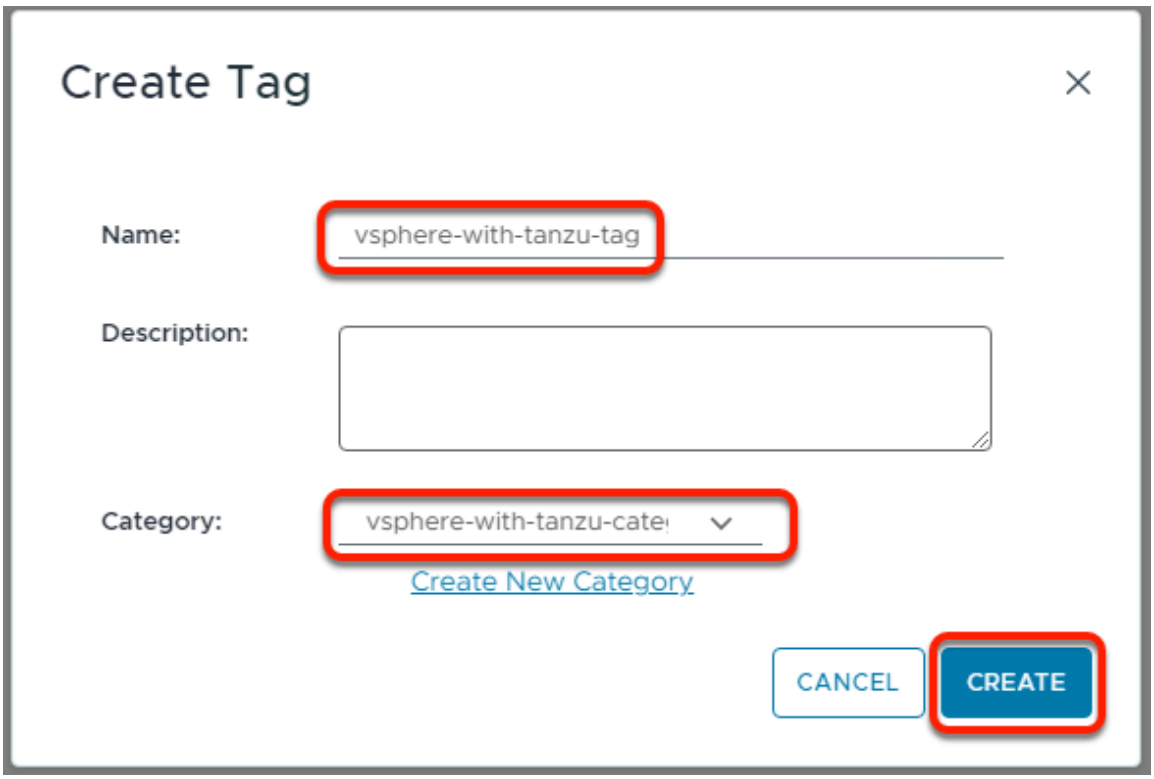
Let's continue with creating a **Tag** that is bound under this **Category**.

Create Tag



Under **Tags & Custom Attributes**,

Select **Tags**, then again **TAGS** --> **NEW**



In the **Create Tag** wizard enter as following:

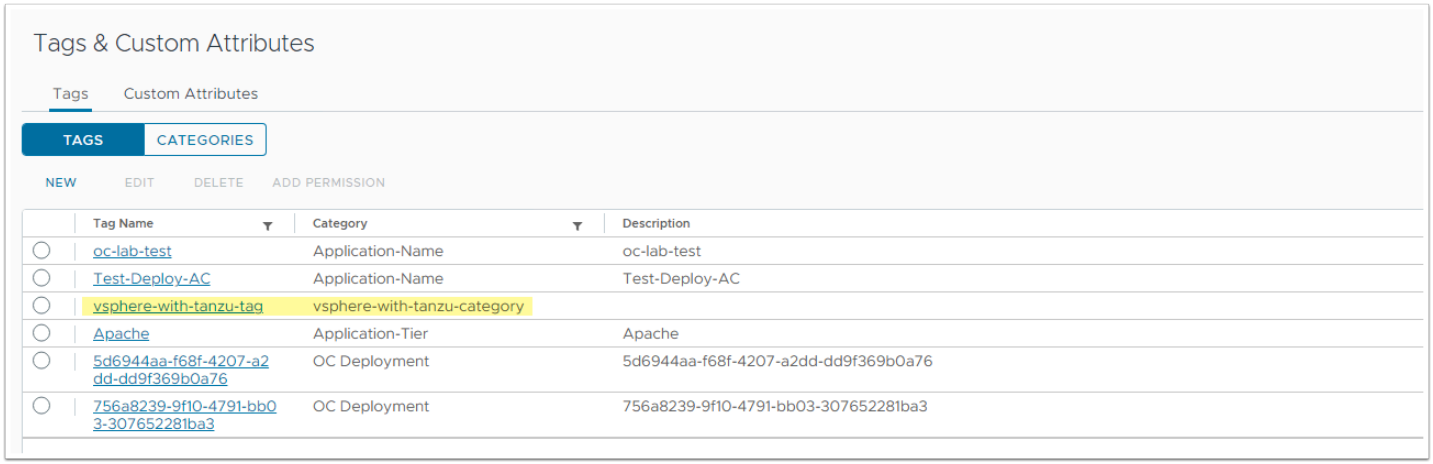
Name:

```
vsphere-with-tanzu-tag
```

Category: vsphere-with-tanzu-category (from the drop-down)

Confirm with **CREATE**

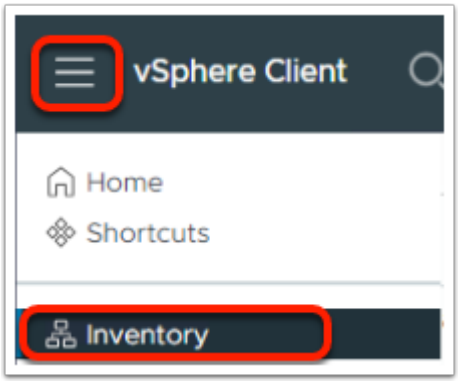
Assign Tag



You will see your Tag created and related to the specified Category you created earlier.

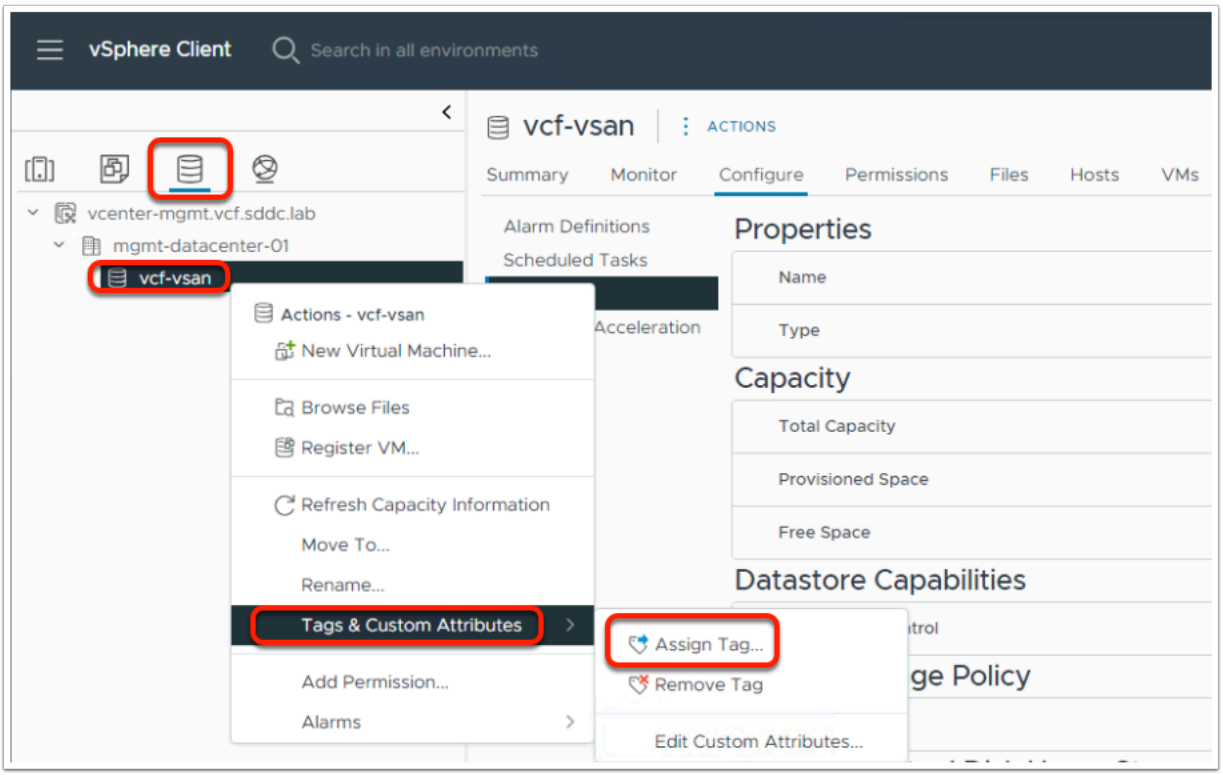
Let's continue with applying this Tag on the vSAN Datastore

Assign Tag



Click on the hamburger (ellipsis) menu in the top left corner

Select Inventory



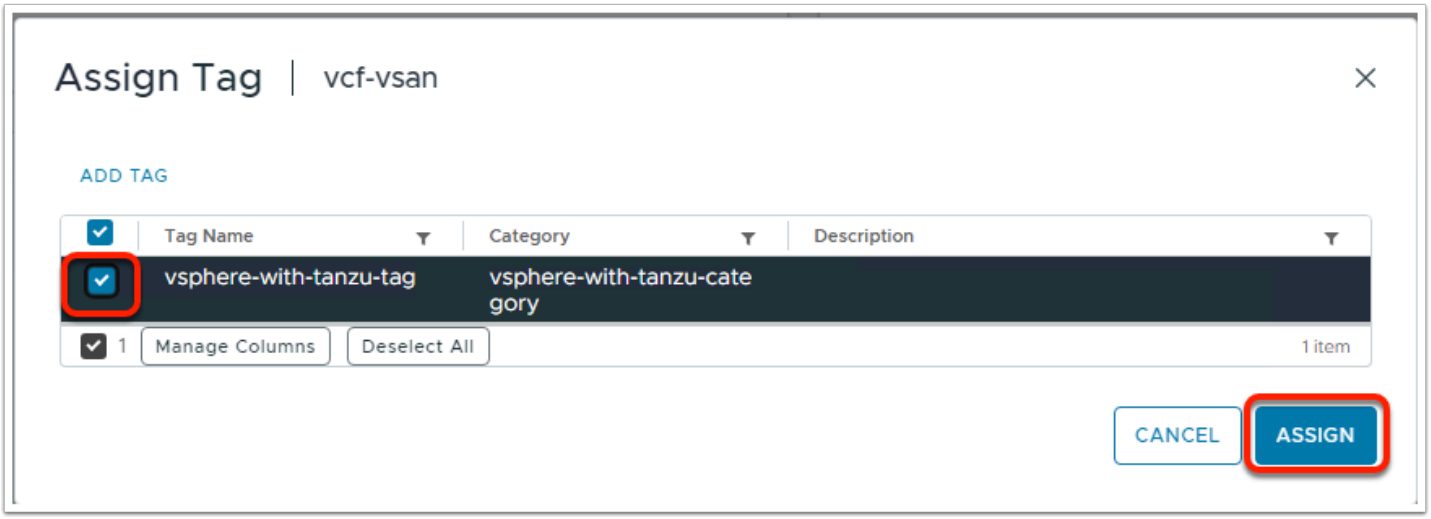
In the Inventory menu:

Select the **Storage** section, Expand the vCenter, Expand the Datacenter

Select the **vcf-vsan** Datastore, right click on it and select **Tags & Custom Attributes**

Expand it and select **Assign Tag**

Assign Tag



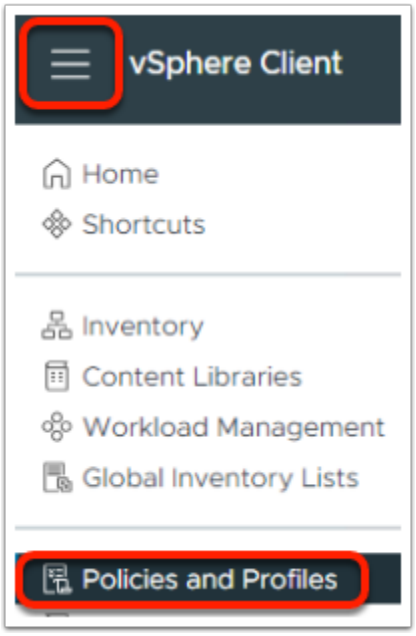
In the **Assign Tag wizard**

Select the **vsphere-with-tanzu-tag** you created earlier

Confirm with **ASSIGN**

Now you can configure a relevant **storage policy**, working with **tag based placement** of the objects, over the **vcf-vsan** Datastore.

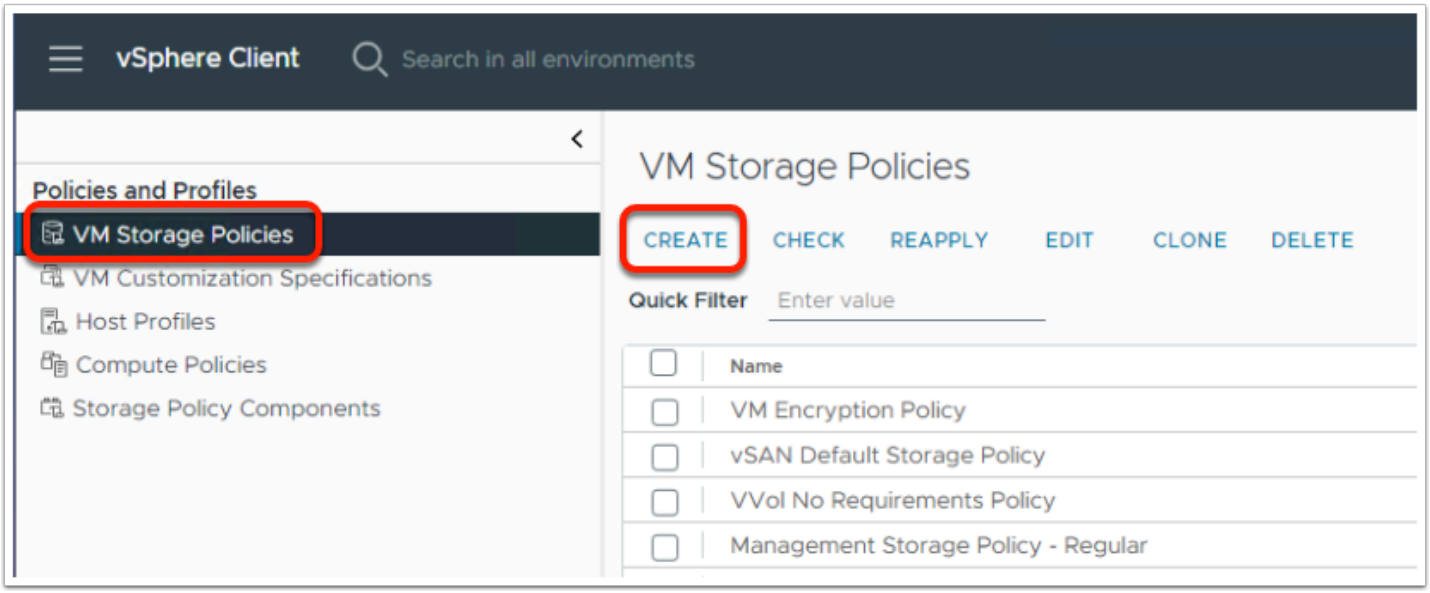
Create Storage Policy



Click on the hamburger menu in the top left corner

Select **Policies and Profiles**

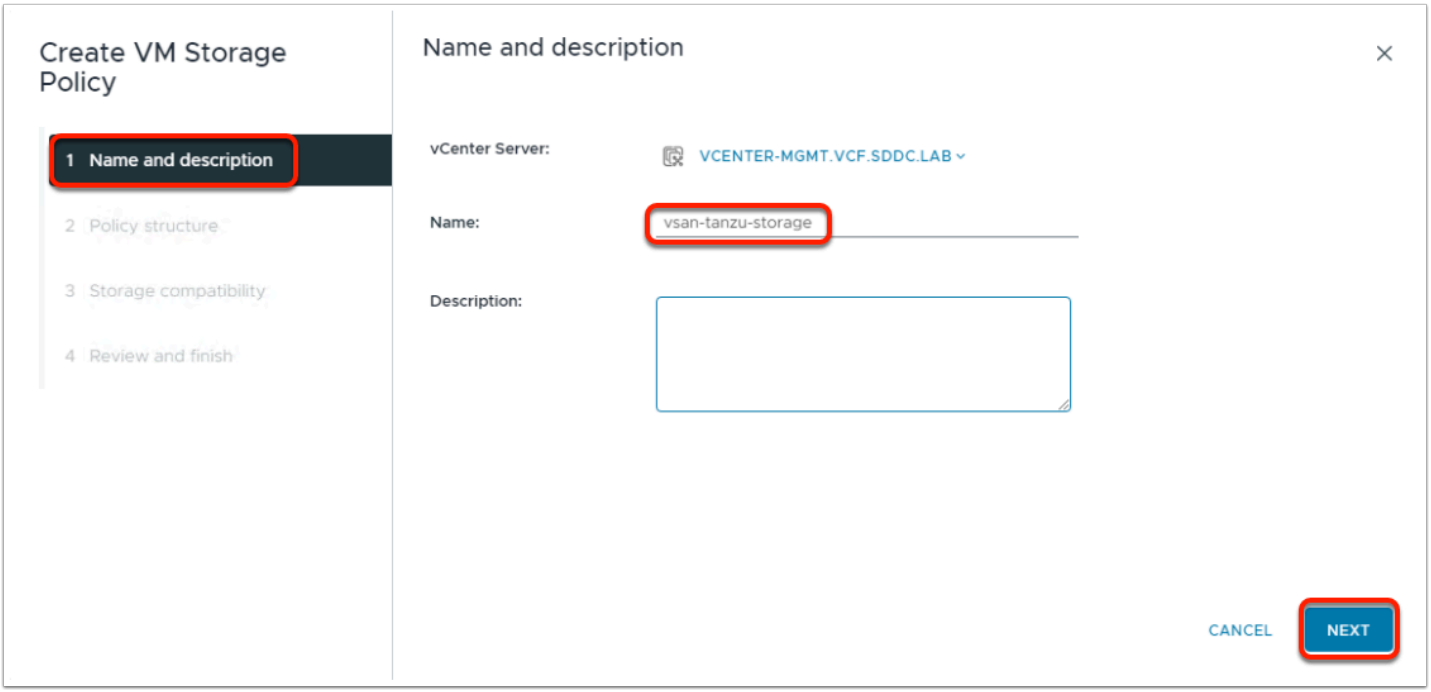
Create Storage Policy



Under **VM Storage Policies**

Select **VM Storage Policies** --> **CREATE**

Create Storage Policy



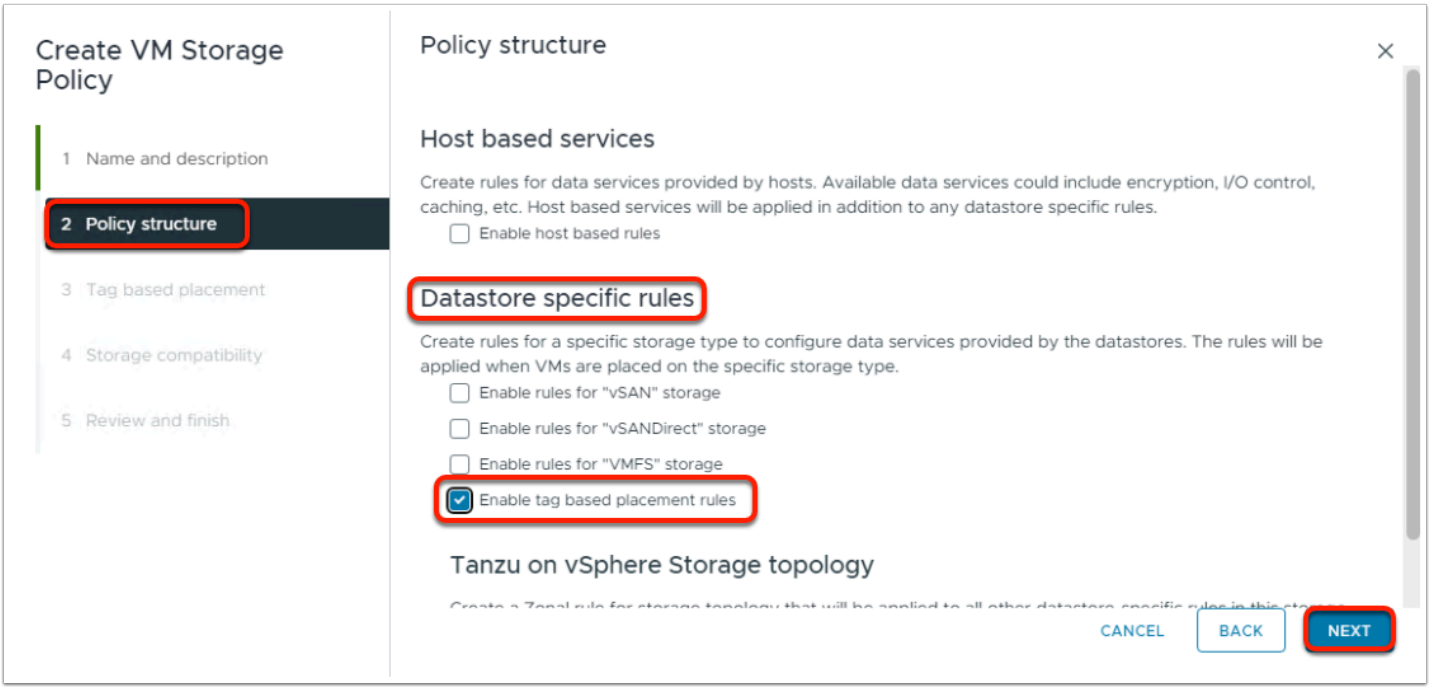
In the **Create VM Storage Policy** wizard,

For **Name and description** enter as following:

```
vsan-tanzu-storage
```

Confirm with **NEXT**

Create Storage Policy

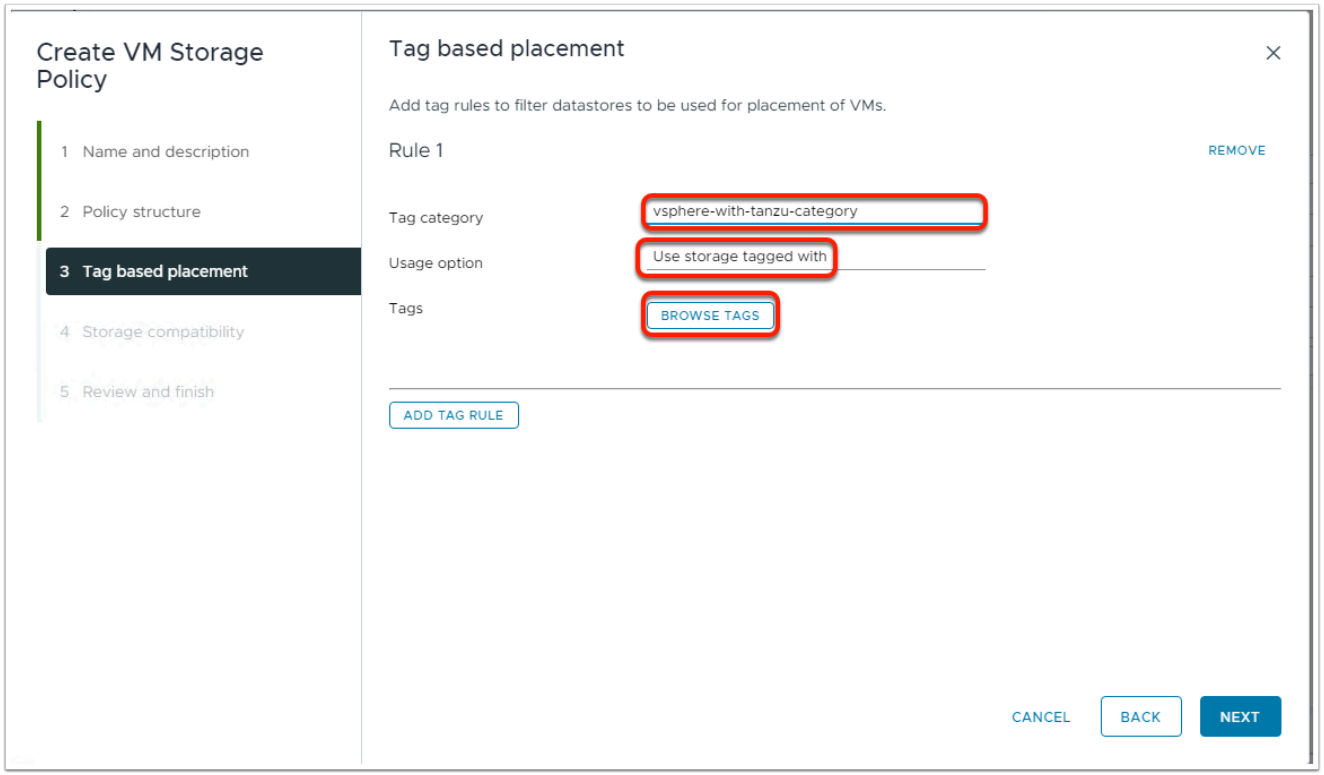


For **Policy structure**, under **Datastore specific rules**,

Select **Enable tag based placement rules**

Confirm with **NEXT**

Create Storage Policy

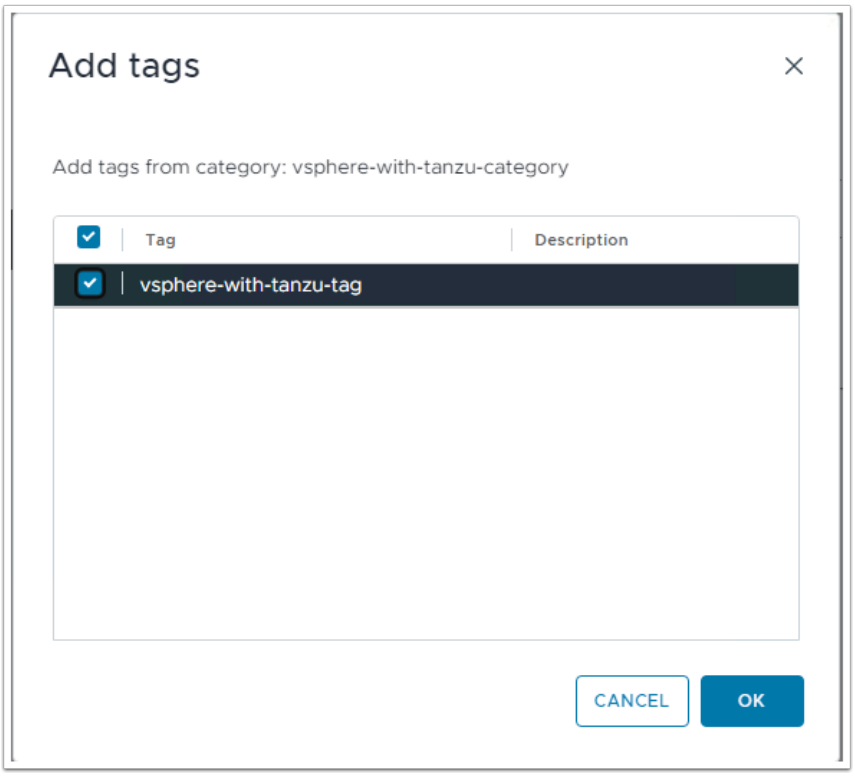


For **Tag Based Placement ,** under **Rule 1**,

Enter the following:

```
Tag category: vsphere-with-tanzu-category
Usage option: Use storage tagged with
Tags: vsphere-with-tanzu-tag
```

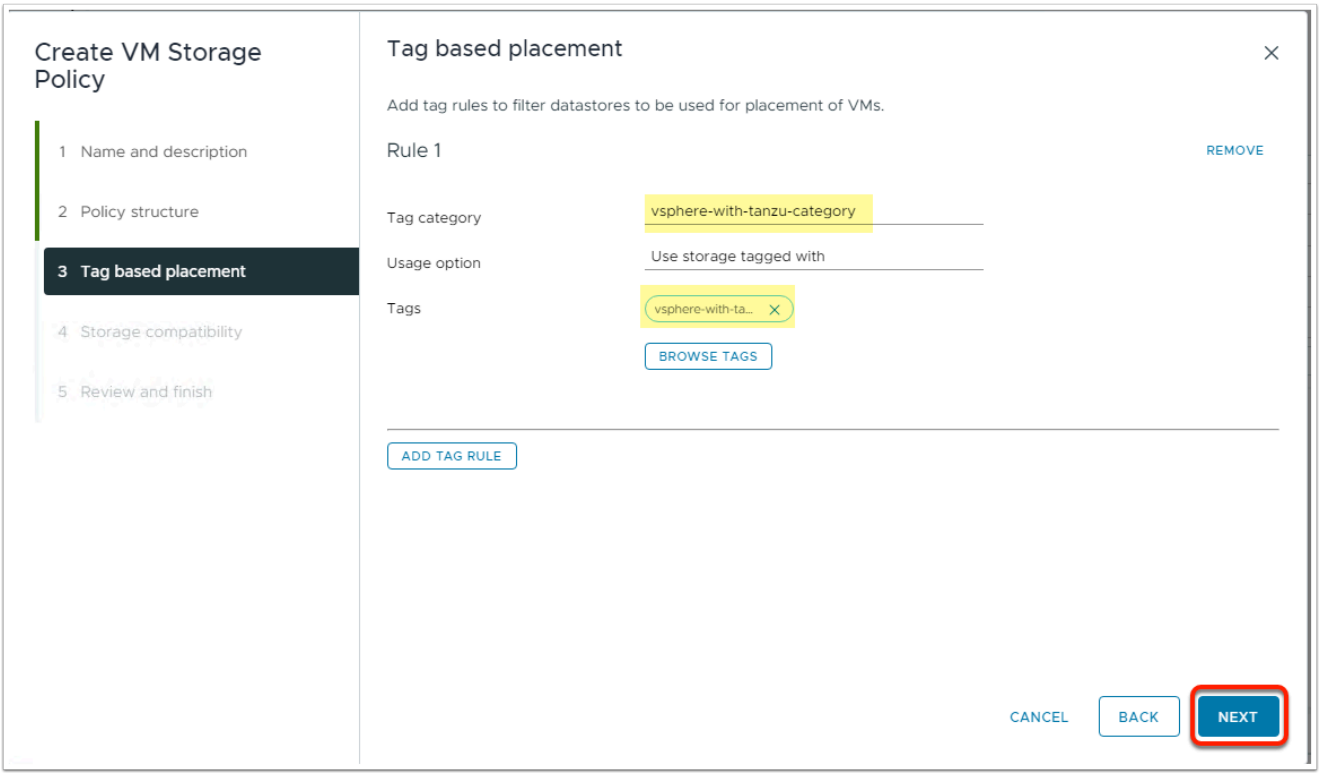
You can select the **tanzu-tag** clicking on **BROWSE TAGS**



In the **Add tags** wizzard,

Select your **vsphere-with-tanzu-tag** and confirm with **OK**

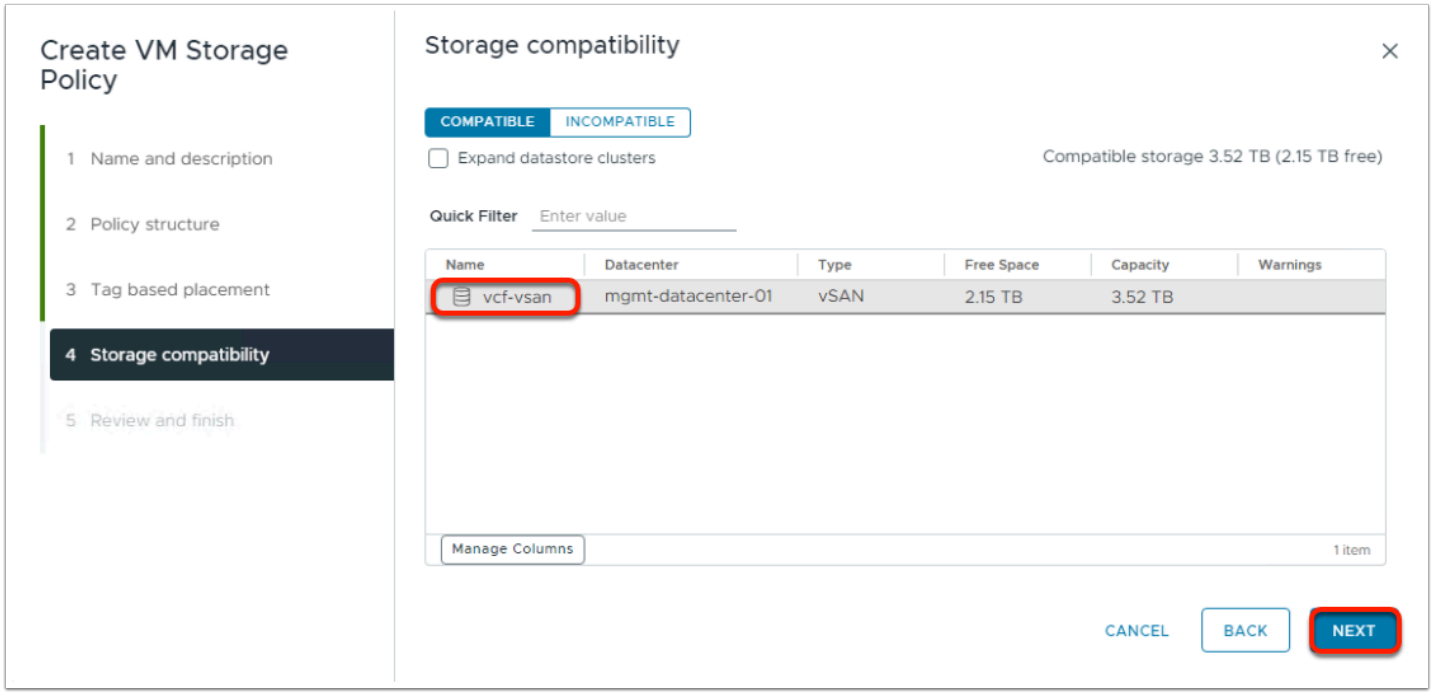
Create Storage Policy



You can see and the needed values for Tag Category and Tags are in place.

Confirm with **NEXT**

Create Storage Policy

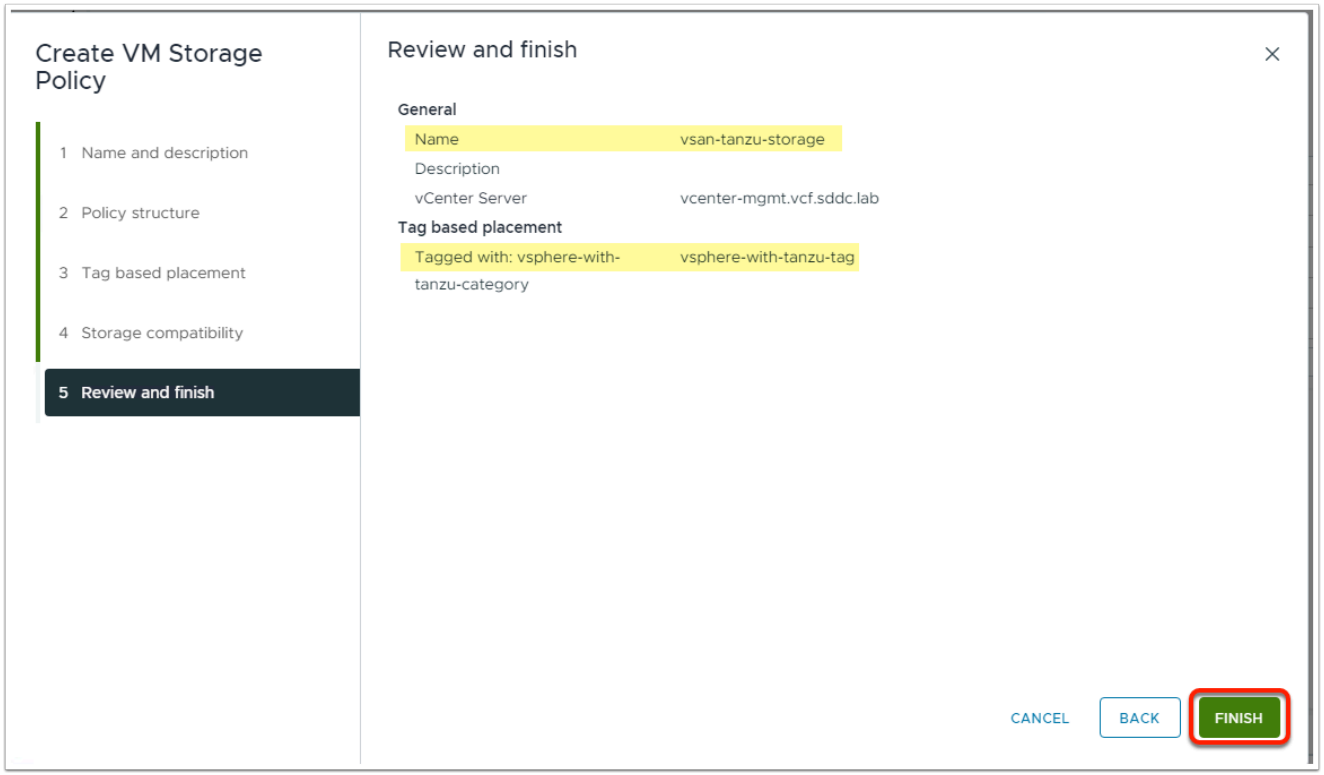


For **Storage Compatibility,** under **COMPATIBLE**,

you can see the listed **vcf-vsan** Datastore.

Confirm with **NEXT**

Create Storage Policy



For Review and finish you can see the configured properties in your new storage policy.

Confirm with **FINISH**

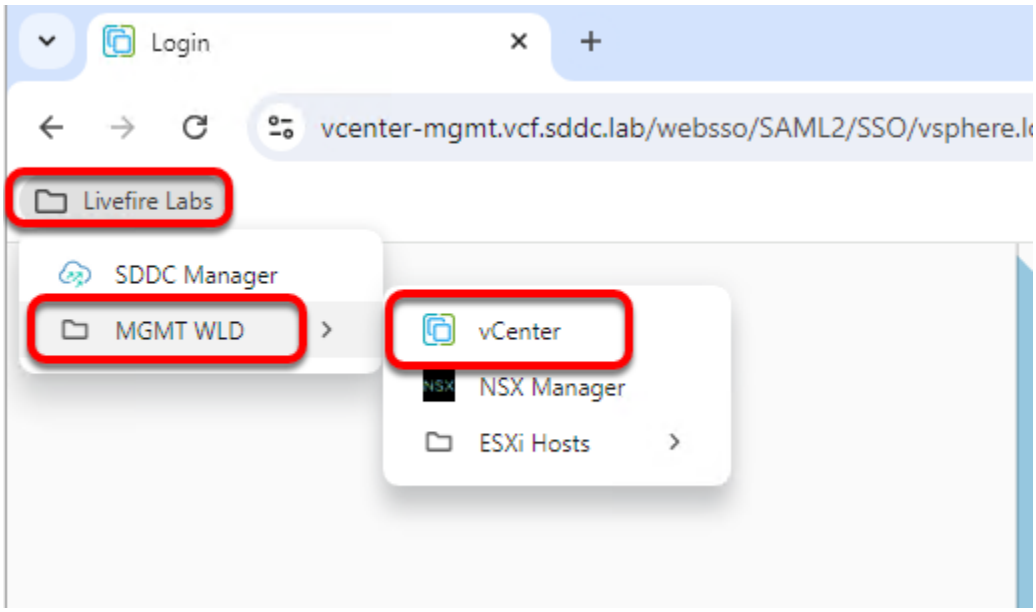
Deployment: Developer Ready Infrastructure

Deploy a Supervisor

Task description and objectives

After you meet the prerequisites, you can begin the deployment of the vSphere with Tanzu environment to support the Developer Ready Infrastructure for VMware Cloud Foundation solution. The deployment of vSphere with Tanzu involves deploying and configuring an ***Iaas Control*** Plane Supervisor Cluster.

Login to vSphere

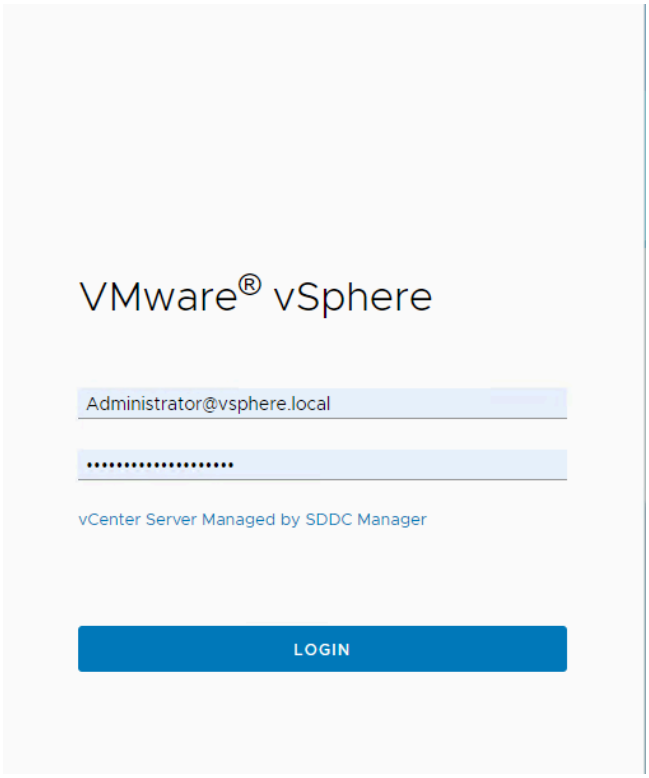


Open Chrome

Open the Livefire Labs Bookmark Folder

Choose --> **MGMT WLD** --> **vCenter**

Login to vSphere



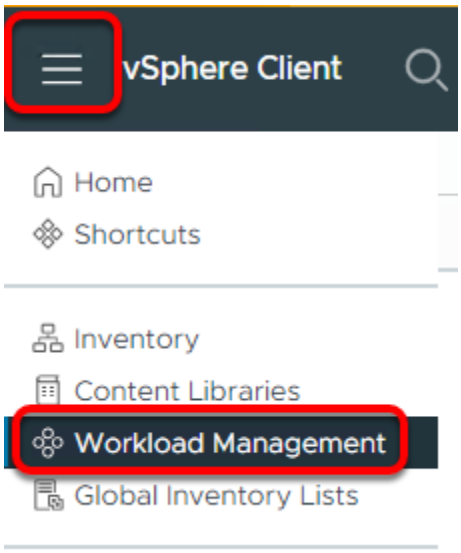
Login with User:

```
Administrator@vsphere.local
```

And Password:

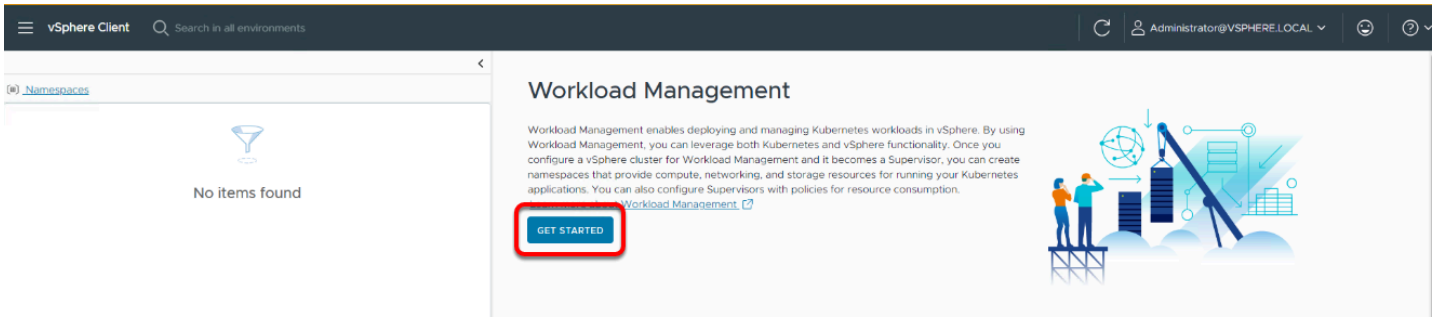
```
VMware123!VMware123!
```

Open Workload Management



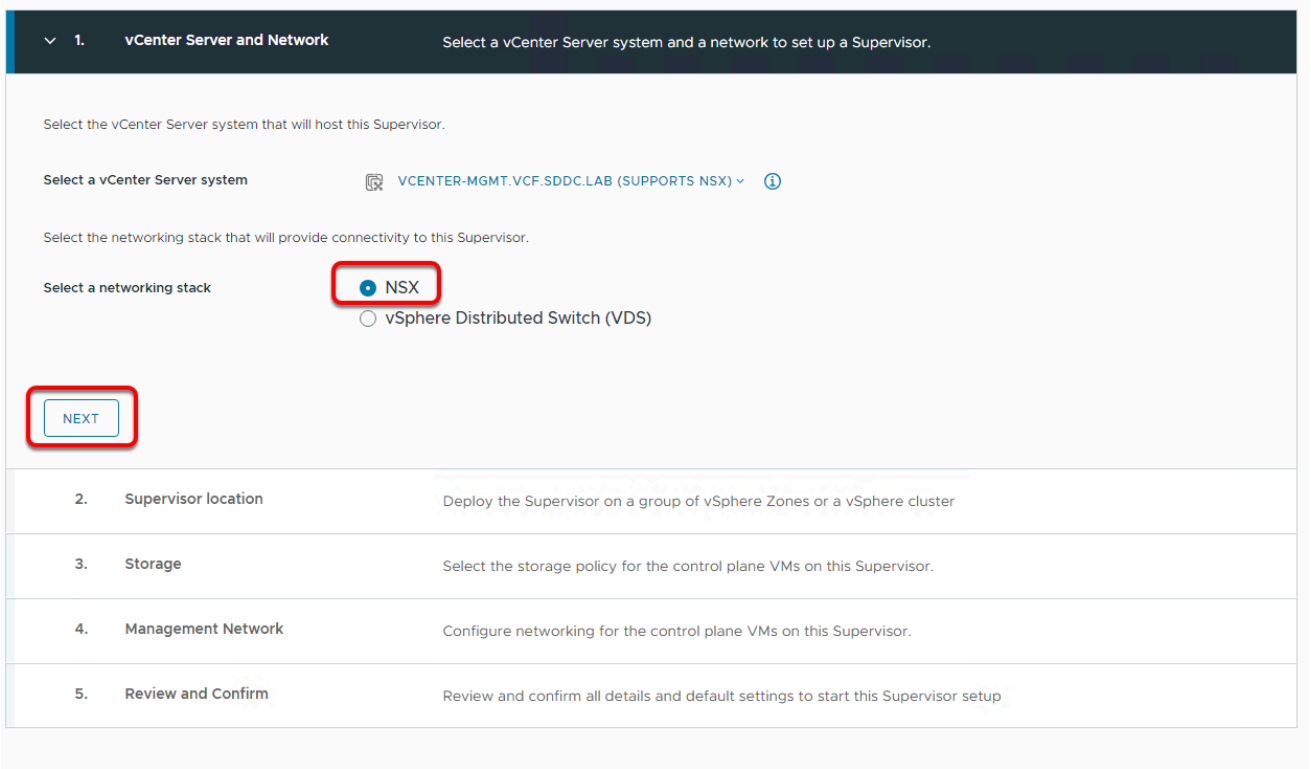
Open the ***Hamberger*** Menu

Choose **Workload Management**



Click **GET STARTED**

vCenter Server and Network

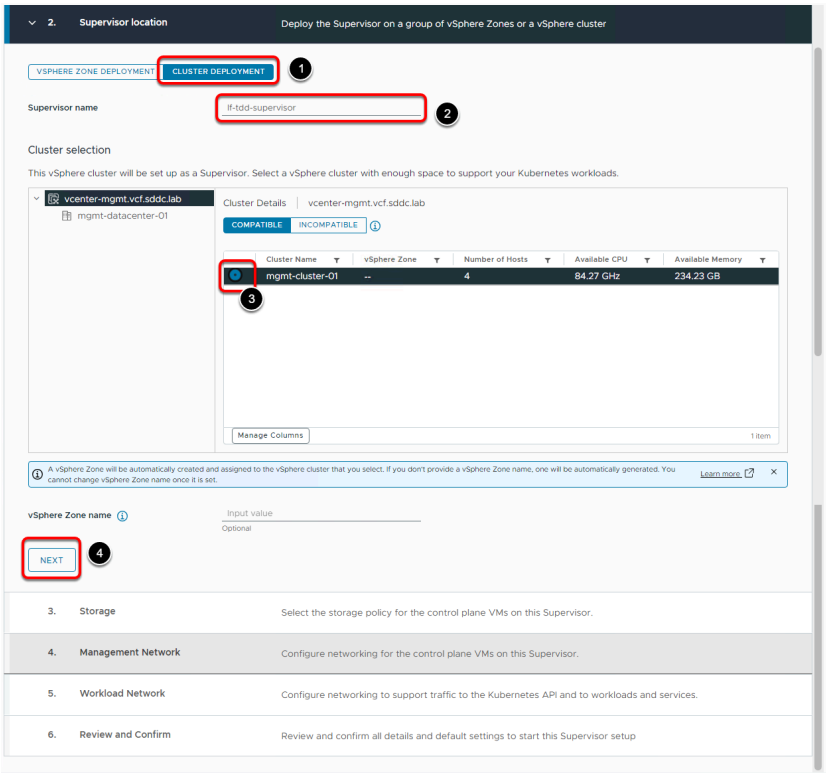


Make Sure the **NSX** Radio button is chosen

Click **NEXT**

No idea why the UI, on occasion, seems to post this step again after you click **NEXT**

Supervisor Location



1. Click **CLUSTER DEPLOYMENT**

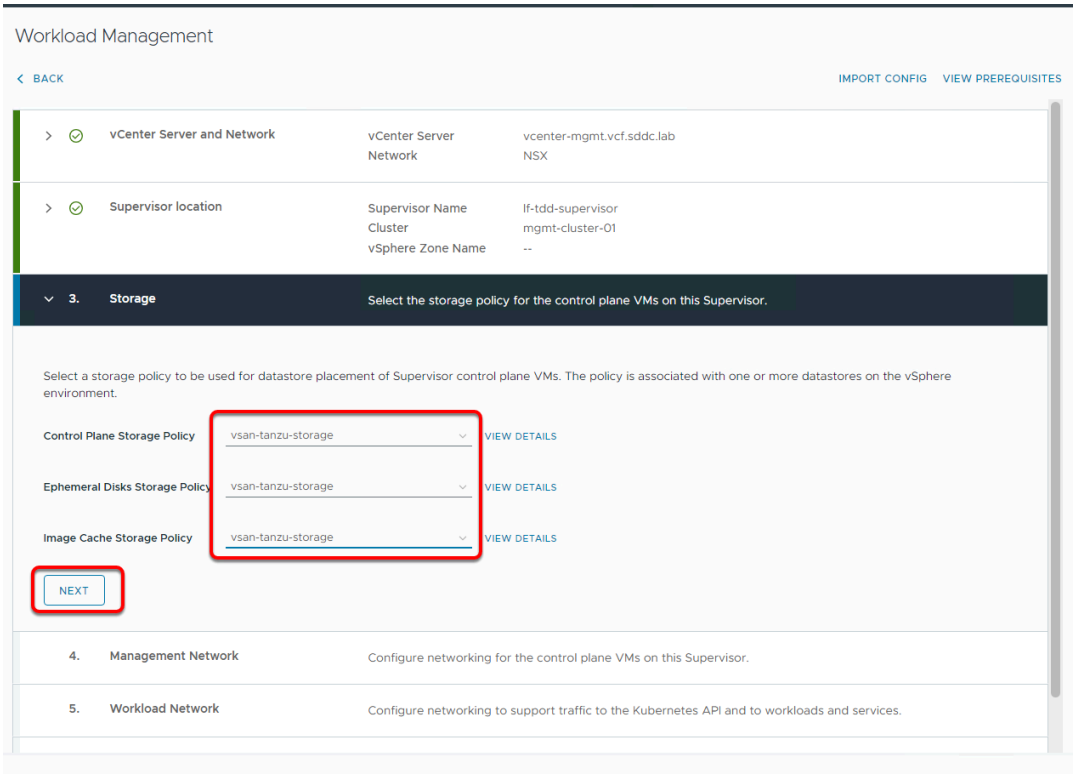
2. Give the Supervisor Cluster a name, i.e.:

```
lf-tdd-supervisor
```

3. Click the Radio button for the one VCF Cluster

4. Click **NEXT**

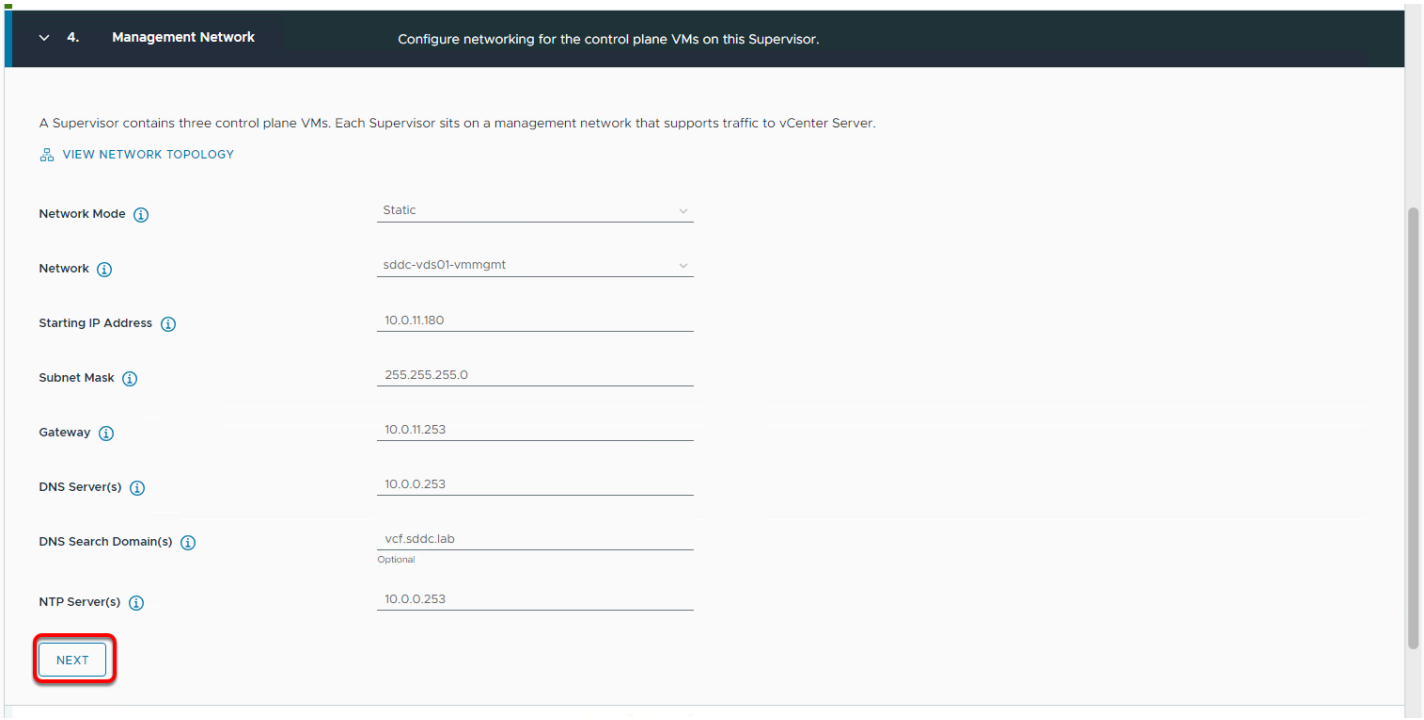
Storage



Choose the single Storage Policy *vsan-tanzu-storage* for each Storage Policy setting

Click **NEXT**

Management Network



Network Model: Static

Network: sddc-vds01-vmmgmt

Starting IP Address:

```
10.0.11.180
```

Subnet Mask:

```
255.255.255.0
```

Gateway:

```
10.0.11.253
```

DNS:

```
10.0.0.253
```

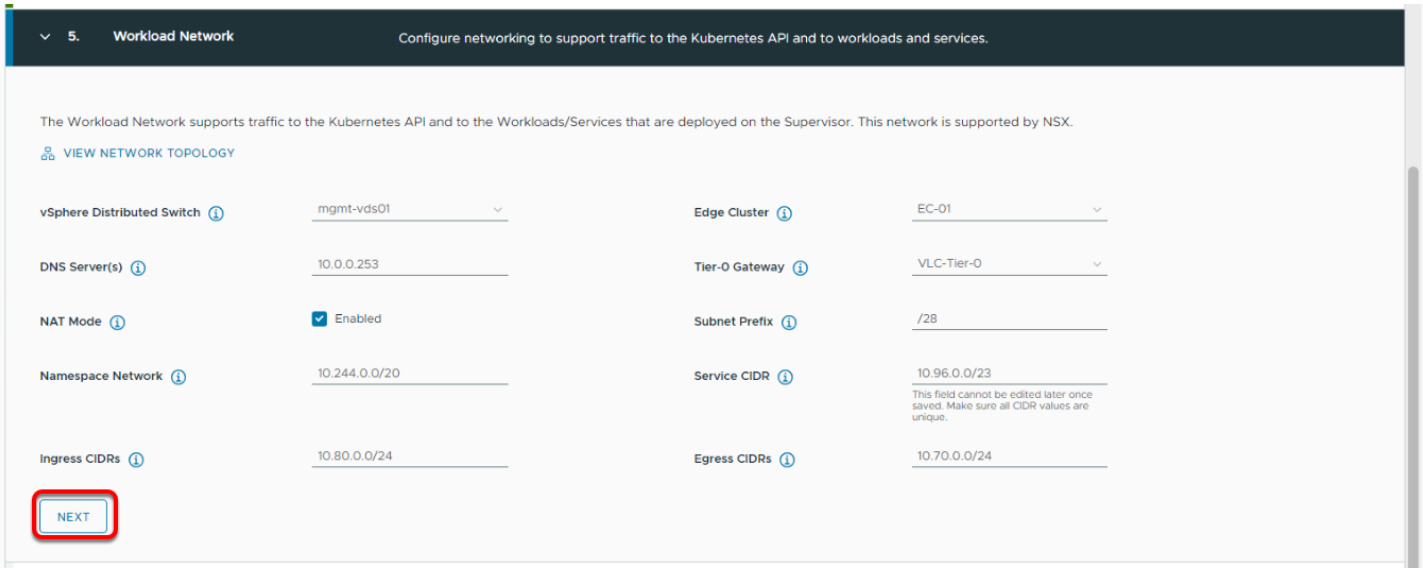
DNS Search Domain(s):

```
vcf.sddc.lab
```

NTP:

```
10.0.0.253
```

Workload Network



vSphere Distributed Switch:

```
mgmt-vds01
```

DNS Server(s):

```
10.0.0.253
```

NAT Mode:

```
Enabled
```

Edge Cluster:

```
EC-01
```

Tier-0 Gateway:

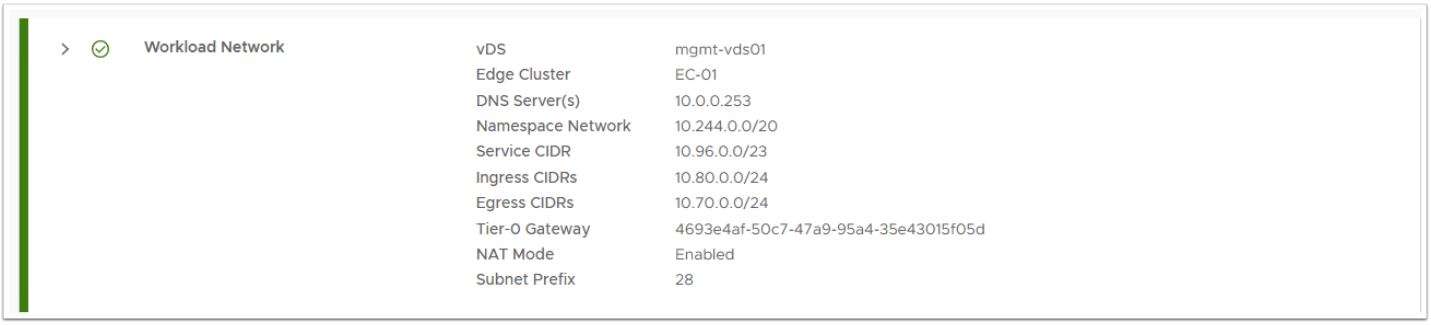
```
VLC-Tier-0
```

Ingress CIDRs:

```
10.80.0.0/24
```

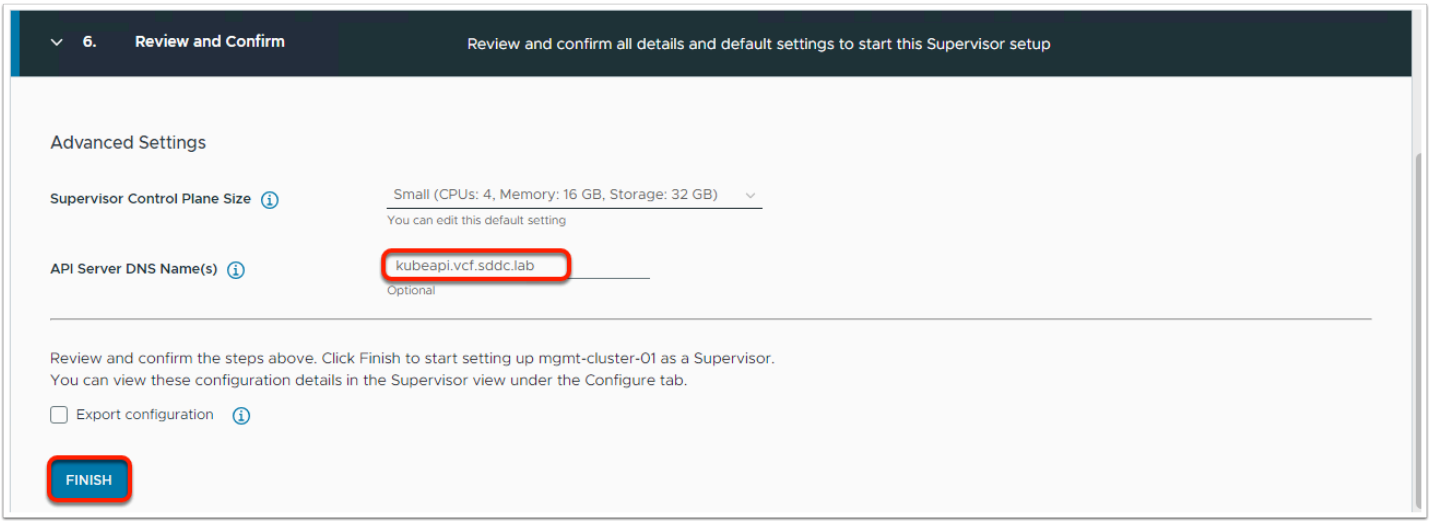
Egress CIDRs:

```
10.70.0.0/24
```



Click **NEXT**

Review and Confirm



Supervisor Control Plane Size:

```
Small
```

API Server DNS Name(s):

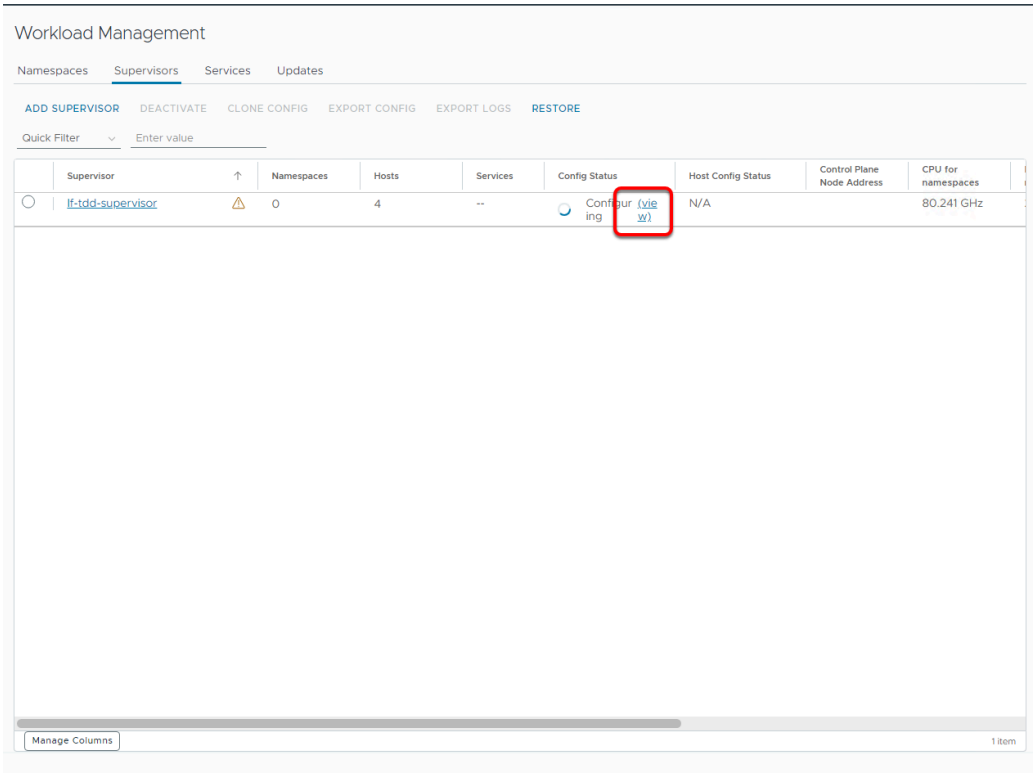
```
kubeapi.vcf.sddc.lab
```

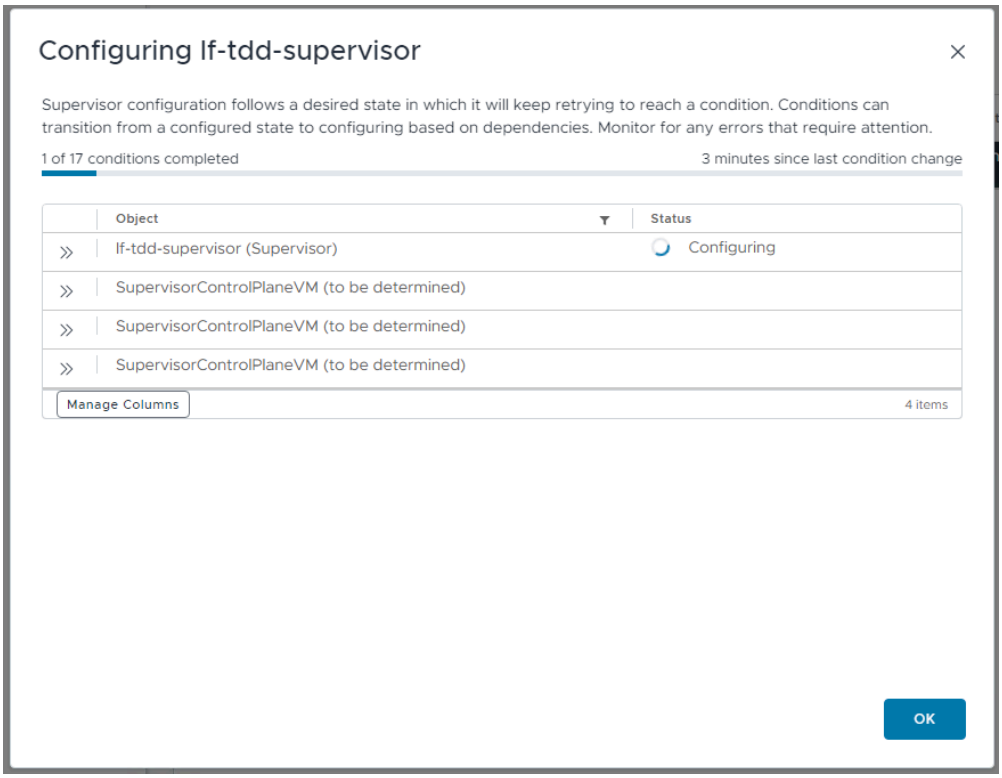
Click **FINISH**

Track Status

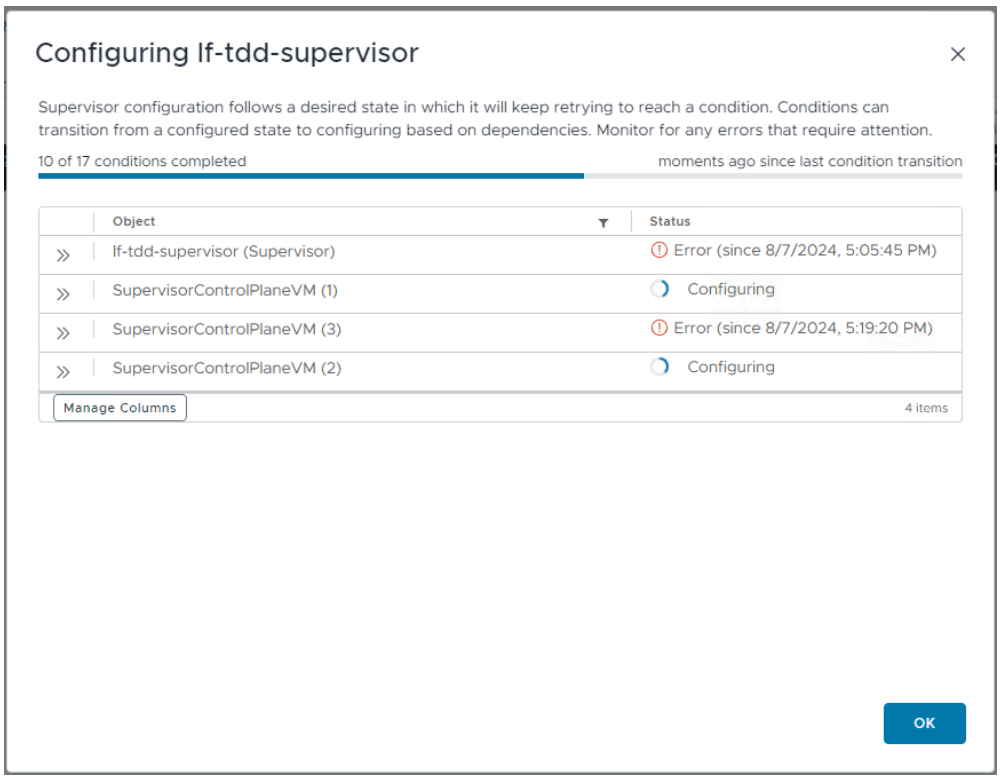
As the Supervisor deploys there are various places to track progress.

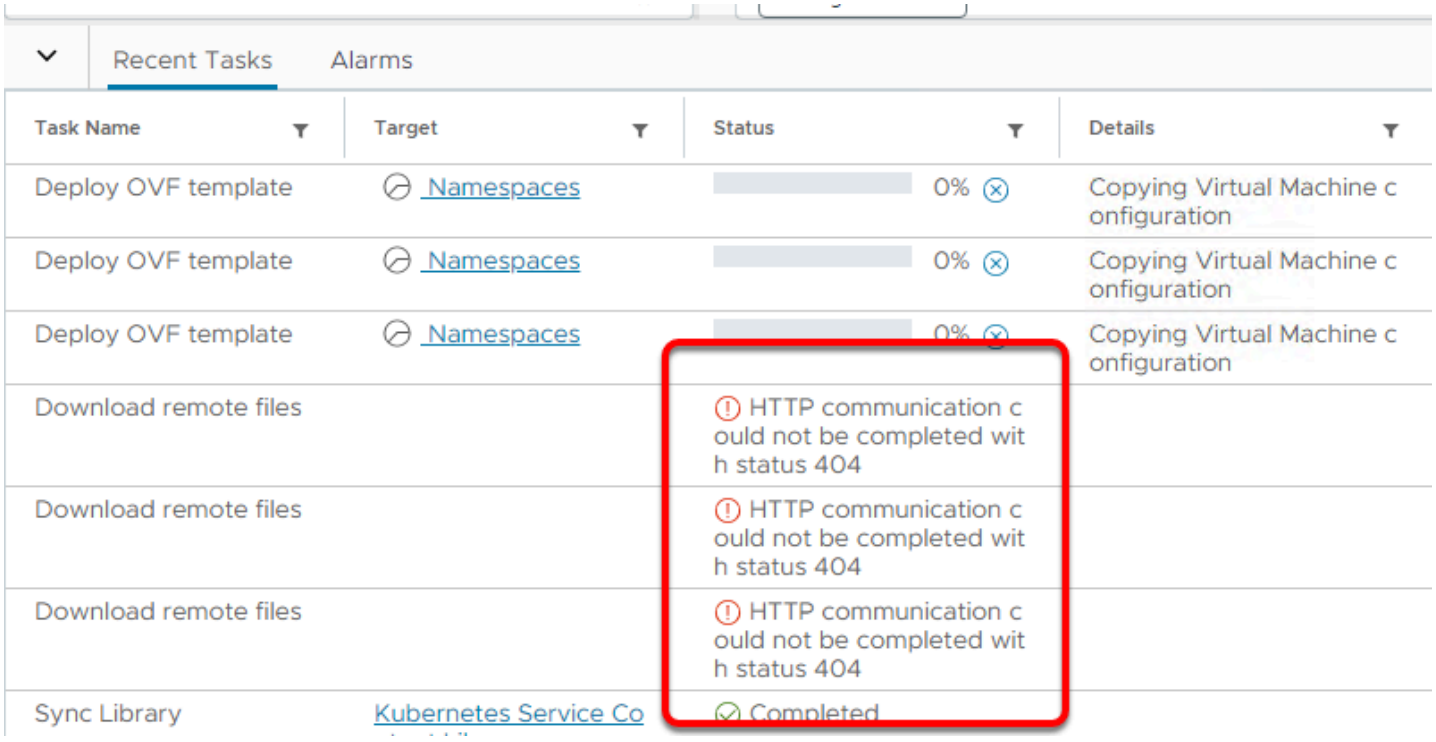
Under **Workload Management** --> **Supervisors Tab** --> Config Status (View)



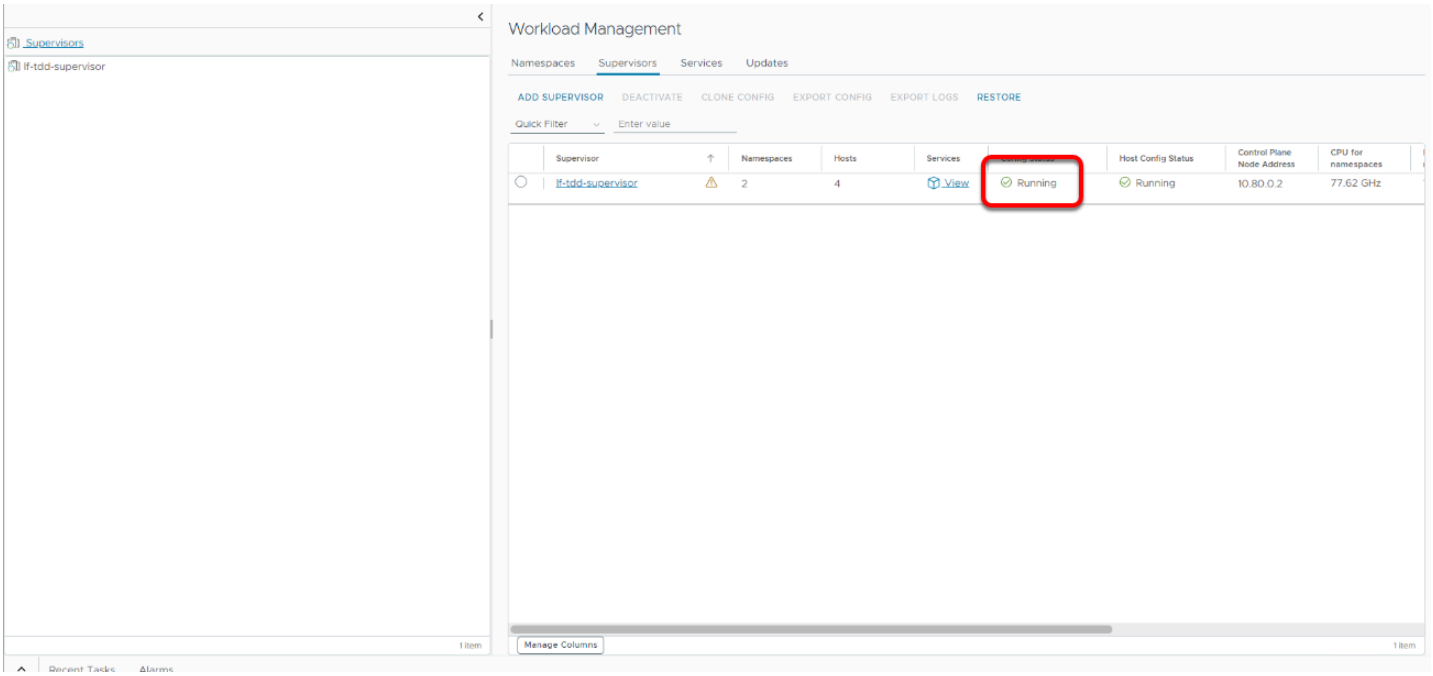


This screen will progress as the Supervisor Nodes Deploy



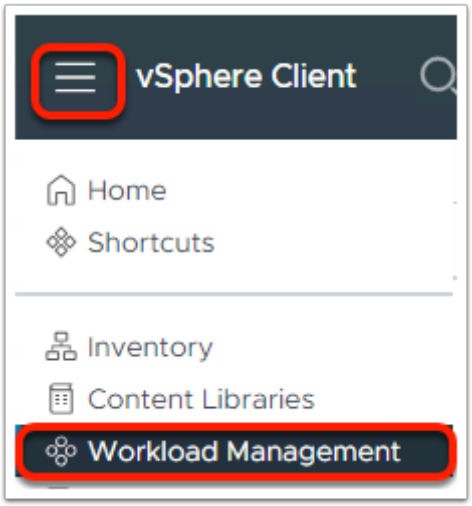


Ignore these errors. They will clear.

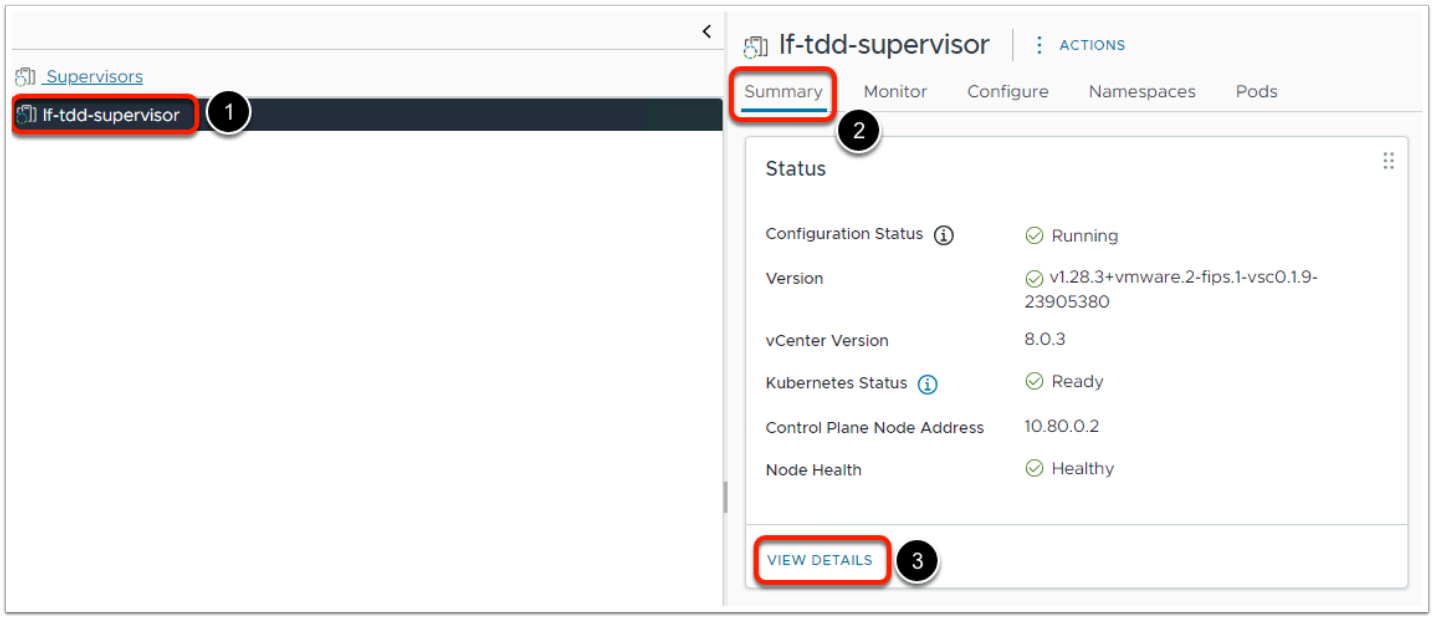


Check the Supervisor Status

Once deployment is complete, please examine and confirm its status

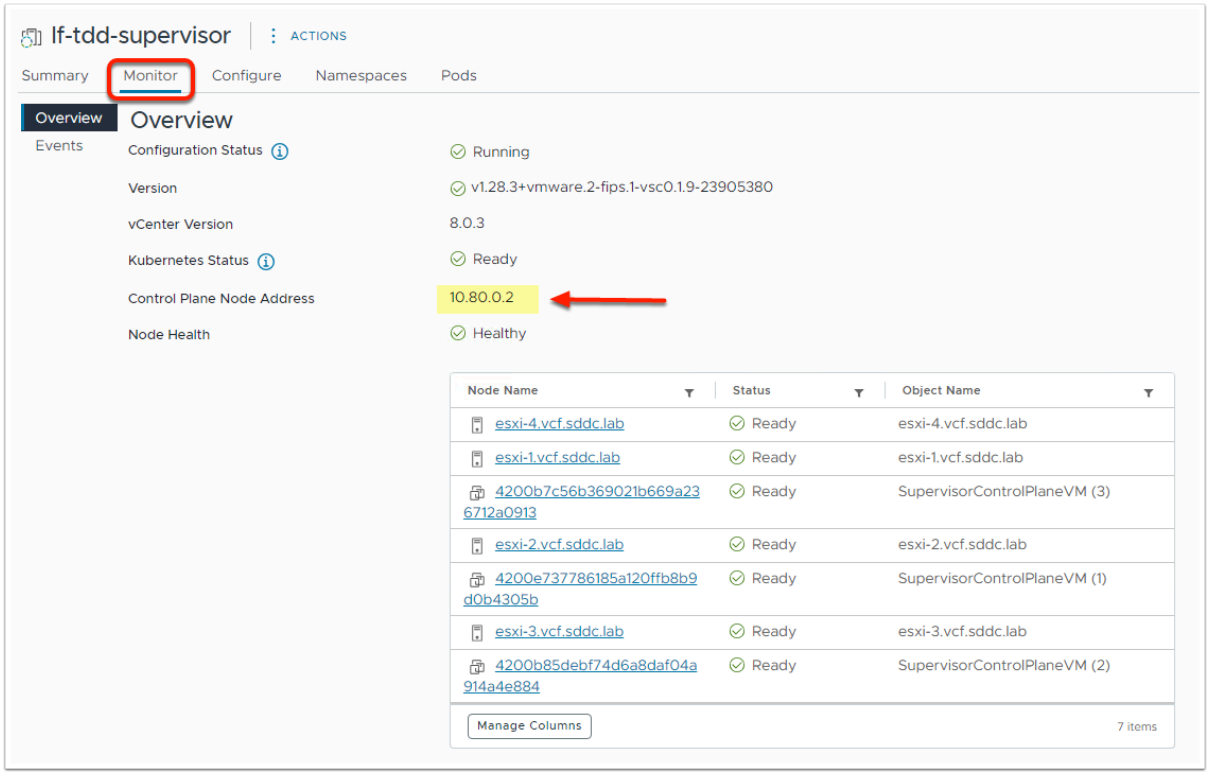


in vSphere client, click on the elipsis in the upper left corner and select Workload Management



In the navigation pane on the left:

1. Go for **Supervisors** -> **lf-tdd-supervisor**
2. This will navigate you to the **Summary** tab: Please note the **Running status**, **Kubernetes status** should be in a **Ready** state. You can also mark the **CP address: 10.80.0.2**
3. Click on **VIEW DETAILS**



This will direct you In Monitor tab:

Install Supervisor Services

Task description and objectives

Supervisor Services are vSphere certified Kubernetes operators that deliver Infrastructure-as-a-Service components and tightly-integrated Independent Software Vendor services to developers. You can install and manage Supervisor Services on the vSphere IaaS control plane environment so that to make them available for use with workloads.

When Supervisor Services are installed on Supervisors, DevOps engineers can consume them in different ways:

Shared Supervisor Services such as Harbor, directly provide functionality to workloads running in TKG clusters, vSphere Pods, or VMs.

Supervisor Services that include an operator, such as MinIO, typically provide API or graphical interfaces, which DevOps engineers can use to create and manage instances of the service in a vSphere Namespace through CRDs. For example, to create a MiniIO bucket, you use a CRD to create the bucket in a vSphere Namespace.

Supervisor Service	vSphere 7	vSphere 8
TKG Service	✗ *	✓ *requires vSphere 8.0 Update 3 or later.*
Consumption Interface	✗	✓
vSAN Data Persistence Platform Services - MinIO, Cloudian and Dell ObjectScale	✓	✓
Backup & Recovery Service - Velero	✓	✓
Certificate Management Service - cert-manager	✗	✓
Cloud Native Registry Service - Harbor	✗ *	✓
Kubernetes Ingress Controller Service - Contour	✗	✓
External DNS Service - ExternalDNS	✗	✓
** The embedded Harbor Registry and TKG Service features are still available and supported on vSphere 7 and onwards.*		

New service will be added overtime with the goal to continue to empower your DevOps communities.

Prior to vSphere 8 Update 1, the Supervisor Services are only available with Supervisor Clusters enabled using VMware NSX-T. With vSphere 8 U1, Supervisor Services are also supported when using the vSphere Distributed Switch networking stack.

Supervisor services can be installed from their catalog, located at https://vsphere-tmm.github.io/Supervisor-Services/

Each one of these goes along with predefined manifest files of two stages: one manifest (service-name.yaml) for the service registration with the Supervisor, and second manifest (config-values.yaml) for particular service settings you need to apply in accordance with the environment.

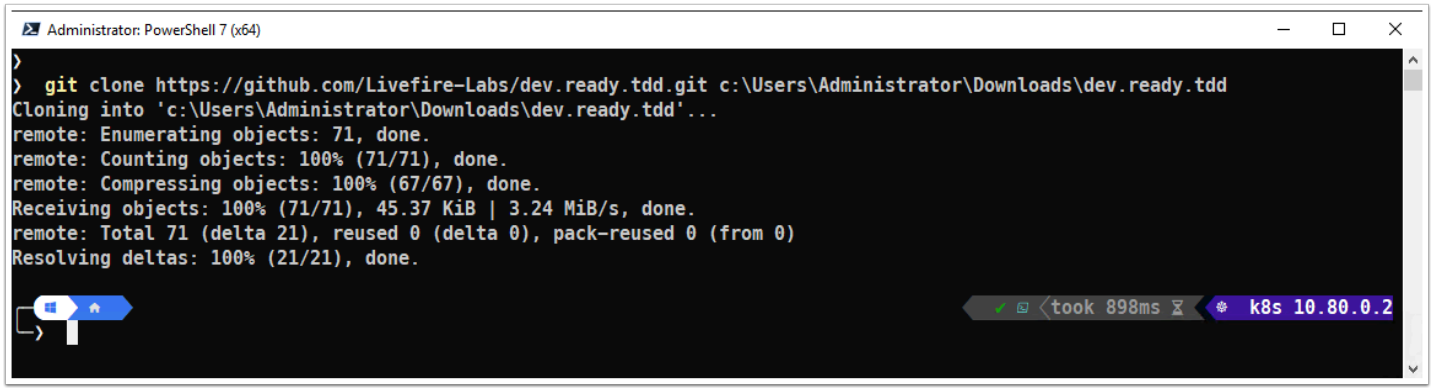
With vSphere 8.0 u3 and VCF 5.2 onwards, couple of services are automatically deployed when you enable the Supervisor:

- VM Service

- Tanzu Kubernetes Grid Service

- Velero vSphere Operator

Clone Github



First things first you will need some files. we've put them in a public github repository./ You will need to clone them to your DOwnloads folder

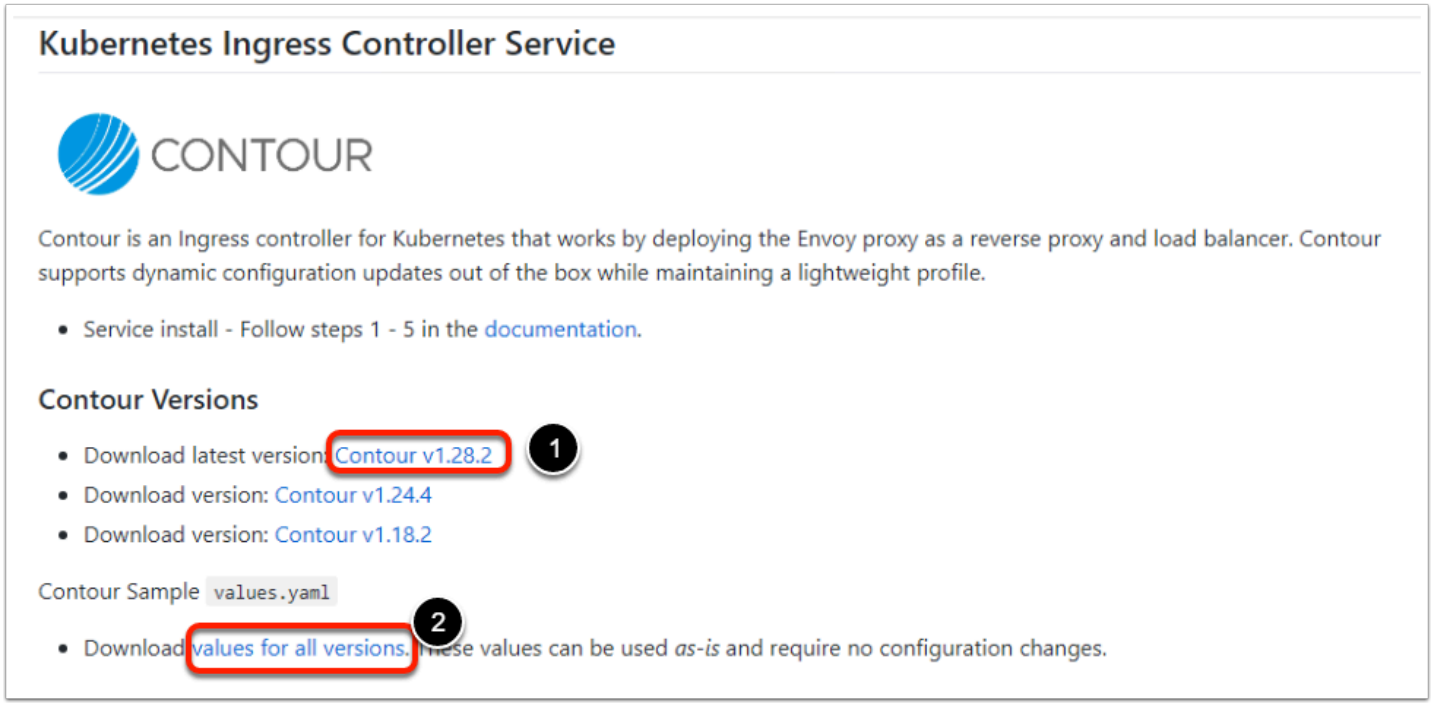
Run the git clone command below:

git clone https://github.com/Livefire-Labs/dev.ready.tdd.git c:\Users\Administrator\Downloads\dev.ready.tdd
This will create a ***dev.ready.tdd*** folder in the ***c:\Users\Administrator\Downloads*** folder

Install Kubernetes Ingress Controller Service (Contour)

For a **Kubernetes Ingress Controller Service** you can use **Contour**.

Contour is an Ingress controller for Kubernetes that works by deploying the Envoy proxy as a reverse proxy and load balancer. Contour supports dynamic configuration updates out of the box while maintaining a lightweight profile.



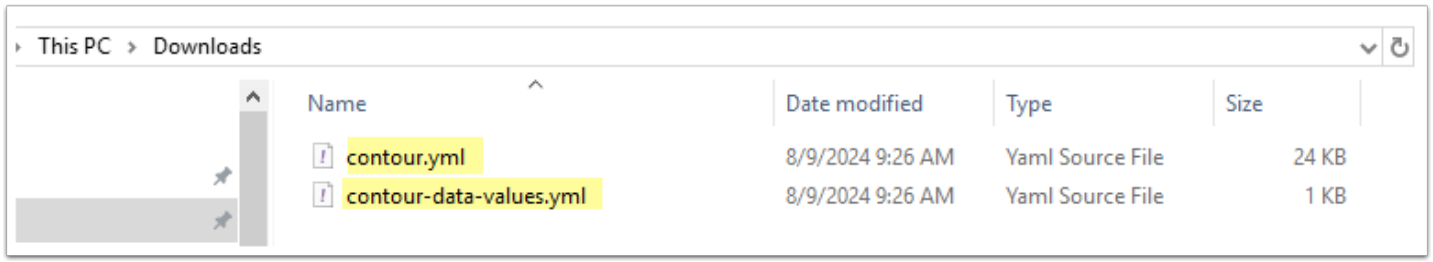
Download the latest available manifests for Contour to your Downloads folder

1. contour.yml

2. contour-data-values.yml

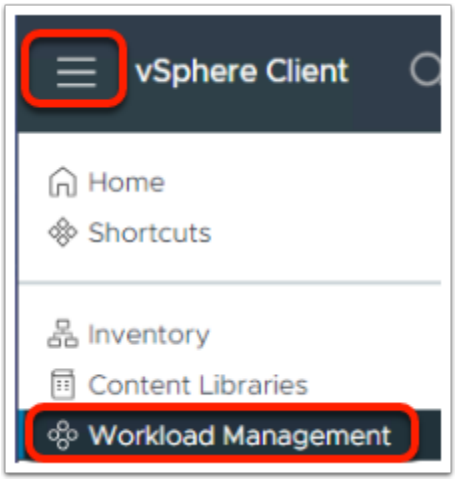
https://vsphere-tmm.github.io/Supervisor-Services/#kubernetes-ingress-controller-service

Install Kubernetes Ingress Controller Service (Contour)



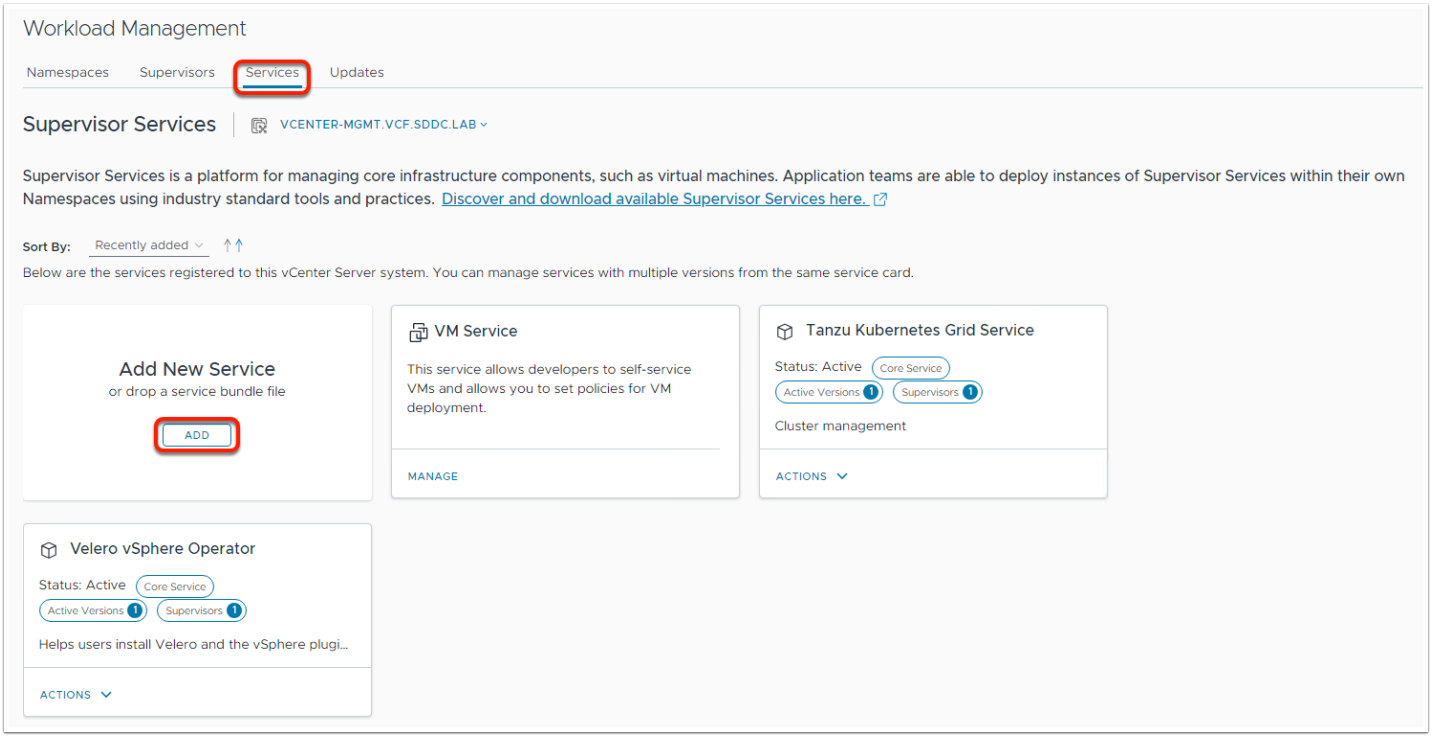
You are ready to deploy Contour as a Supervisor service!

Install Kubernetes Ingress Controller Service (Contour)



From the vSphere Client home menu, select **Workload Management**.

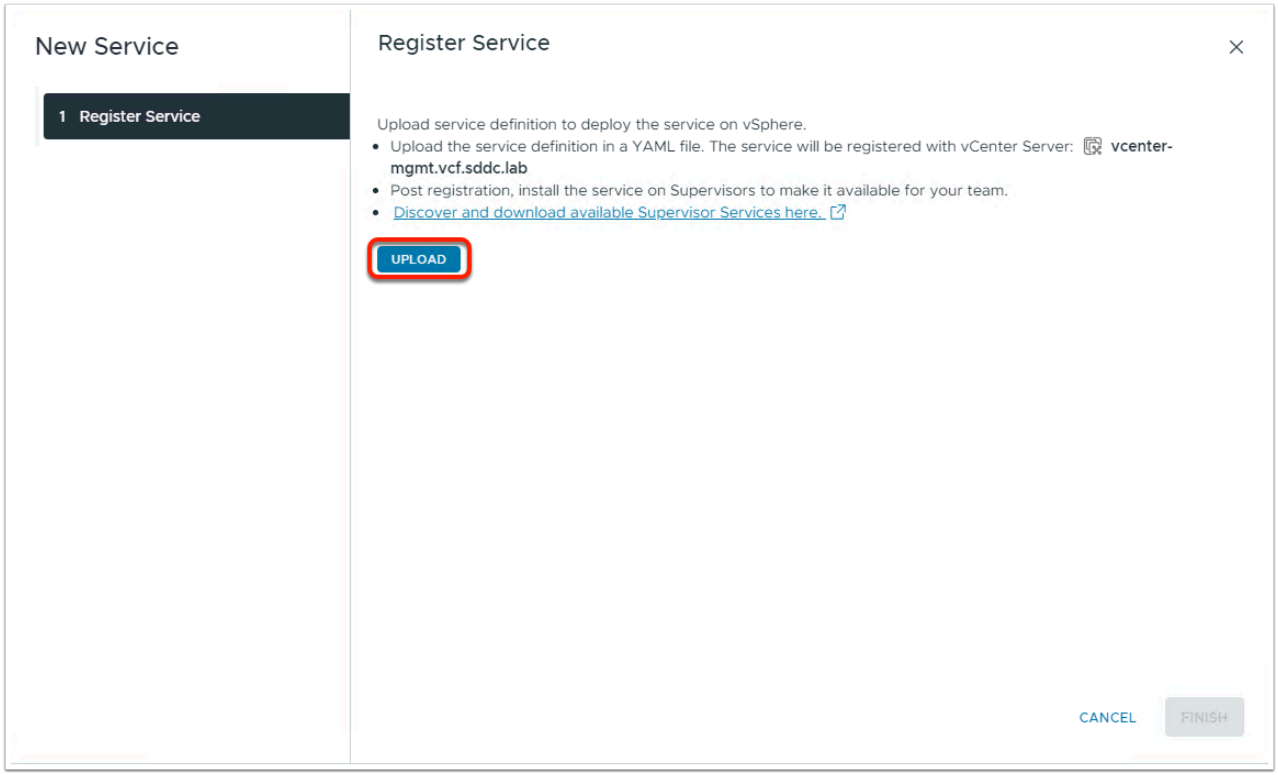
Install Kubernetes Ingress Controller Service (Contour)



In the **Workload Management** section, select **Services**.

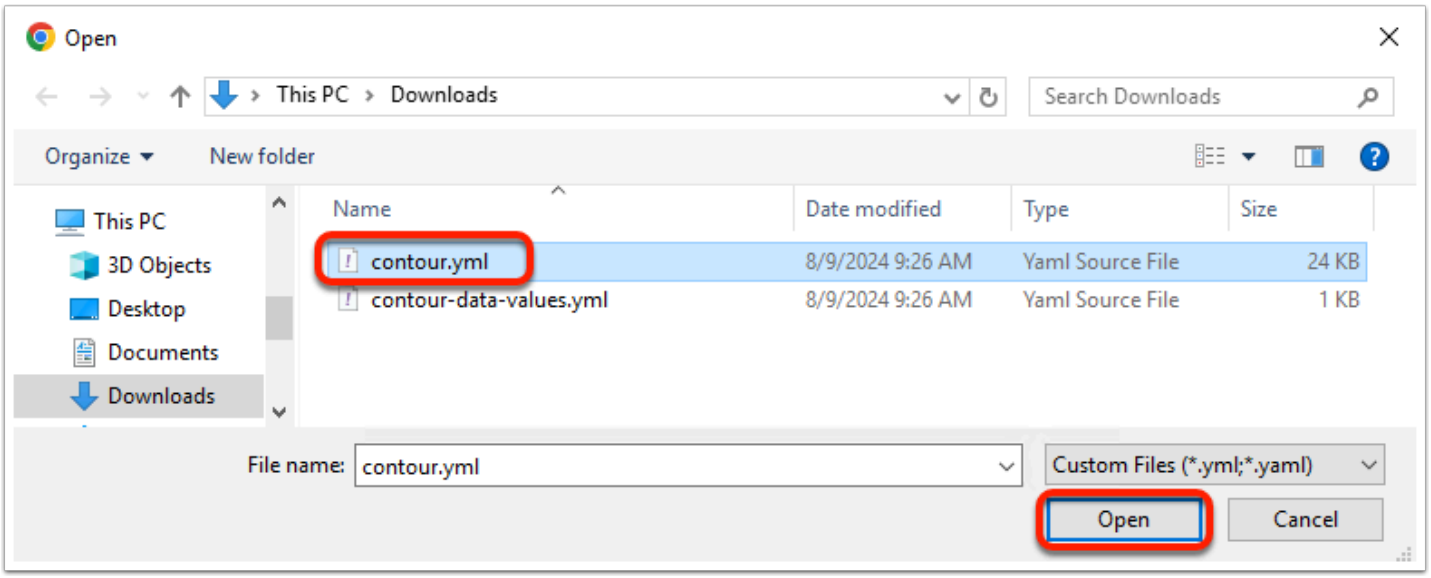
Locate the **Add New Service** card and click on **ADD**

Install Kubernetes Ingress Controller Service (Contour)



In the **New Service** wizard, for **Register Service** select **UPLOAD**

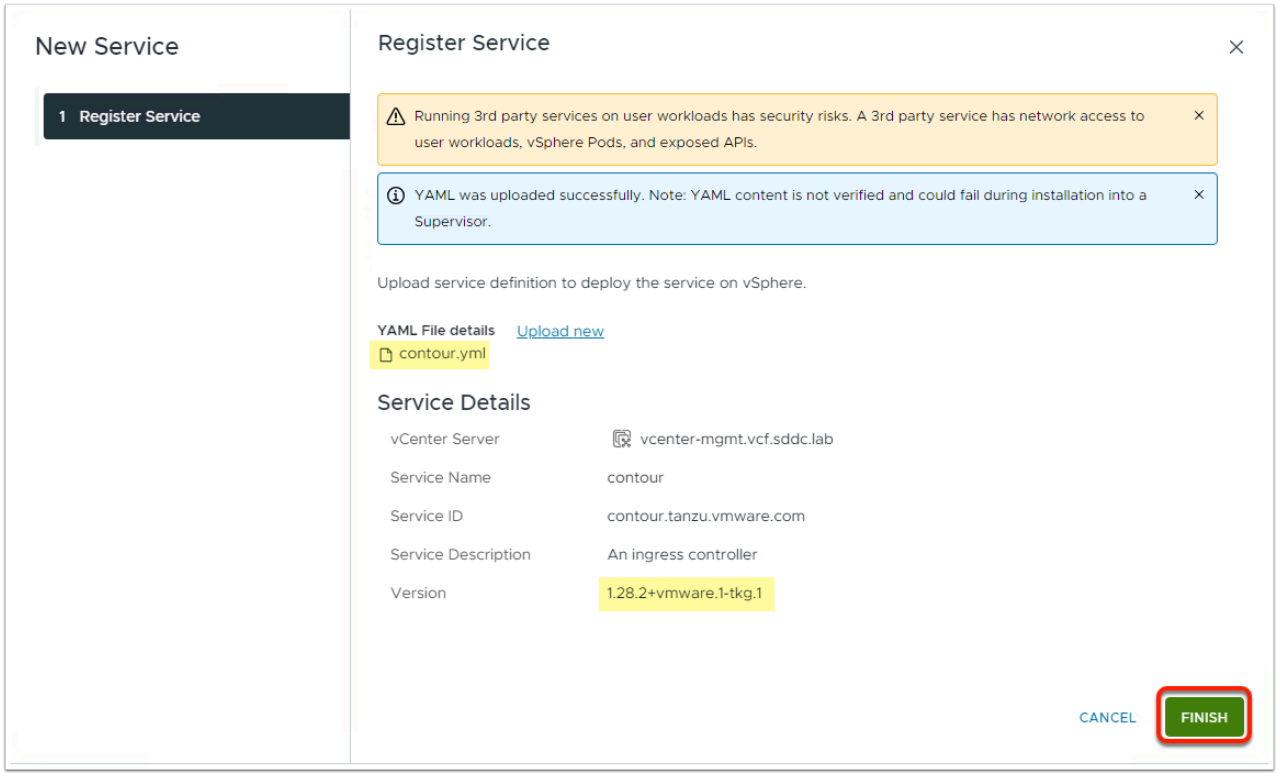
Install Kubernetes Ingress Controller Service (Contour)



In the Chrome **Open** menu go to **Downloads** folder and select the **contour.yml** you downloaded recently.

Confirm with clicking on **Open** button

Install Kubernetes Ingress Controller Service (Contour)

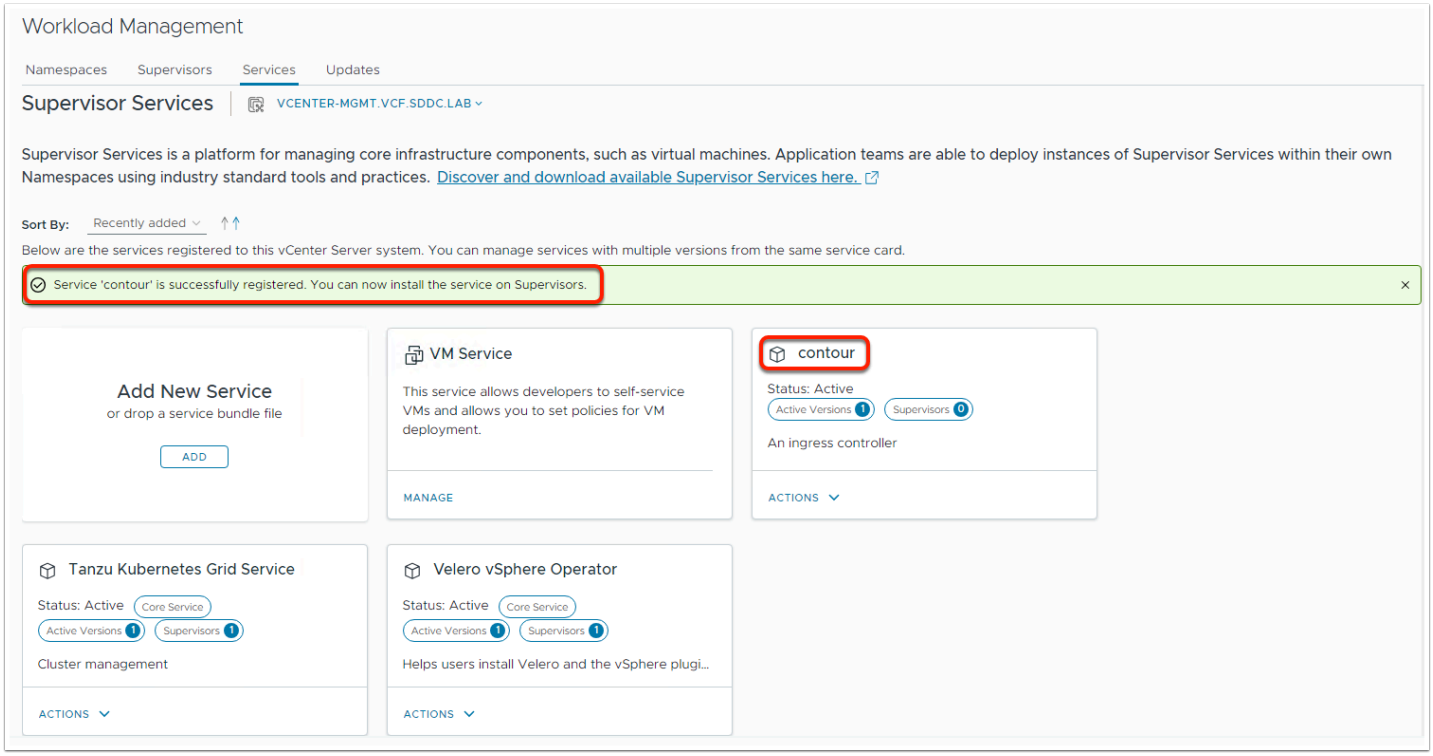


In the **New Service** wizard, for **Register Service** make sure you can see the aforementioned **contour.yml** file.

Its displayed version should match the one you downloaded.

Confirm with **FINISH**

Install Kubernetes Ingress Controller Service (Contour)

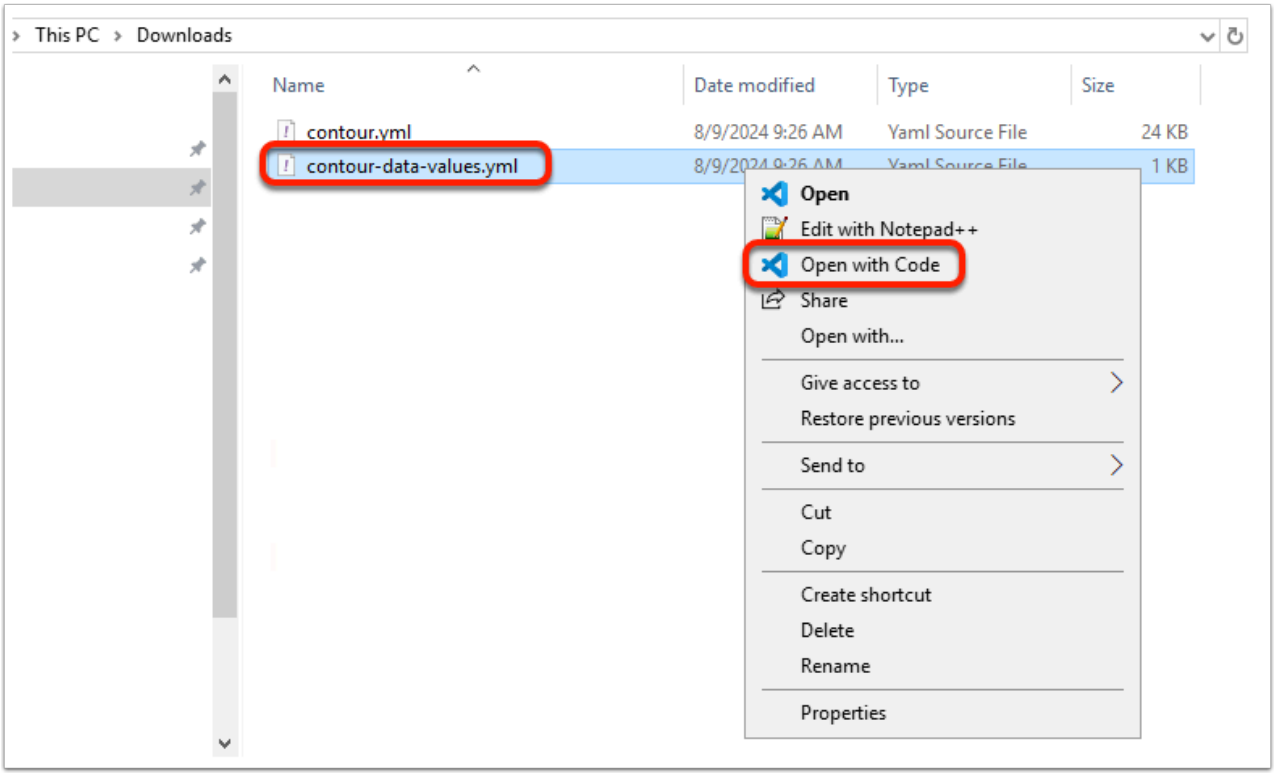


In a while you will see a displayed message for the Contour service successfully registered with your Supervisor.

You need to activate it with particular configuration settings.

You can safely close the message in green.

Install Kubernetes Ingress Controller Service (Contour)



Let's examine the particular file. You can review/edit it with any available editor

In Windows Explorer, navigate to your **Downloads** folder and locate the recently downloaded file **contour-data-values.yml**.

Right click and **Open with Code**

Install Kubernetes Ingress Controller Service (Contour)

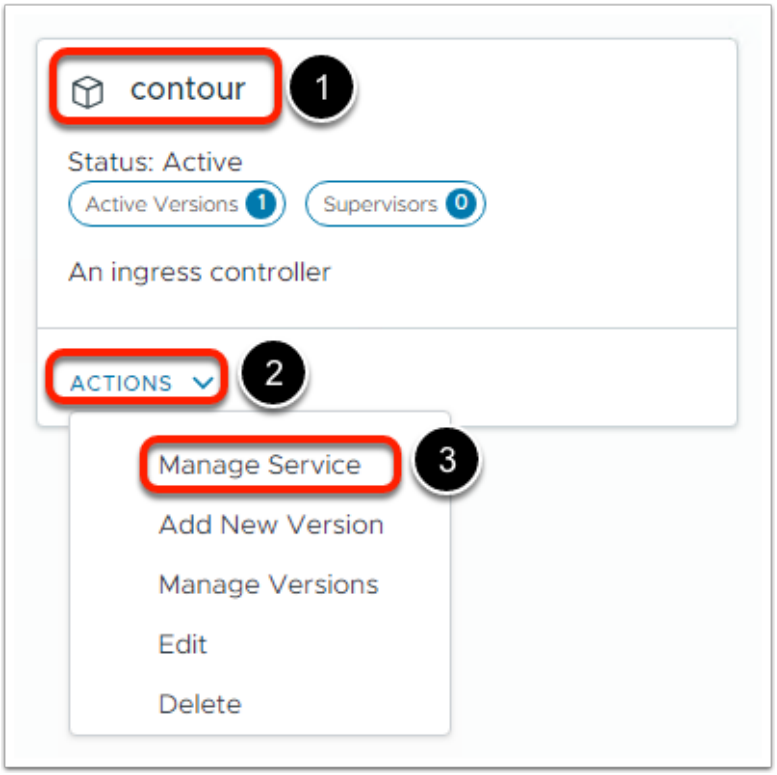
C: > Users > Administrator > Downloads > contour-data-values.yml >

```yaml
contour:
  configFileContents: {}
  useProxyProtocol: false
  replicas: 2
  pspNames: ""
  logLevel: info
envoy:
  service:
    type: LoadBalancer
    externalTrafficPolicy: Cluster
    disableWait: false
  hostPorts:
    enable: false
    http: 80
    https: 443
  hostNetwork: false
  terminationGracePeriodSeconds: 300
  logLevel: info
  pspNames: ""
certificates:
  duration: 8760h
  renewBefore: 360h

```

For Contour data values you don't need to edit anything.

Don't close the file, you will need the values shortly!

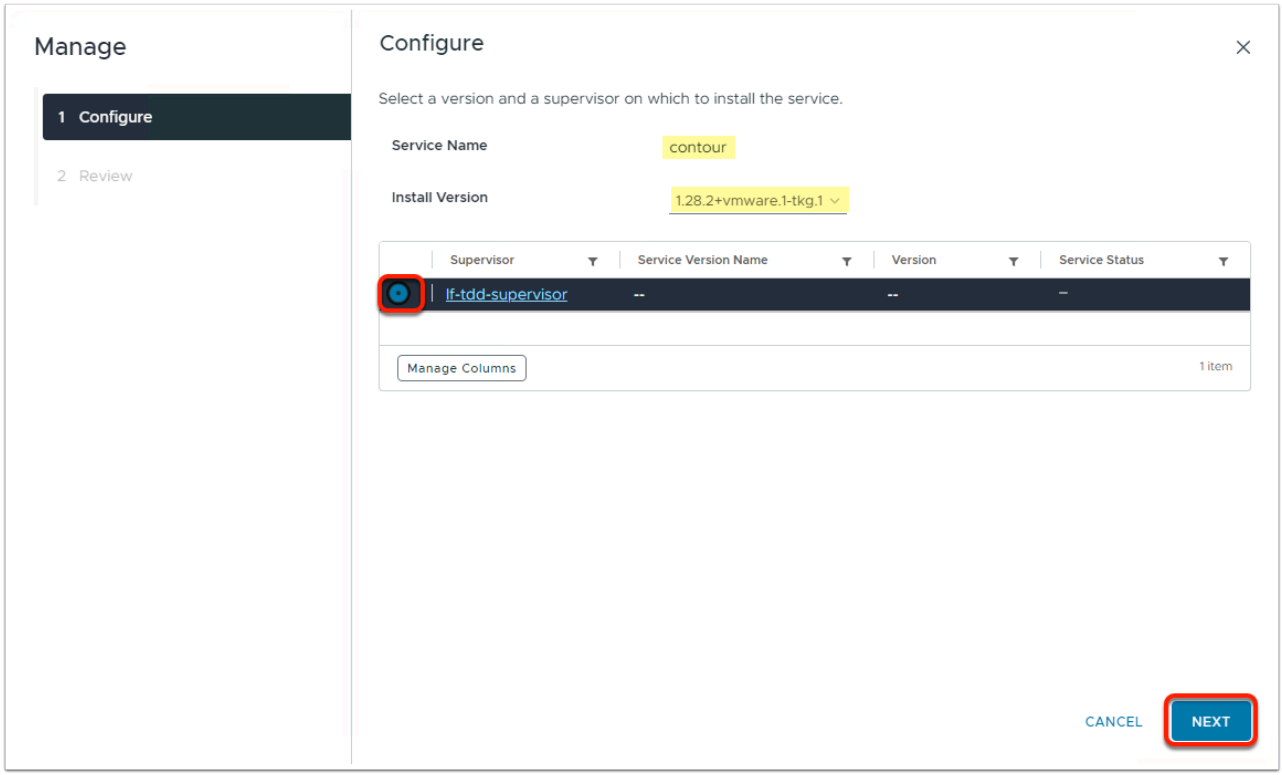
Install Kubernetes Ingress Controller Service (Contour)



Go back to Chrome -> vSphere UI -> Workload Management -> Services

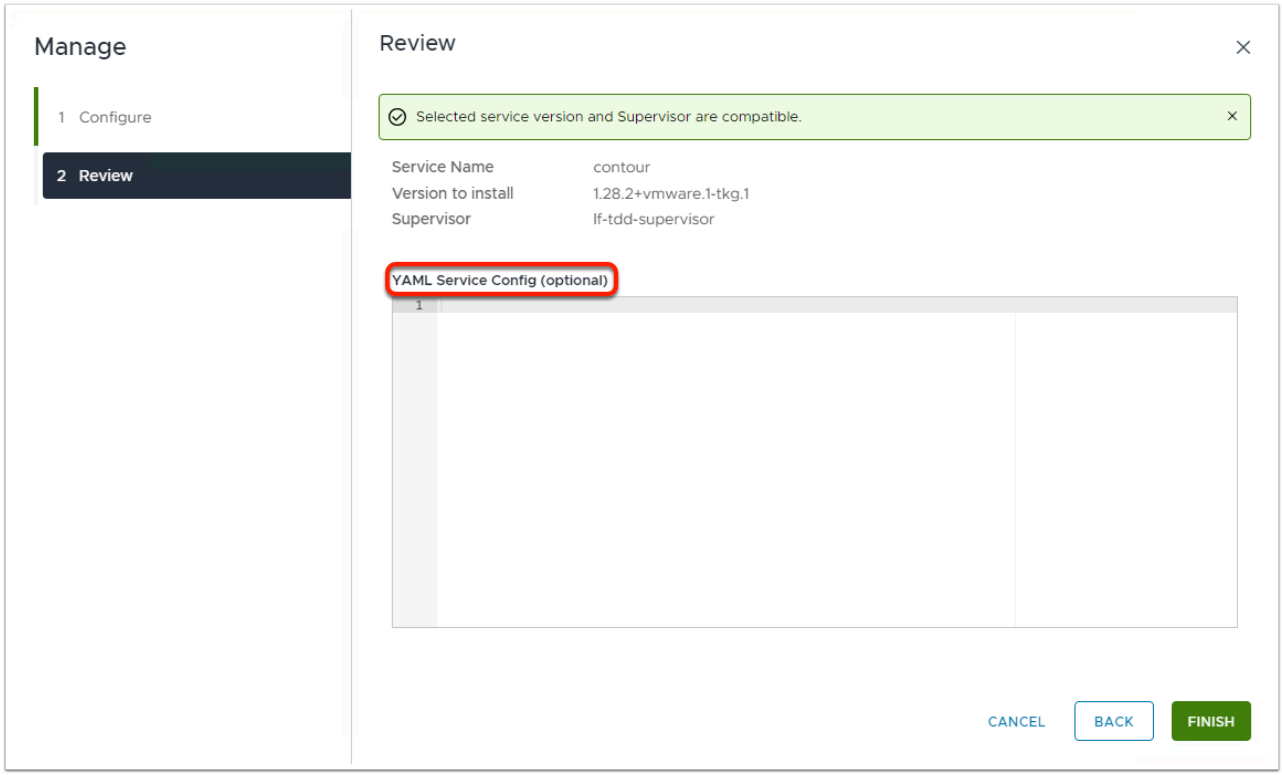
1. Locate your recently registered **contour** service
2. From **Actions** drop down menu
3. select **Manage Service**

Install Kubernetes Ingress Controller Service (Contour)



In the **Manage** wizard, from **Configure** select your **Supervisor** and confirm **NEXT**

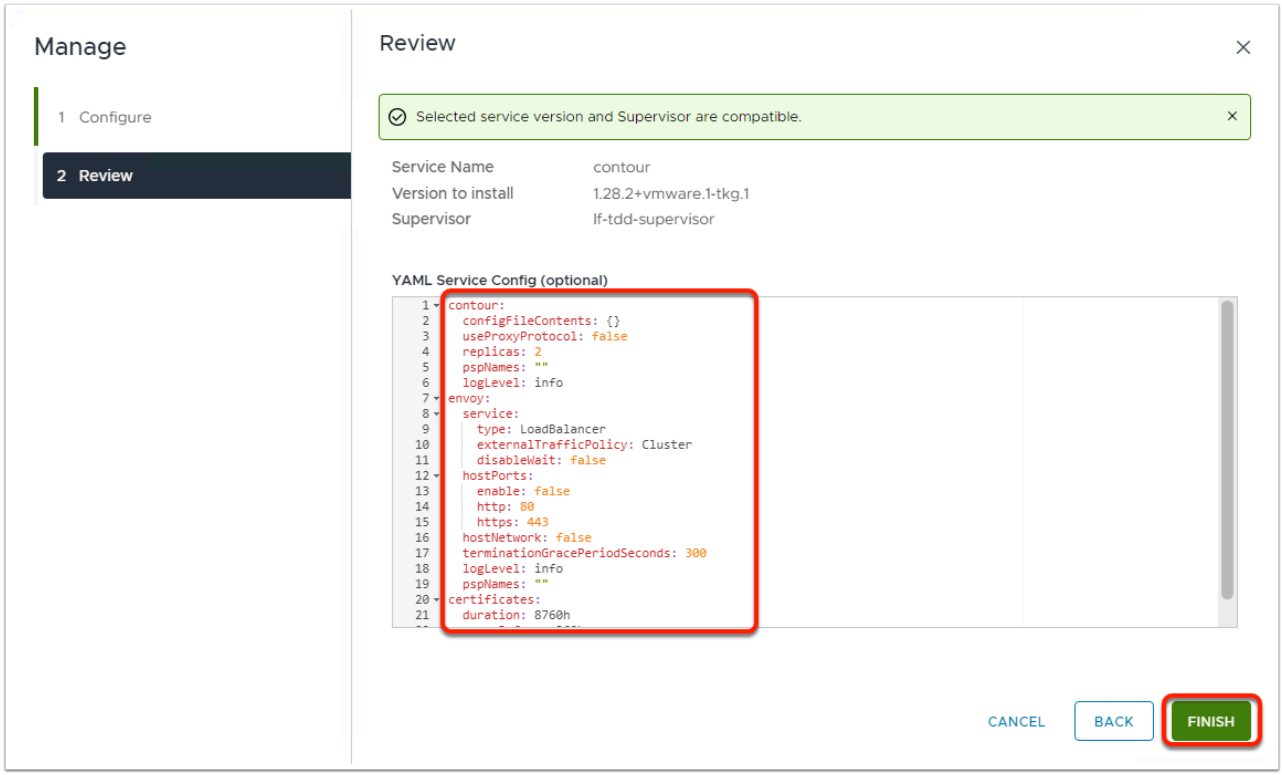
Install Kubernetes Ingress Controller Service (Contour)



You need to provide the service configuration values.

Copy these from the **Visual Studio Code**, where you lastly reviewed **contour-data-values.yml**

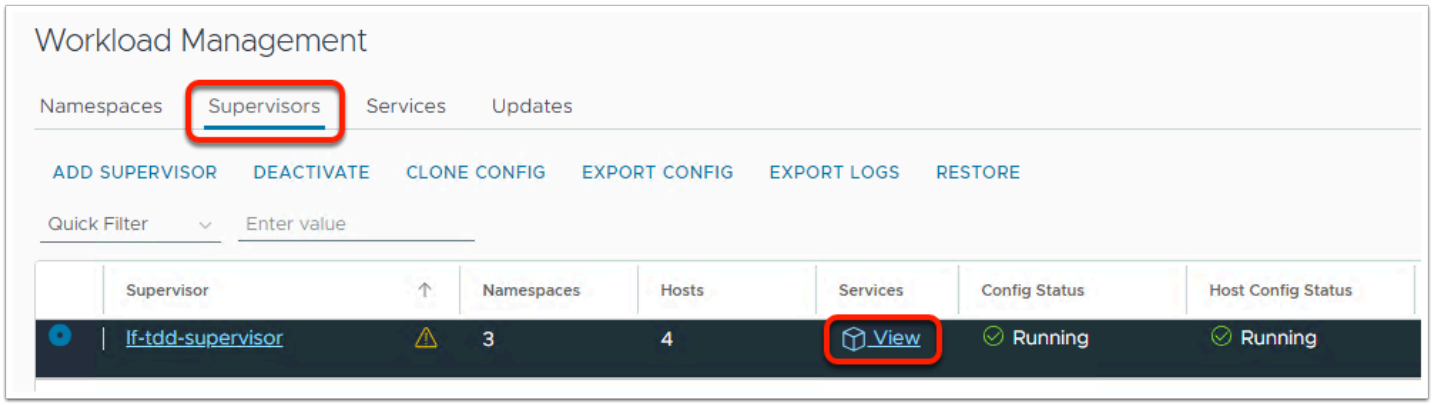
Install Kubernetes Ingress Controller Service (Contour)



Select all and paste into the **YAML Service Config** section in the UI wizard.

Confirm with **FINISH**

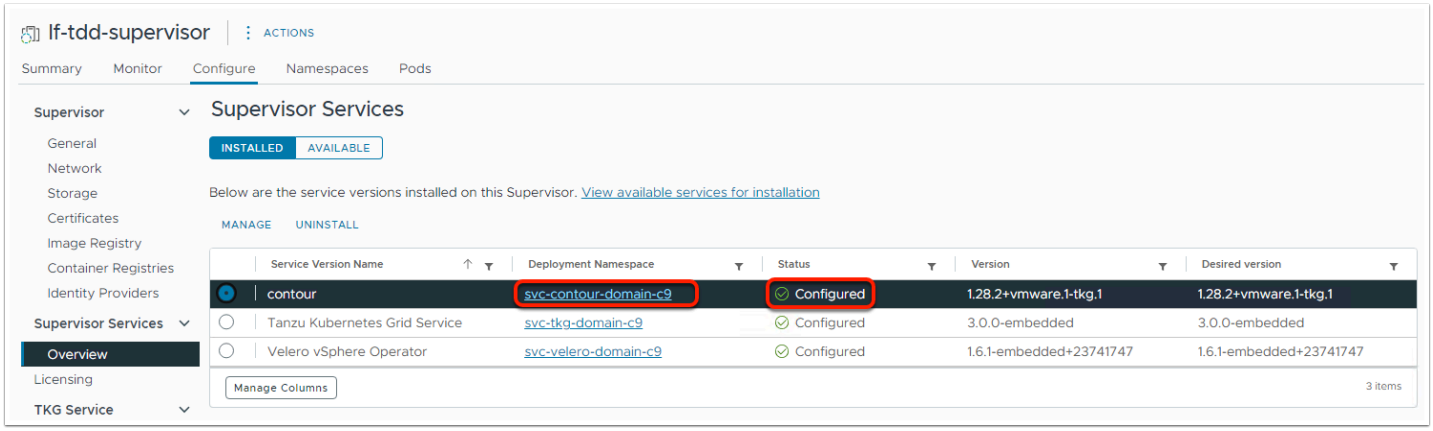
Install Kubernetes Ingress Controller Service (Contour)



Go back in the **Workload Management** section

Select **Supervisors** and click on **View** under **Services**

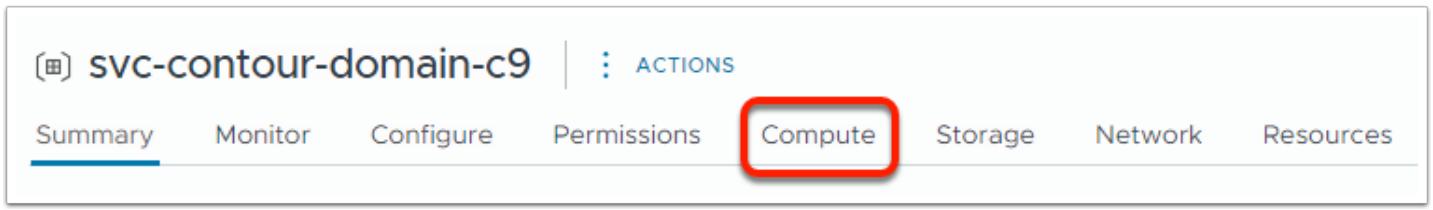
Install Kubernetes Ingress Controller Service (Contour)



In a couple of minutes, the **contour** service **Status** will change from Configuring to **Configured**

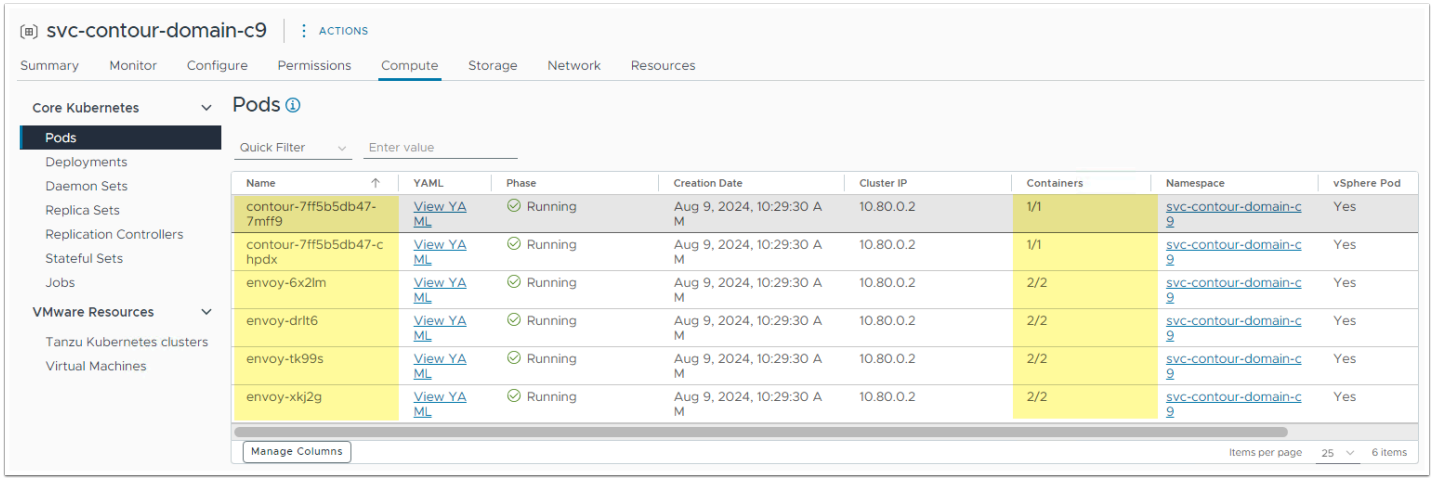
Click on the **svc-contour-domain-c9** under **Deployment Namespace**

Install Kubernetes Ingress Controller Service (Contour)



For the **svc-contour-domain-c9** Namespace, click on **Compute**

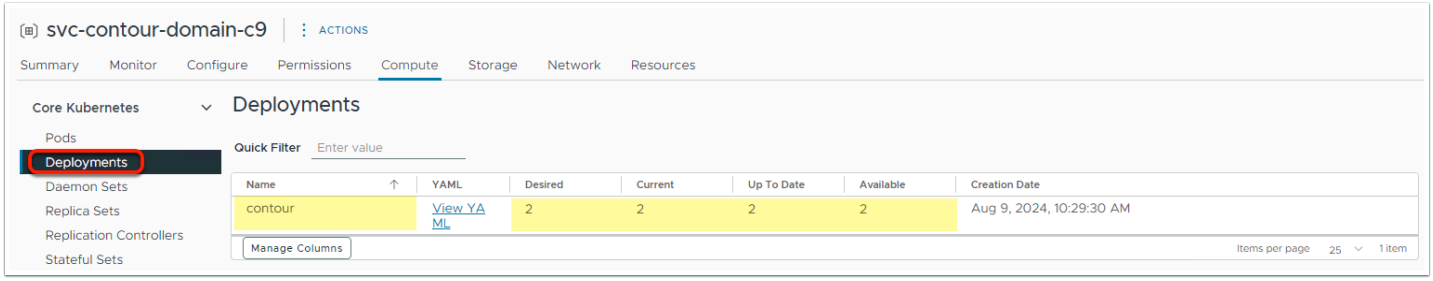
Install Kubernetes Ingress Controller Service (Contour)



Under **Compute**, you can review all the **Kubernetes Core** configurations for harbor.

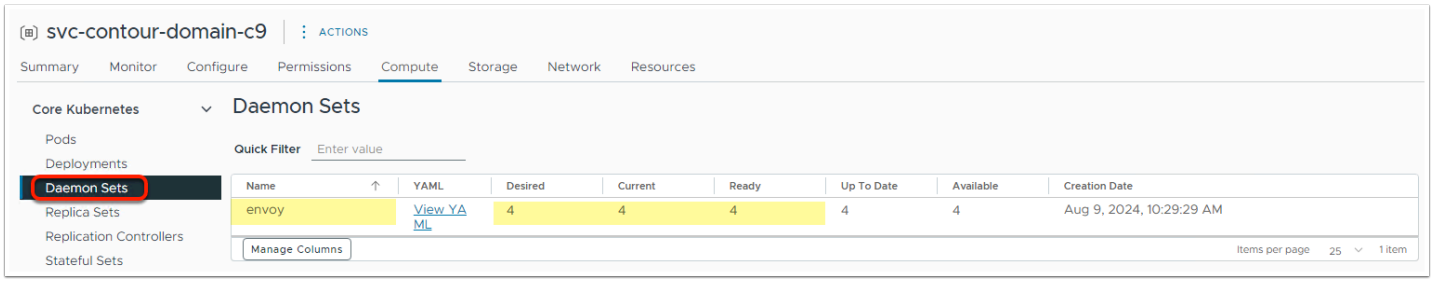
Select **Pods** and you will see all the **contour** needed pods, running as **vSphere Pods**

Install Kubernetes Ingress Controller Service (Contour)



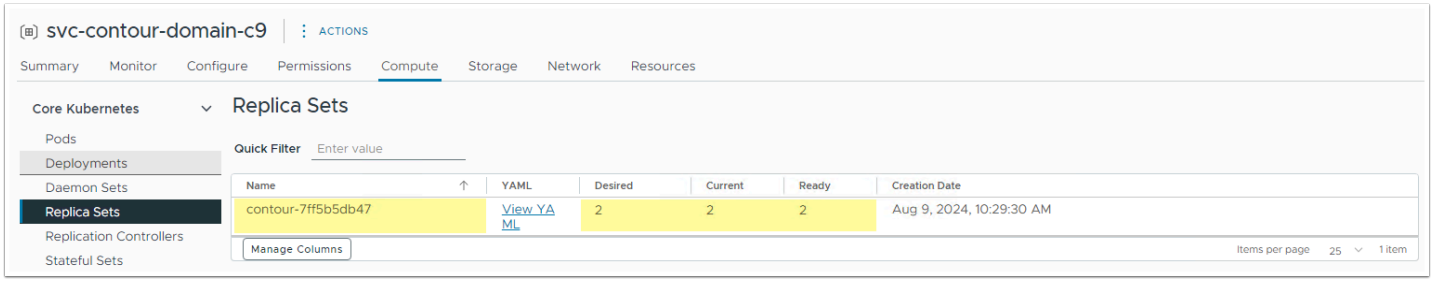
Select **Deployments** and you will see all the **contour** needed deployments in place

Install Kubernetes Ingress Controller Service (Contour)



Select **Daemon Sets** and you will see all the **contour** needed daemon sets in place: for envoy

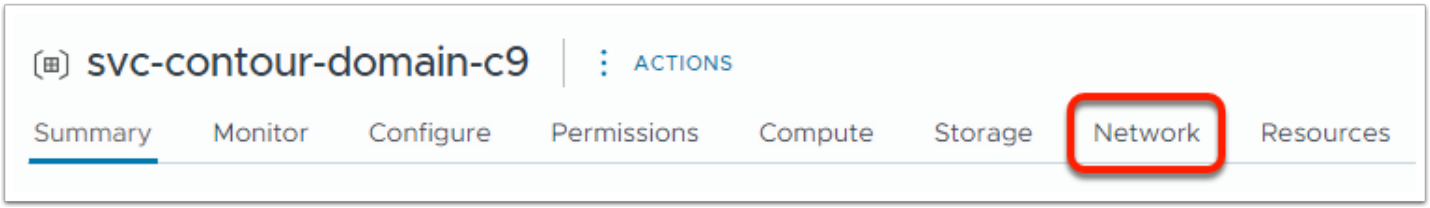
Install Kubernetes Ingress Controller Service (Contour)



Select **Replica Sets** and you will see all the **contour** needed replica sets in place

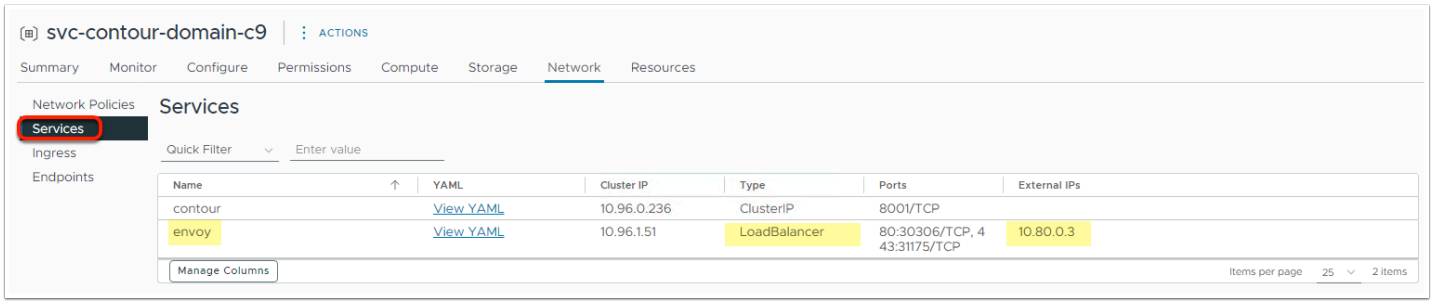
Okay, let's switch to the Network configurations for **contour** service:

Install Kubernetes Ingress Controller Service (Contour)



For the **svc-contour-domain-c9** Namespace, click on **Network**

Install Kubernetes Ingress Controller Service (Contour)



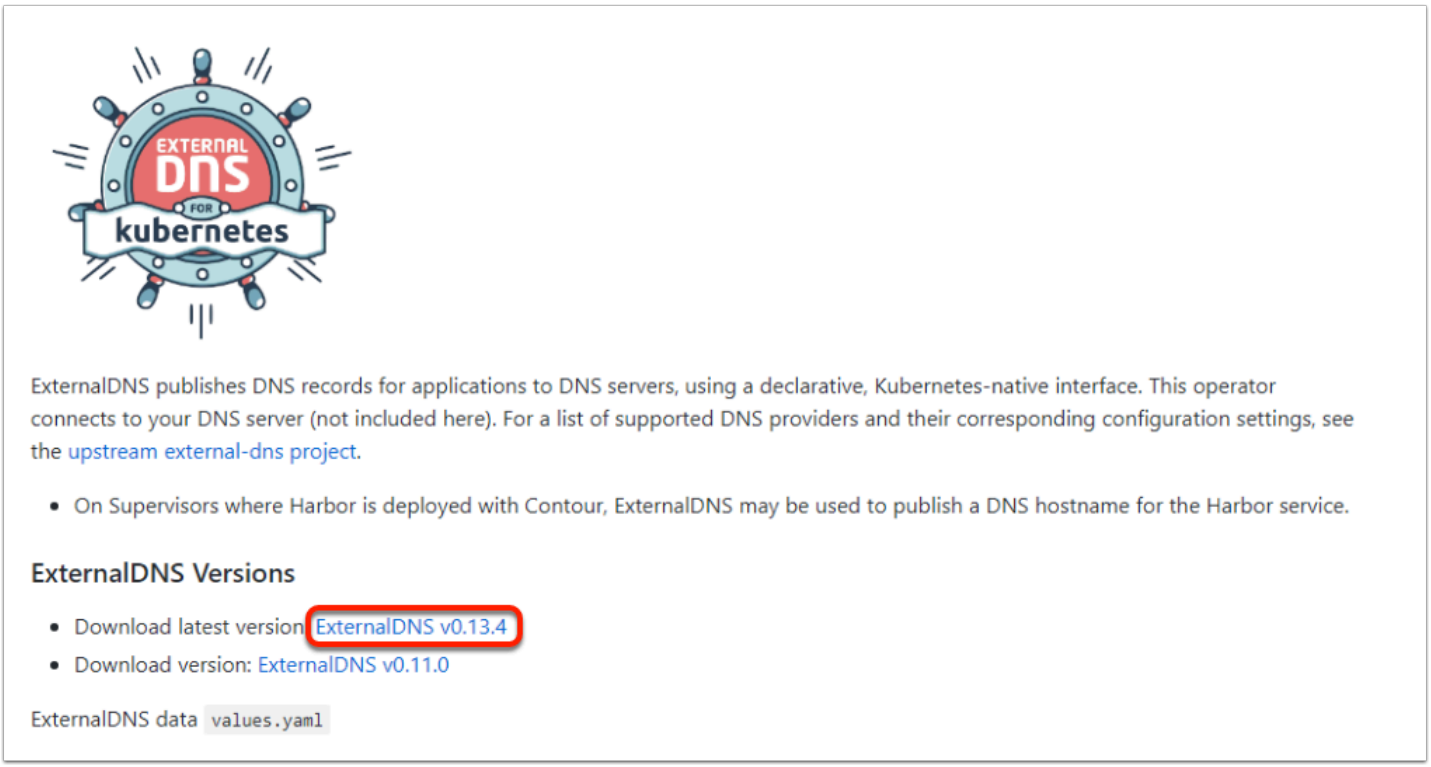
Under **Network**, you can review all the configured settings.

Select **Services** and you will see the **External IP Address** that should be used as **LoadBalancer**, in this case it is **10.80.0.3**

Remember this value, you are going to use it shortly!

Install ExternalDNS

For publishing **DNS records for Applications** to DNS servers, you can use **ExternalDNS** Service.

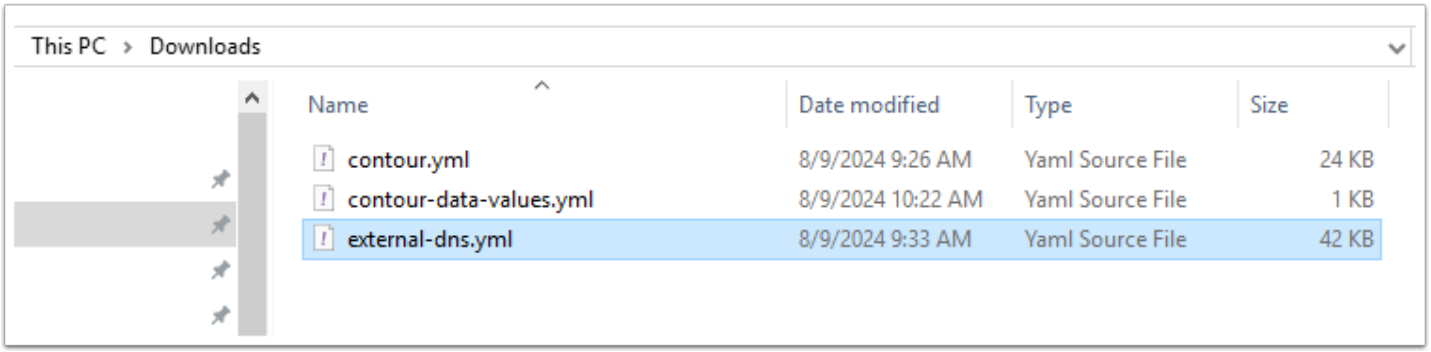


Download the latest available manifests for External DNS to your Downloads folder

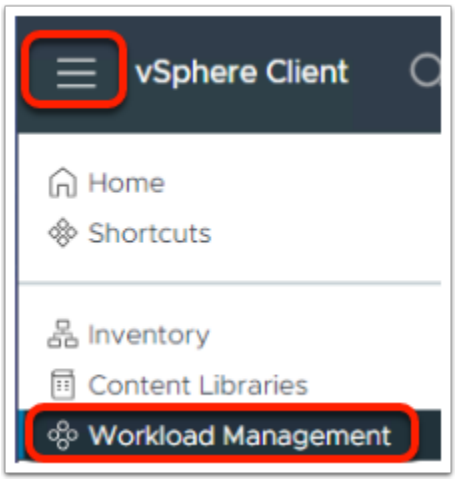
1. external-dns.yml

https://vsphere-tmm.github.io/Supervisor-Services/#external-dns-service

Install ExternalDNS

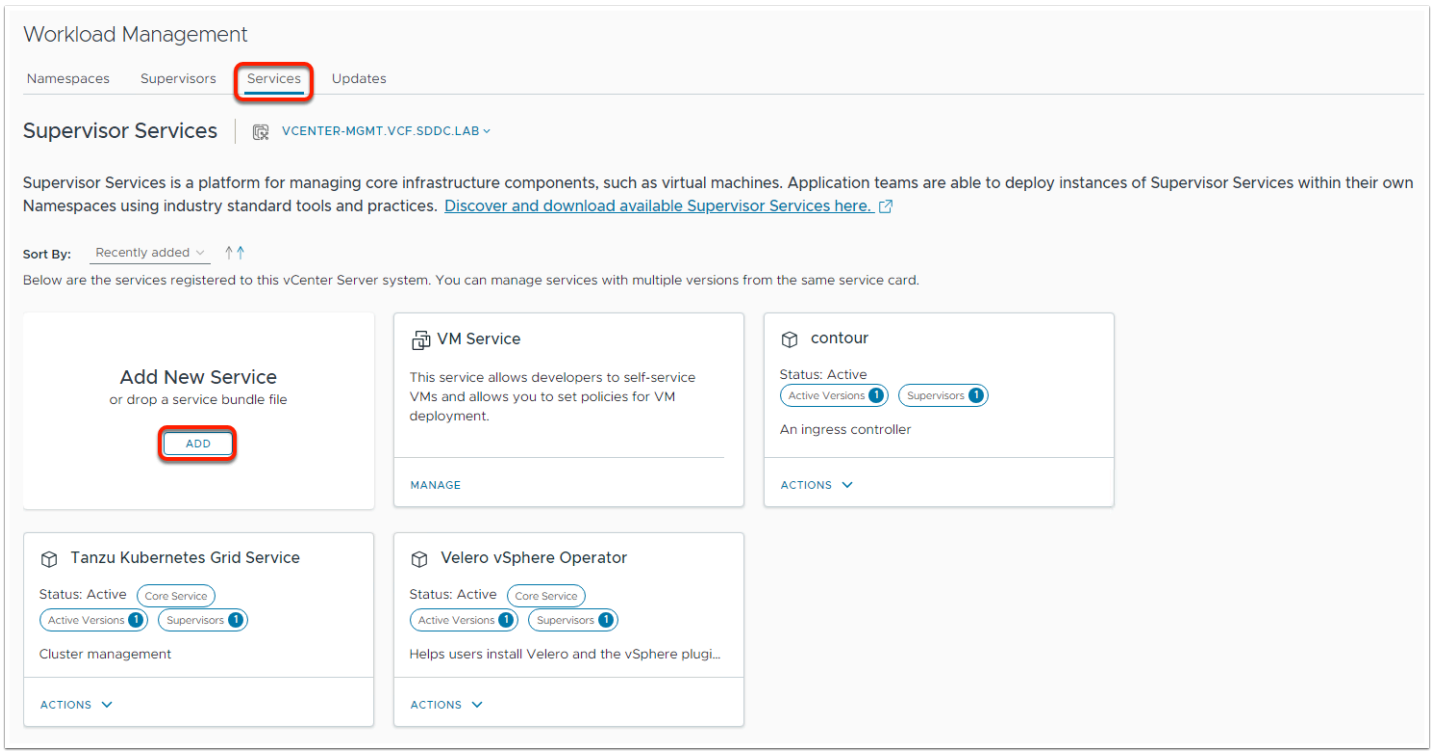


You are ready to deploy External DNS as a Supervisor service!



From the vSphere Client home menu, select **Workload Management**.

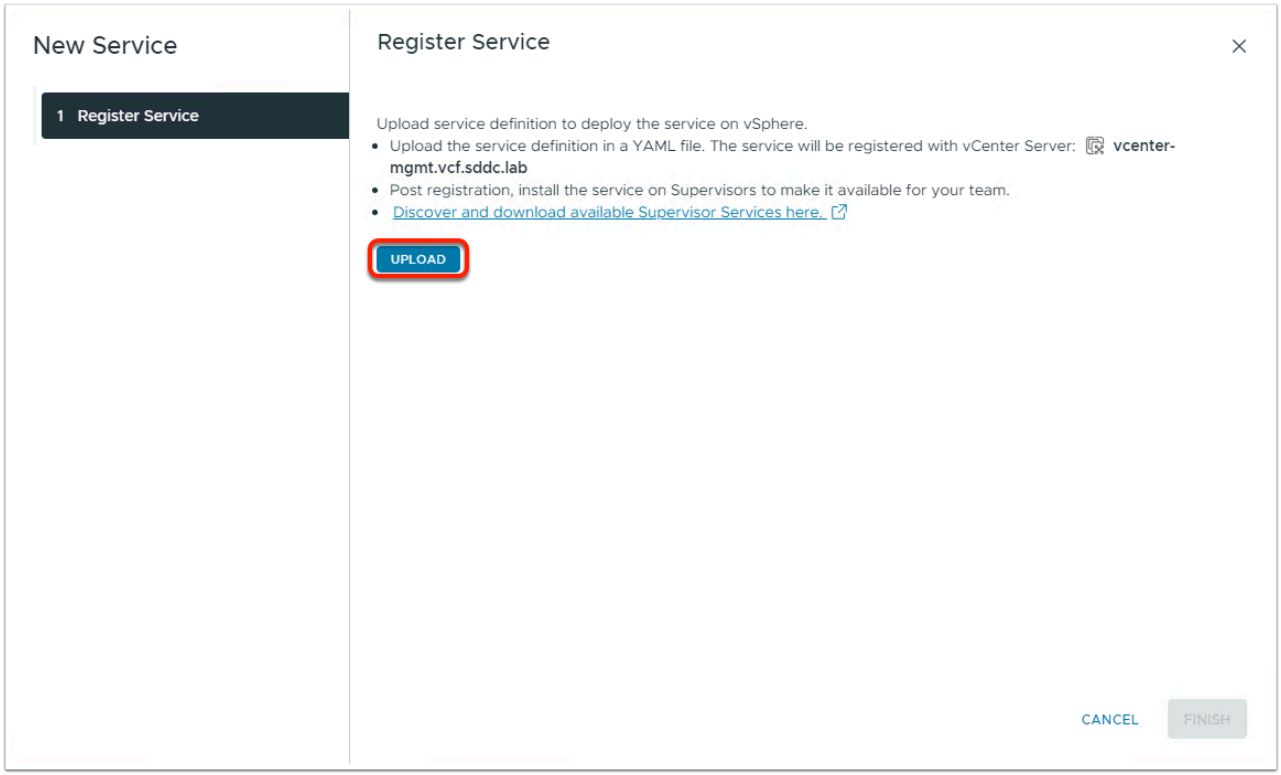
Install ExternalDNS



In the **Workload Management** section, select **Services**.

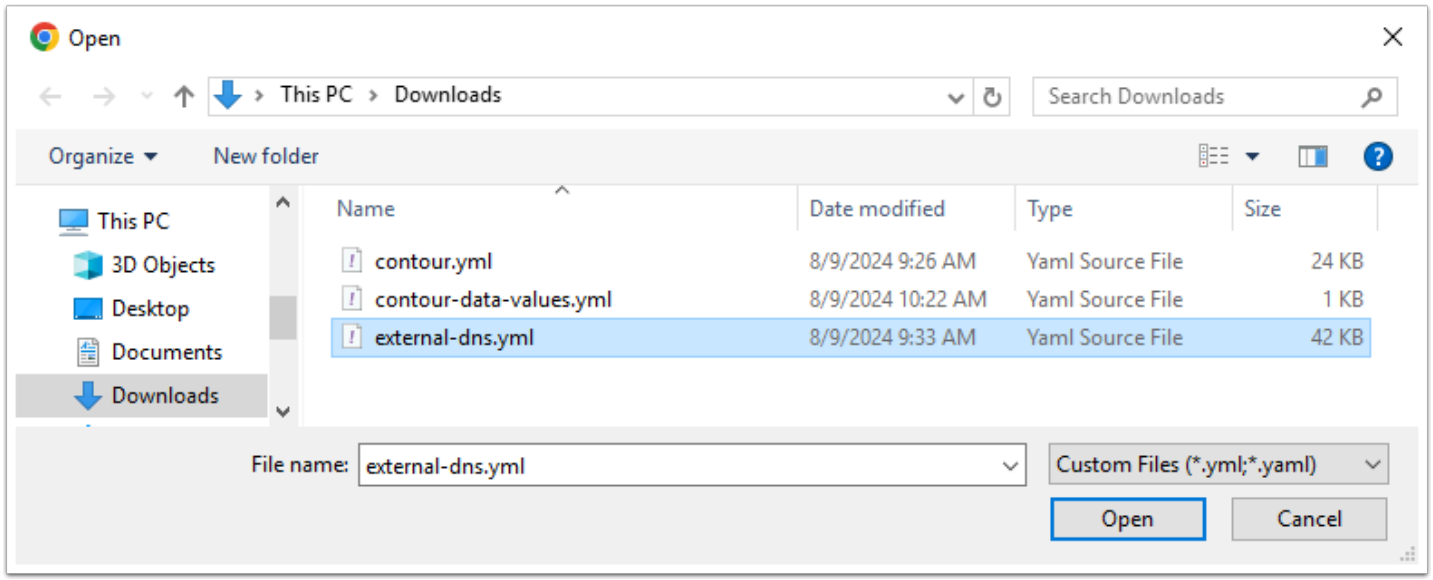
Locate the **Add New Service** card and click on **ADD**

Install ExternalDNS



In the **New Service** wizard, for **Register Service** select **UPLOAD**

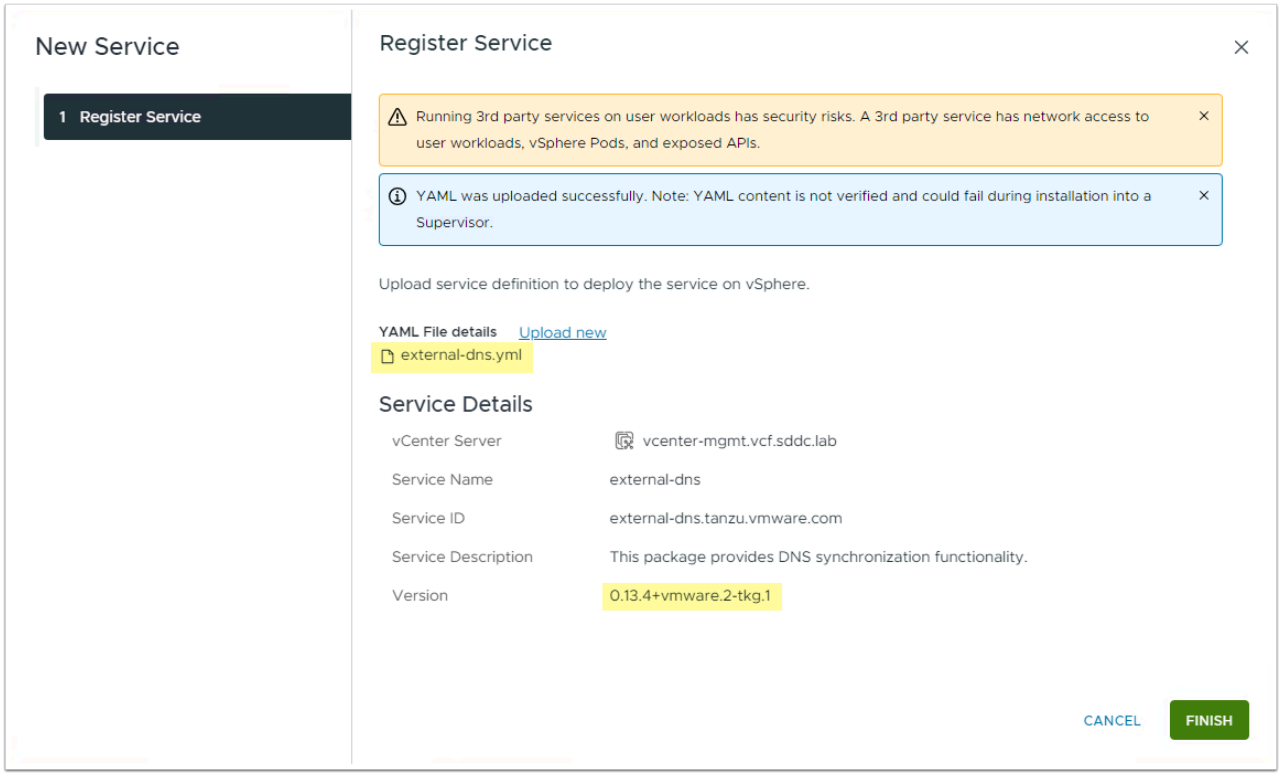
Install ExternalDNS



In the Chrome **Open** menu go to **Downloads** folder and select the **external-dns.yml** you downloaded recently.

Confirm with clicking on **Open** button

Install ExternalDNS

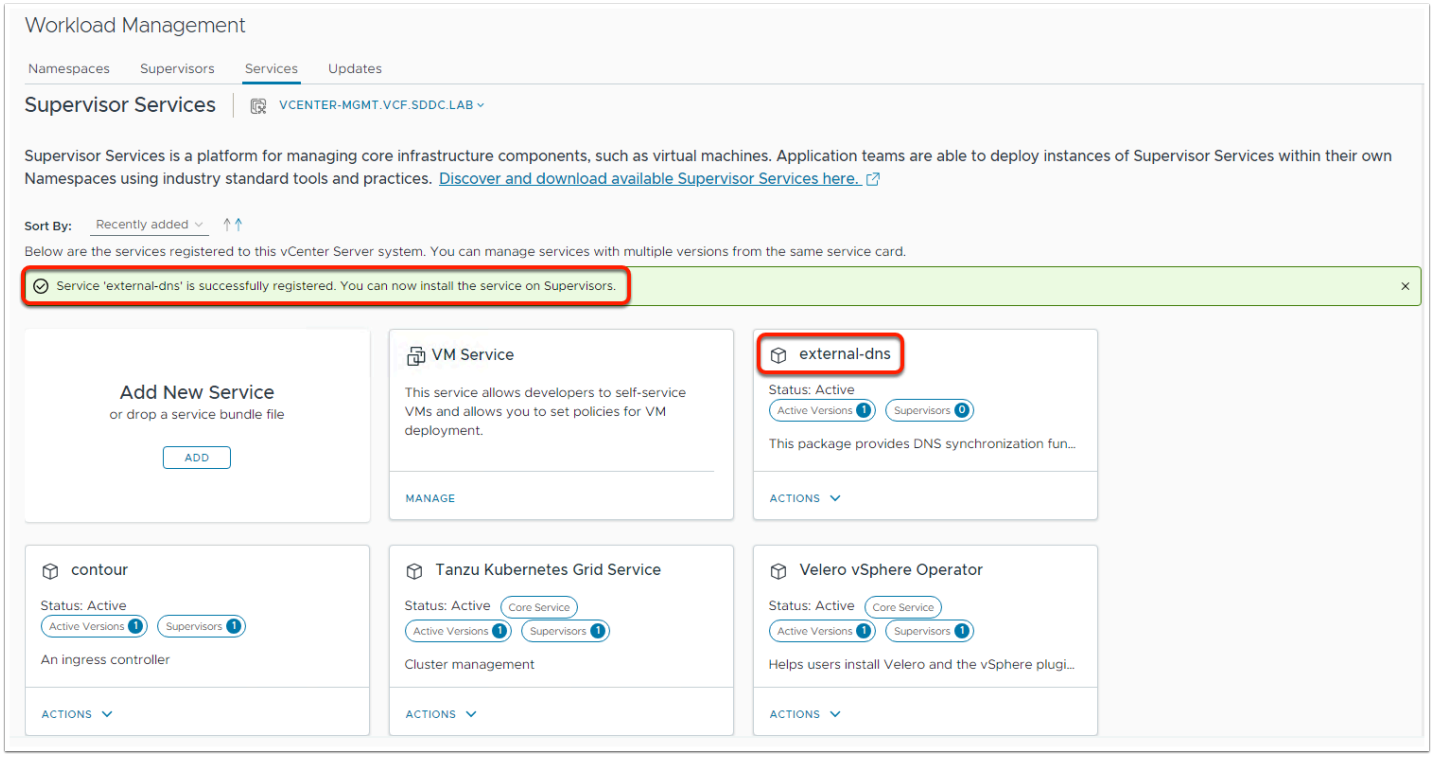


In the **New Service** wizard, for **Register Service** make sure you can see the aforementioned **external-dns.yml** file.

Its displayed version should match the one you downloaded.

Confirm with **FINISH**

Install ExternalDNS



In a while you will see a displayed message for the ExternalDNS service successfully registered with your Supervisor.

You need to activate it with particular configuration settings.

You can safely close the message in green.

Install ExternalDNS

ExternalDNS data `values.yaml`

- Because of the large list of supported DNS providers, we do not supply complete sample configuration values here. If you're deploying ExternalDNS with Harbor and Contour, make sure to include `source=contour-httpproxy` in the configuration values. An *incomplete* example of the service configuration is included below. Make sure to setup API access to your DNS server and include authentication details with the service configuration.

```
deployment:
  args:
  - --source=contour-httpproxy
  - --source=service
  - --log-level=debug
```

Normally you need to create the **ExternalDNS** data values manifest file on your own, to match your environment. But we have done that for you.

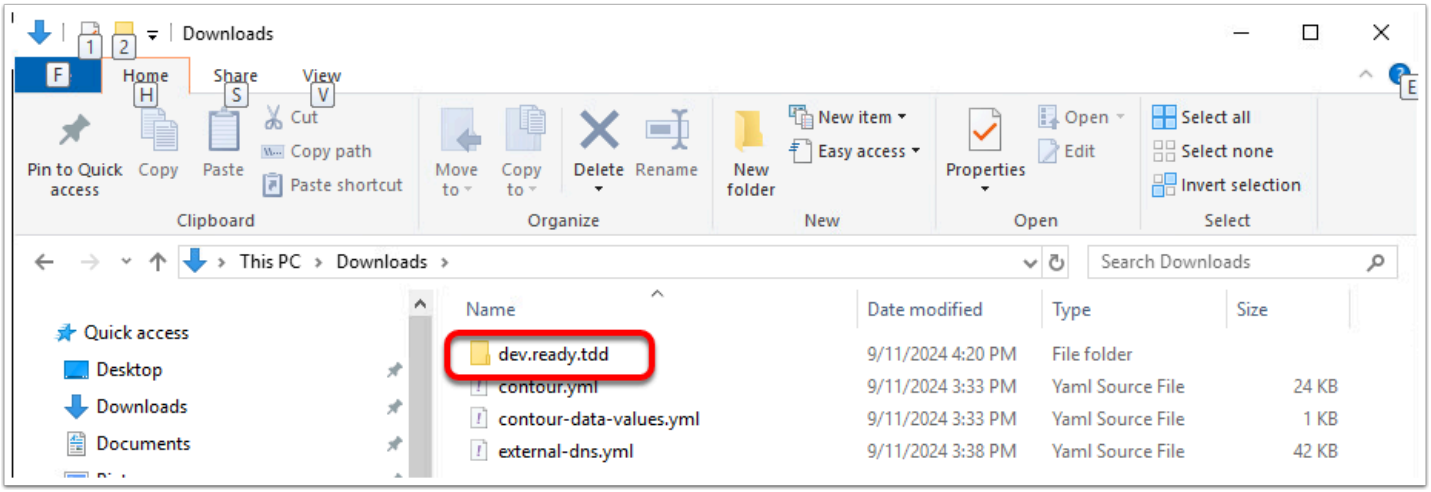
Install ExternalDNS

C: > Users > Administrator > Downloads > external-dns-data-values.yml

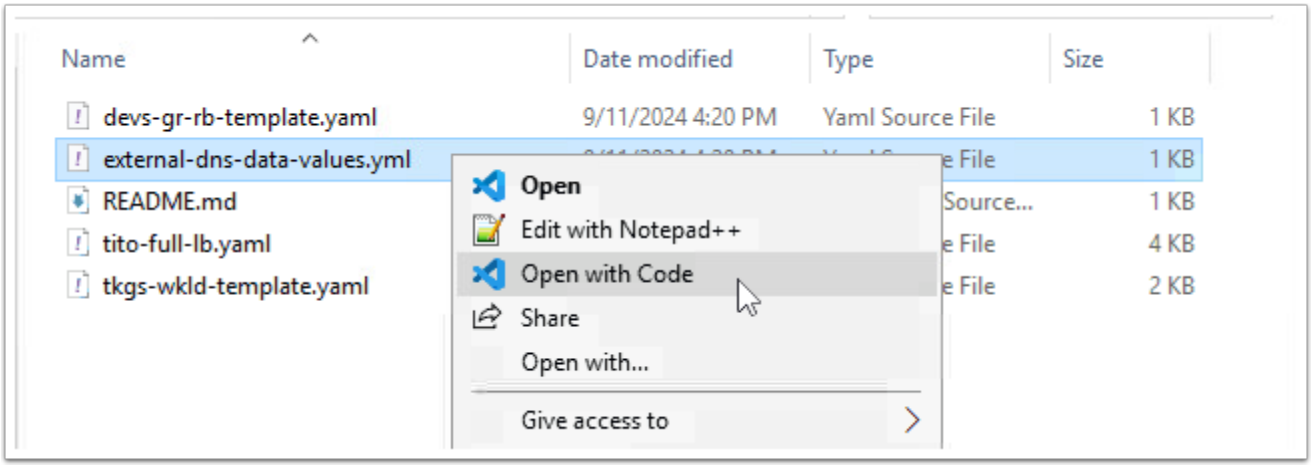
```
1  deployment:
2    args:
3    - --registry=txt
4    - --txt-prefix=external-dns-
5    - --txt-owner-id=tanzu
6    - --provider=rfc2136
7    - --rfc2136-host=10.0.0.201
8    - --rfc2136-port=53
9    - --rfc2136-zone=vcf.holo.lab
10   - --rfc2136-insecure
11   - --rfc2136-tsig-axfr
12   - --source=service
13   - --source=contour-httpproxy
14   - --source=ingress
15   - --domain-filter=vcf.holo.lab
16 namespace: svc-external-dns-domain-c9
```

The github repository you cloned earlier has the file you need to deploy this service to the Supervisor Cluster.

dev.ready.tdd

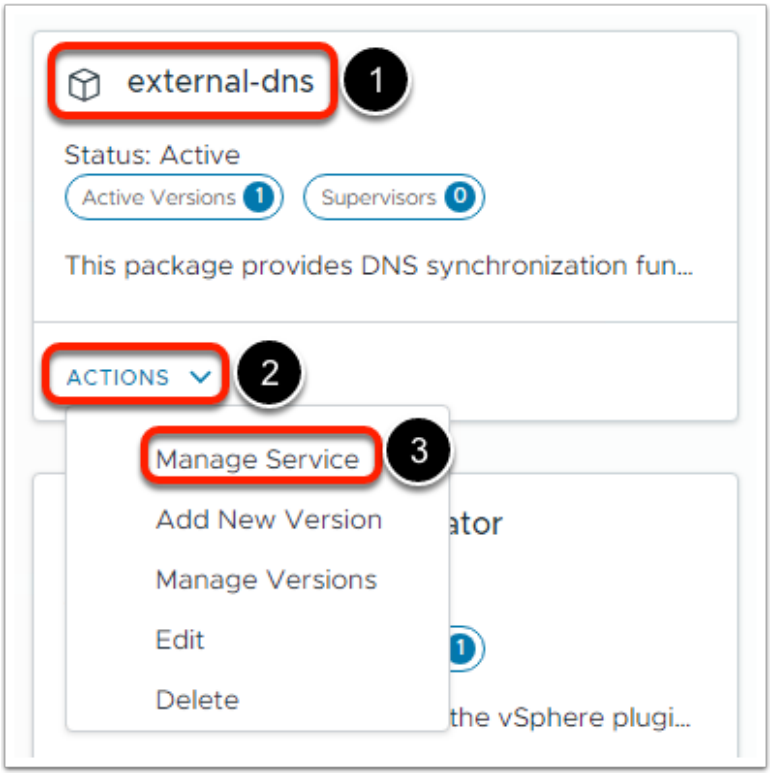


Have a look in this folder and you will find the **external-dns-data-values.yml** file that you need for this part of the lab. Remember this folder for later in the labs



Right click and open this file in VSCode. Go back to vCenter and add the external-dns service to the supervisor cluster using the values in your clipboard

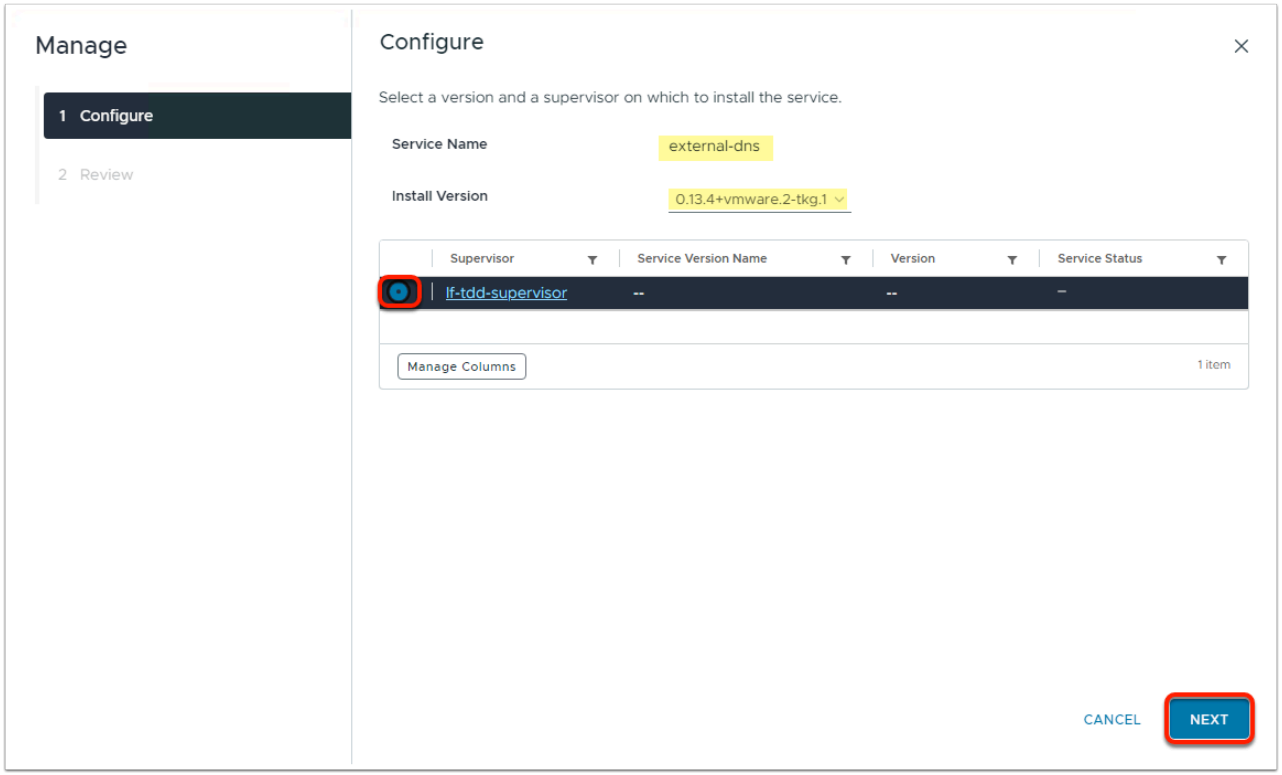
Install ExternalDNS



Go back to Chrome -> vSphere UI -> Workload Management -> Services

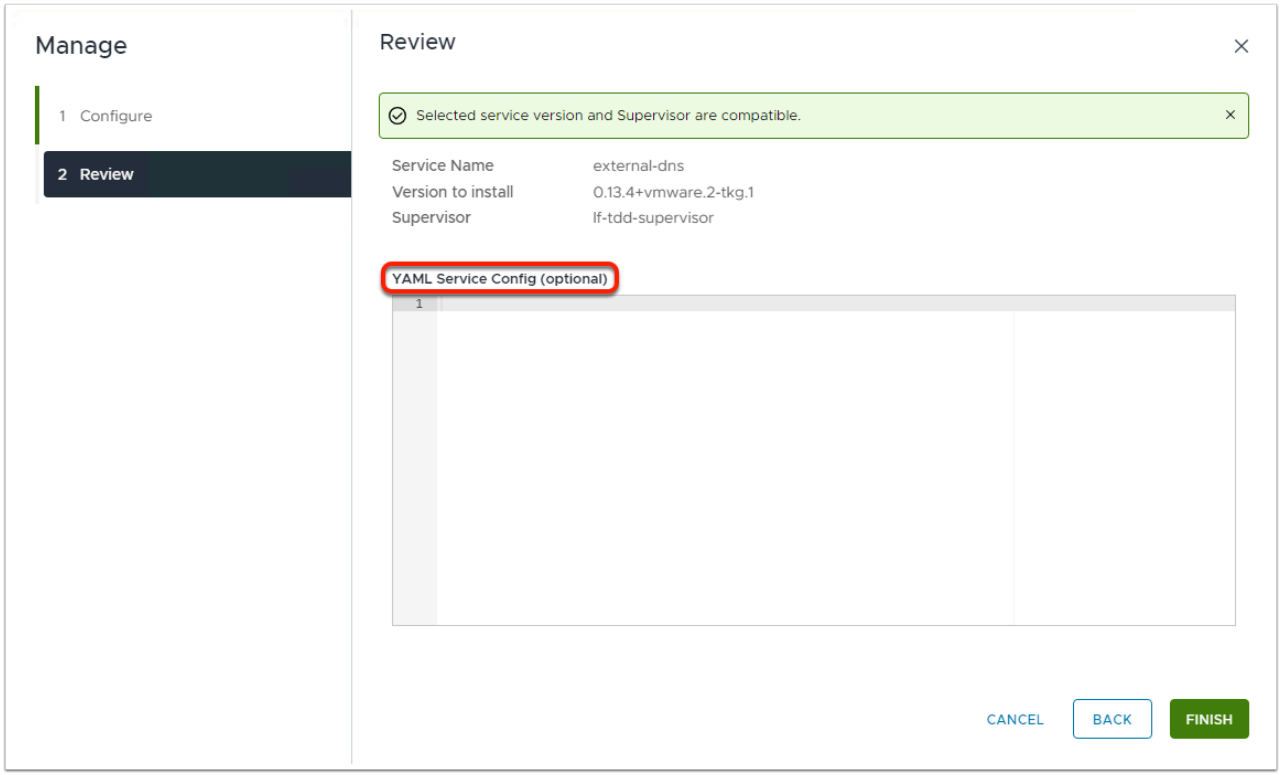
1. Locate your recently registered **external-dns** service
2. From **ACTIONS** drop down menu
3. select **Manage Service**

Install ExternalDNS



In the **Manage** wizard, from **Configure** select your Supervisor and confirm **NEXT**

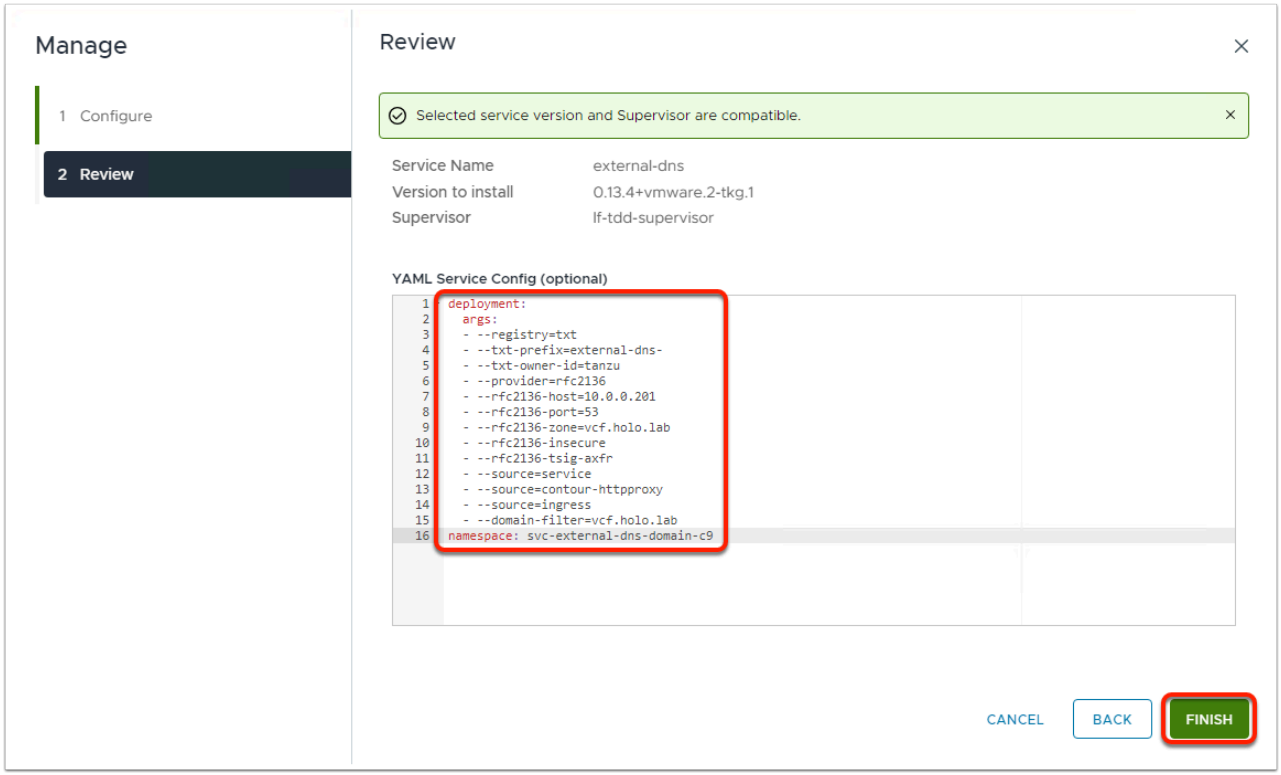
Install ExternalDNS



You need to provide the service configuration values.

Copy these from the Visual Studio Code, where you lastly edited **external-dns-data-values.yml**

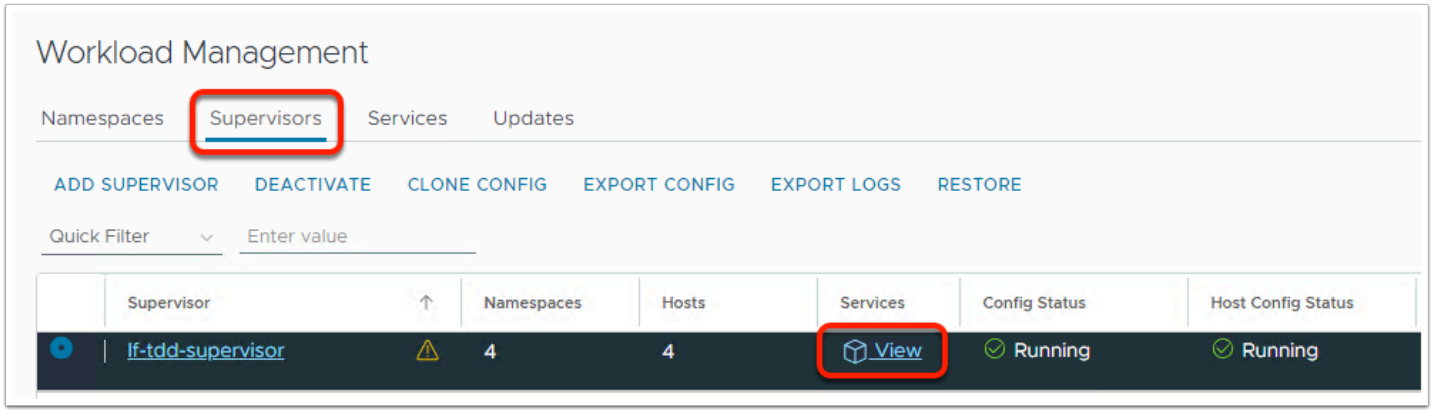
Install ExternalDNS



Select all and paste into the **YAML Service Config** section in the UI wizard.

Confirm with **FINISH**

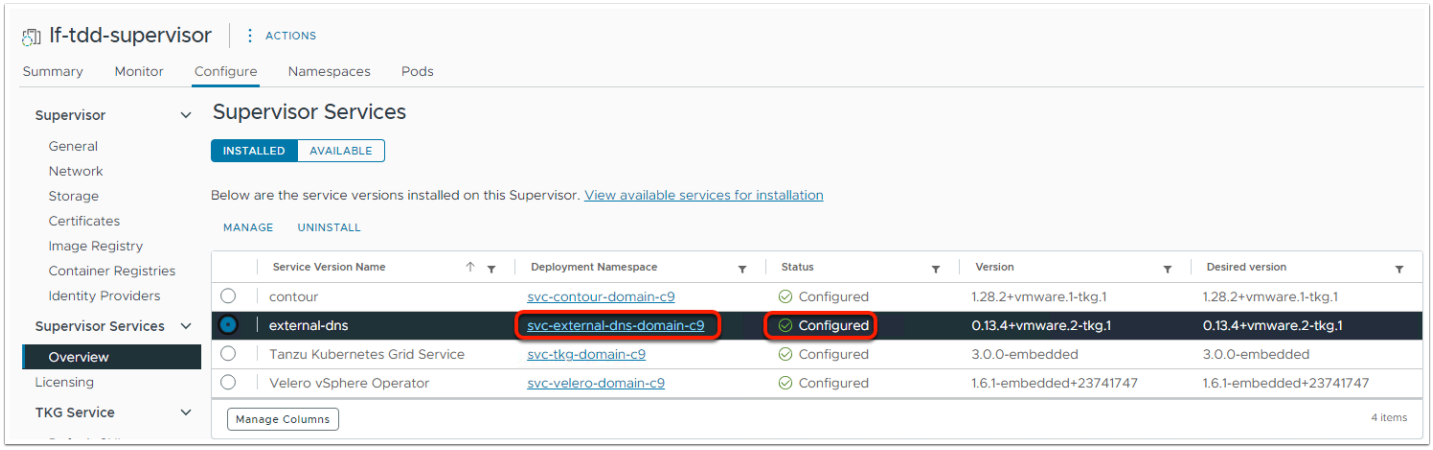
Install ExternalDNS



Go back in the **Workload Management** section

Select **Supervisors** and click on **View** under **Services**

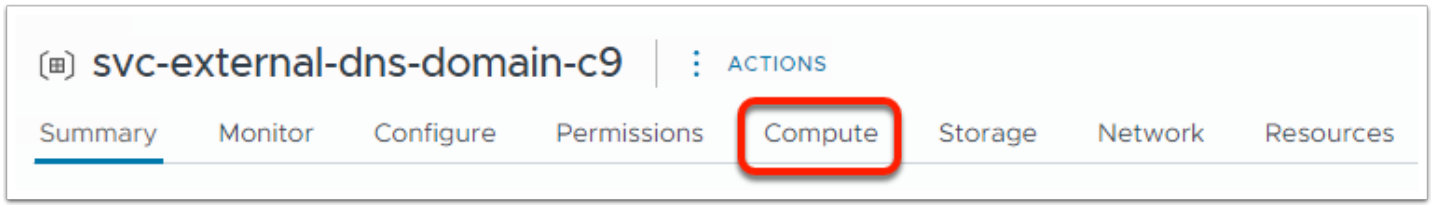
Install ExternalDNS



In a while the **external-dns** service Status will change from Configuring to **Configured**

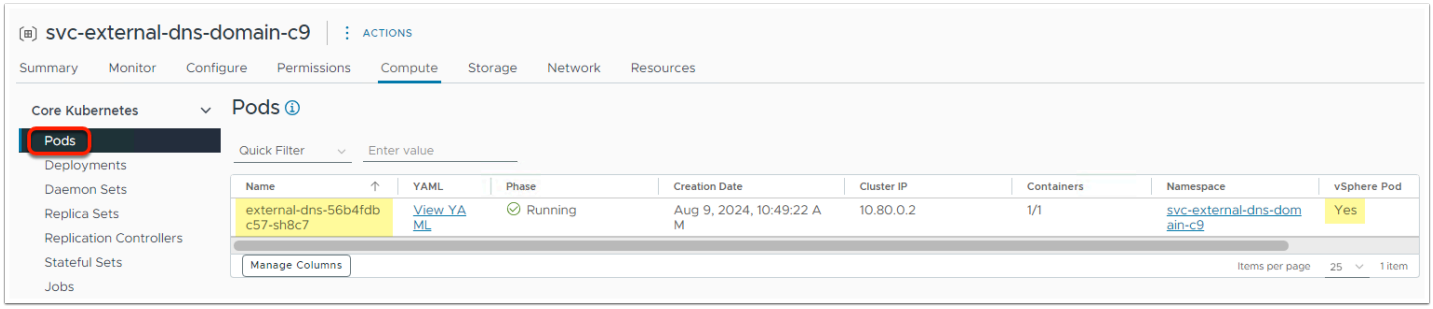
Click on the **svc-external-dns-domain-c9** under **Deployment Namespace**

Install ExternalDNS



For the **svc-external-dns-domain-c9** Namespace, click on **Compute**

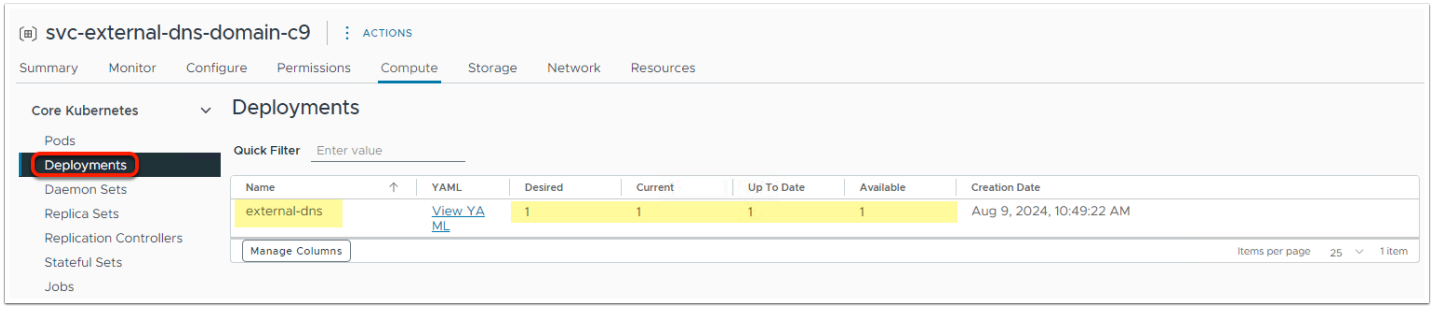
Install ExternalDNS



Under **Compute**, you can review all the **Kubernetes Core** configurations for ExternalDNS.

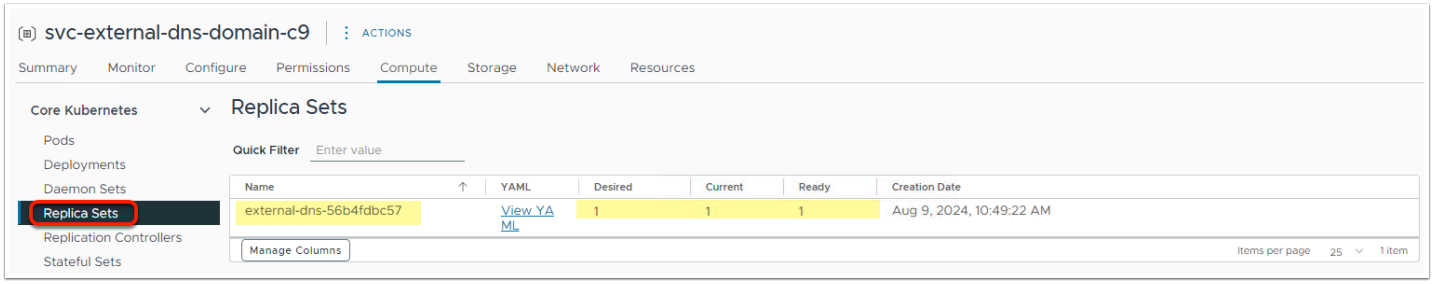
Select **Pods** and you will see all the **ExternalDNS** needed pods, running as **vSphere Pods**

Install ExternalDNS



Select Deployments and you will see all the **external-dns** needed deployments in place

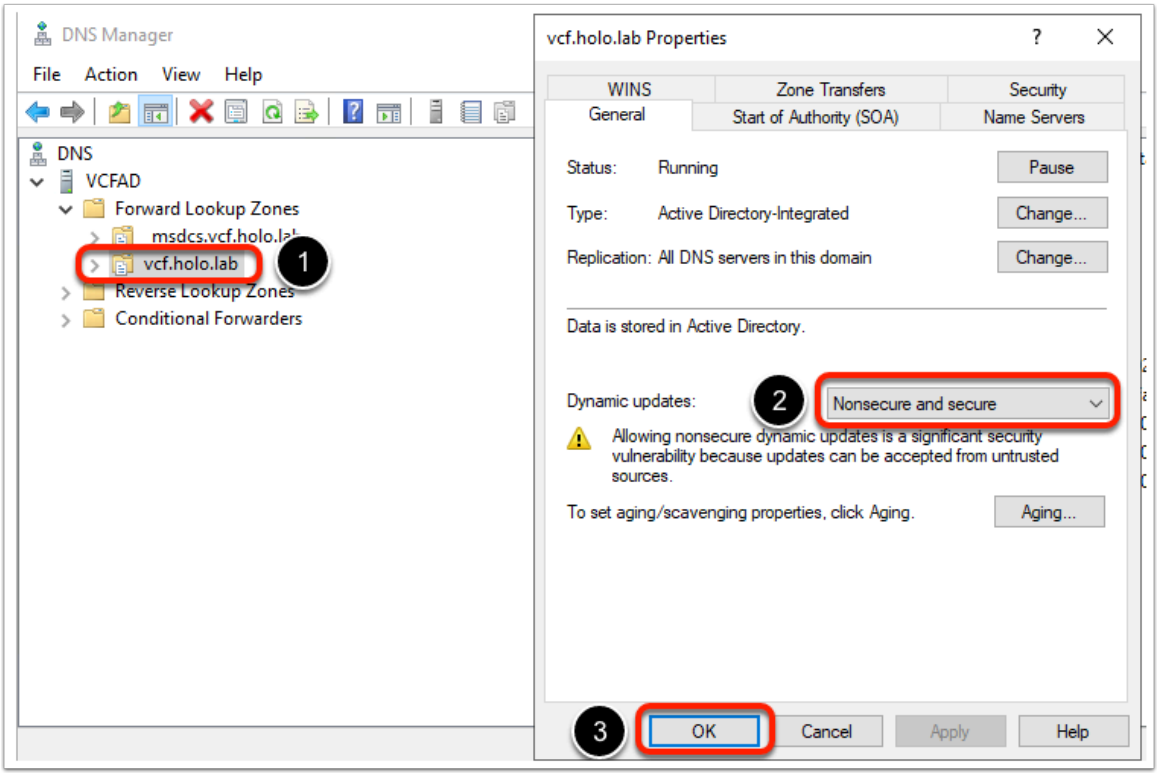
Install ExternalDNS



Select Replica Sets and you will see all the ExternalDNS needed replica sets in place

> ⚠ Please, note: With the configuration provided, ExternalDNS is on par with RFC2139 , hence you might need to consider the default behaviour, which is providing non-secure dynamic Updates in Domain Name System.

Install ExternalDNS



From your Windows desktop: start the DNS Manager, expand the Forward Lookup Zones under VCFAD

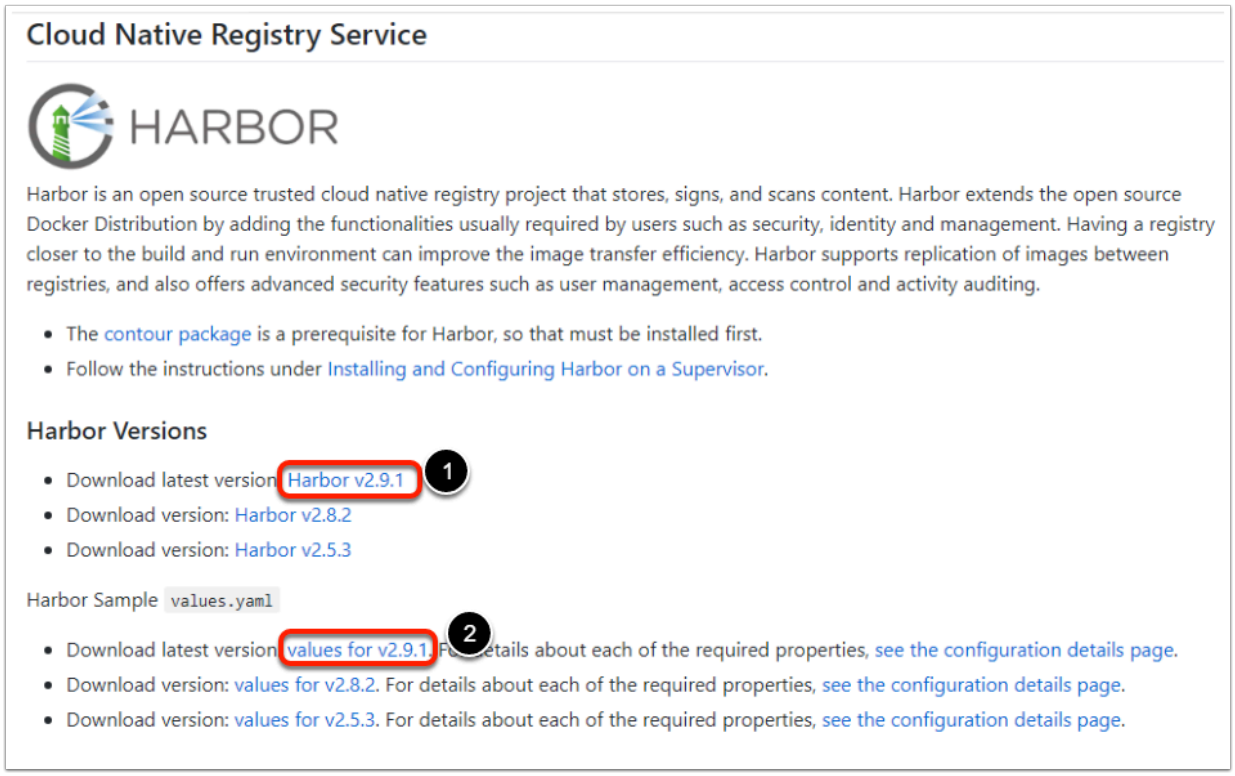
1. right click on the **vcf.holo.lab** domain, select **Properties**
2. in the **Properties** window, under **General**, Change the **Dynamic updates** option to **Nonsecure and secure**
3. confirm with **OK**

Now the DNS server is ready to accept Dynamic updates from our **external-dns** service.

Install Cloud Native Registry Service (Harbor)

For a **Cloud Native Registry Service** you can use **Harbor**.

Harbor is an open source trusted cloud native registry project that stores, signs, and scans content. Harbor extends the open source Docker Distribution by adding the functionalities usually required by users such as security, identity and management. Having a registry closer to the build and run environment can improve the image transfer efficiency. Harbor supports replication of images between registries, and also offers advanced security features such as user management, access control and activity auditing.



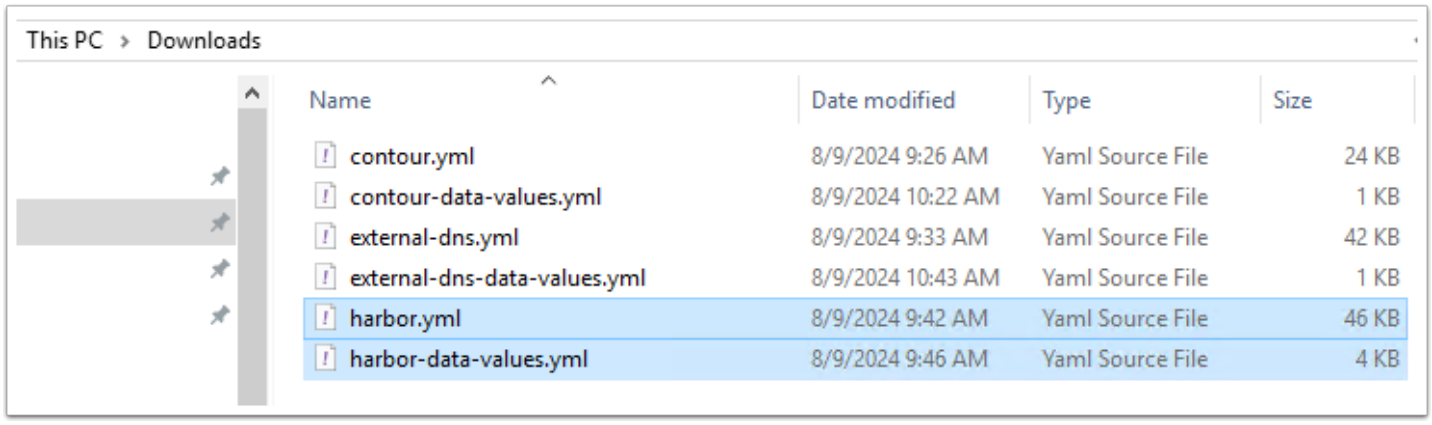
Download the latest available manifests for **harbor** to your **Downloads** folder

1. harbor.yml

2. harbor-data-values.yml

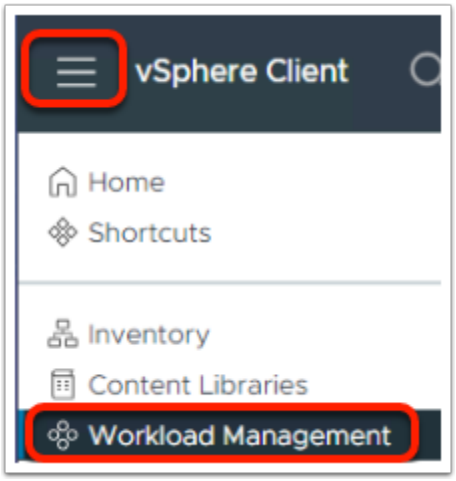
https://vsphere-tmm.github.io/Supervisor-Services/#cloud-native-registry-service

Install Cloud Native Registry Service (Harbor)



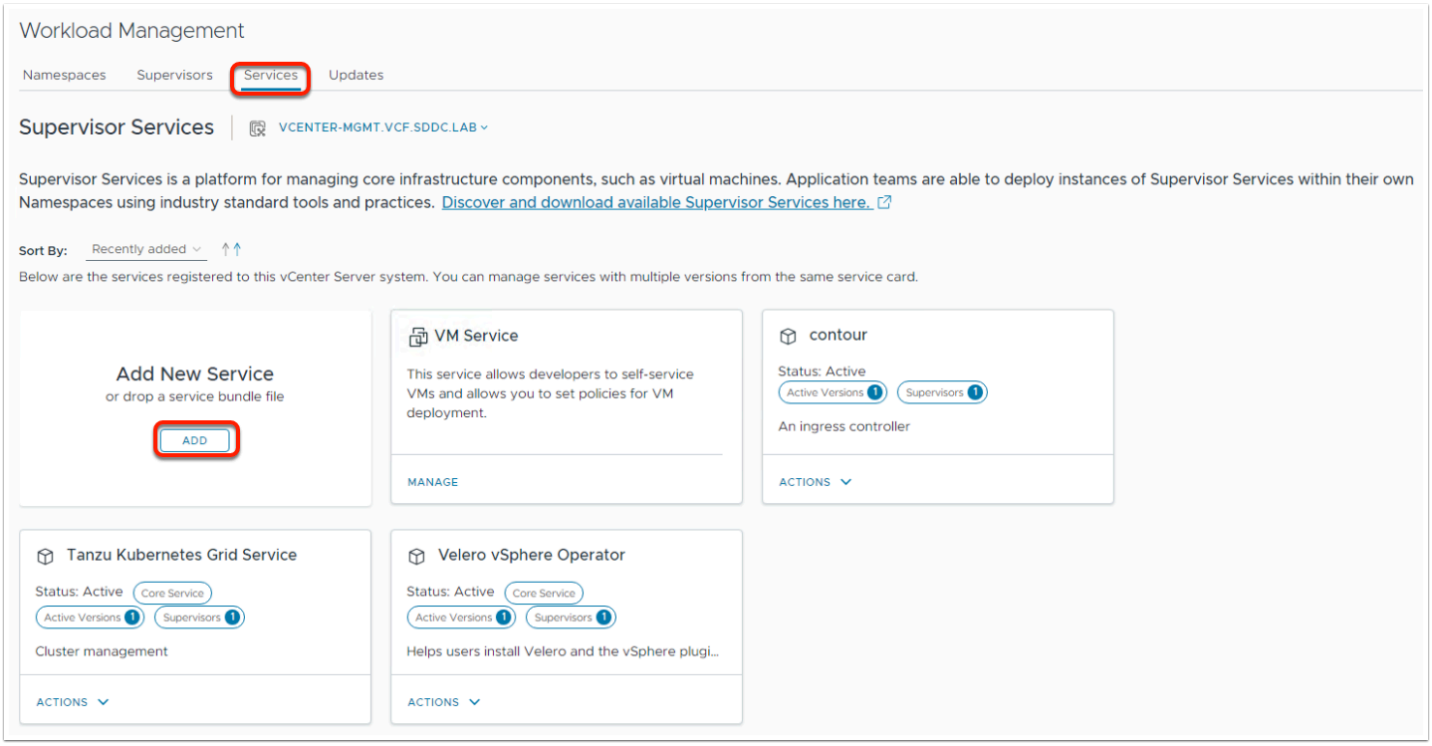
You are ready to deploy Harbor as a Supervisor service!

Install Cloud Native Registry Service (Harbor)



From the vSphere Client home menu, select **Workload Management**.

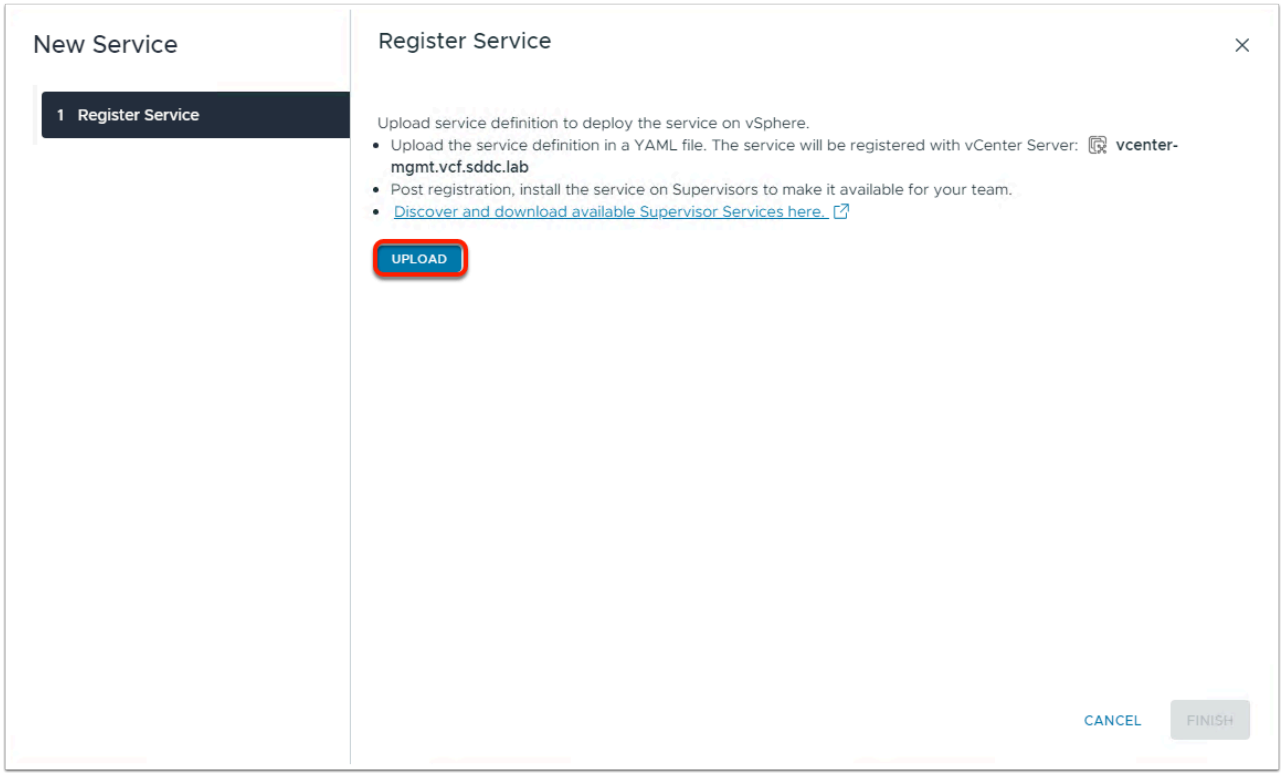
Install Cloud Native Registry Service (Harbor)



In the **Workload Management** section, select **Services**.

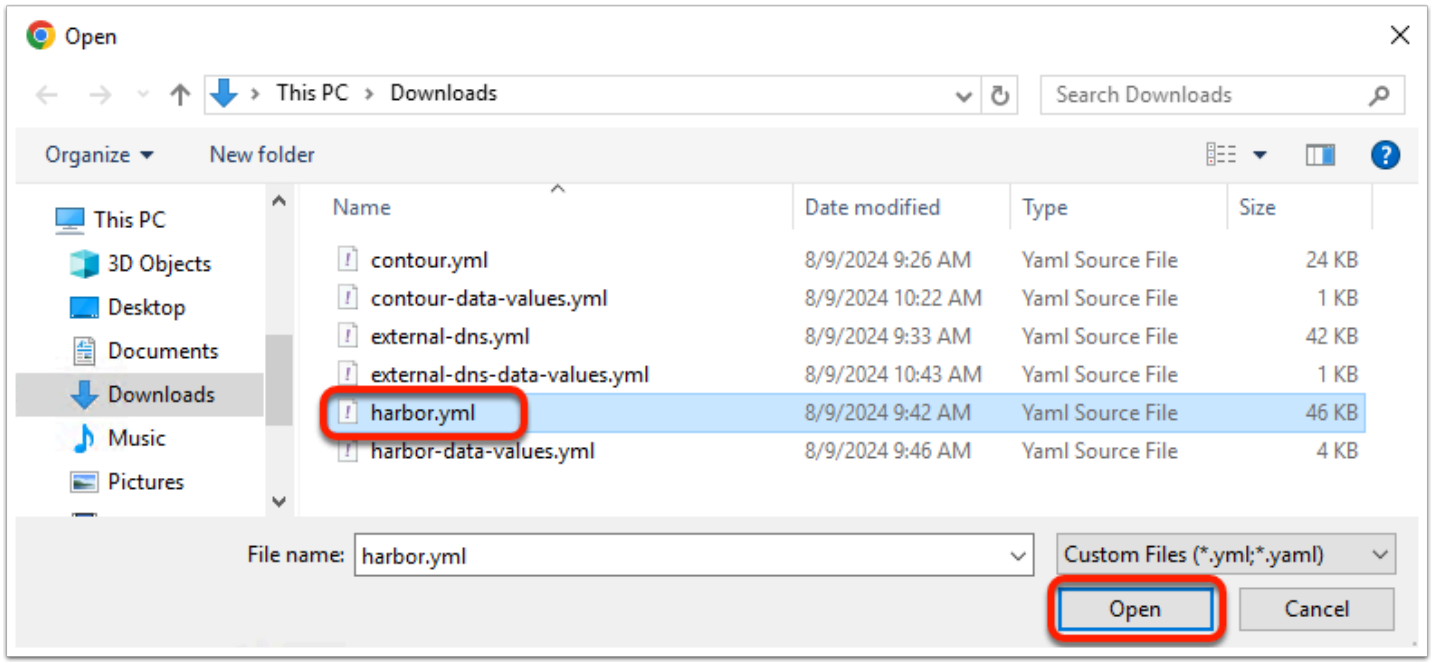
Locate the **Add New Service** card and click on **ADD**

Install Cloud Native Registry Service (Harbor)



In the **New Service** wizard, for **Register Service** select **UPLOAD**

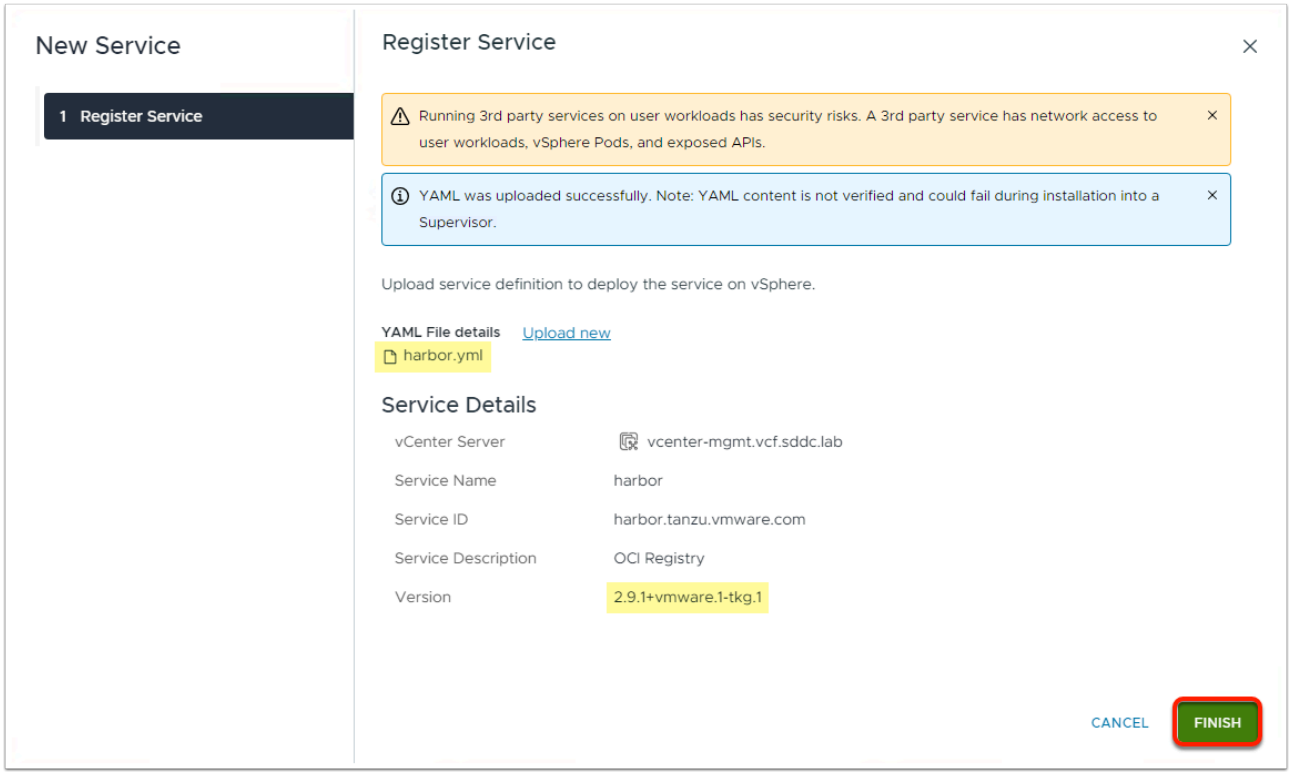
Install Cloud Native Registry Service (Harbor)



In the Chrome **Open** menu go to **Downloads** folder and select the **harbor.yml** you downloaded recently.

Confirm with clicking on **Open** button

Install Cloud Native Registry Service (Harbor)

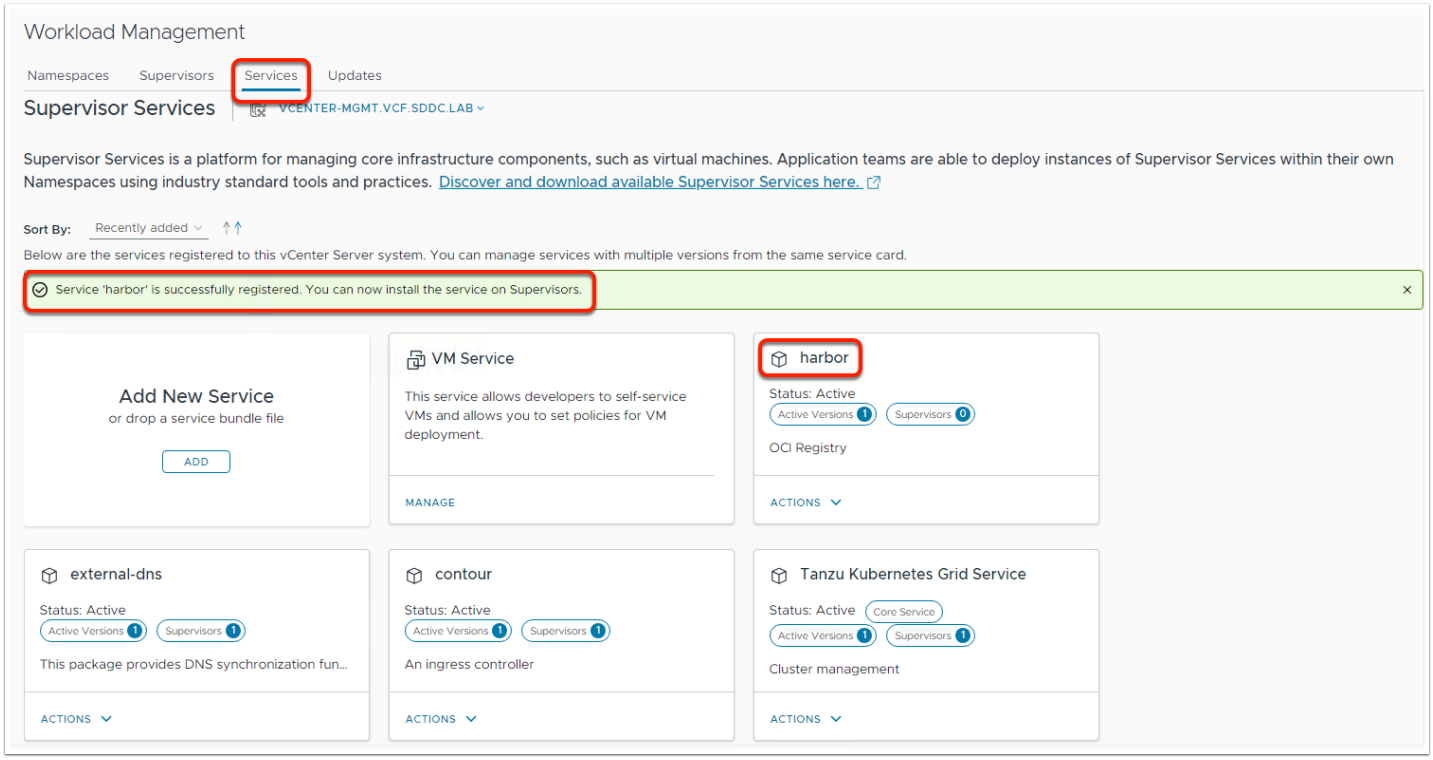


In the **New Service** wizard, for **Register Service** make sure you can see the aforementioned **harbor.yml** file.

Its displayed version should match the one you downloaded.

Confirm with **FINISH**

Install Cloud Native Registry Service (Harbor)

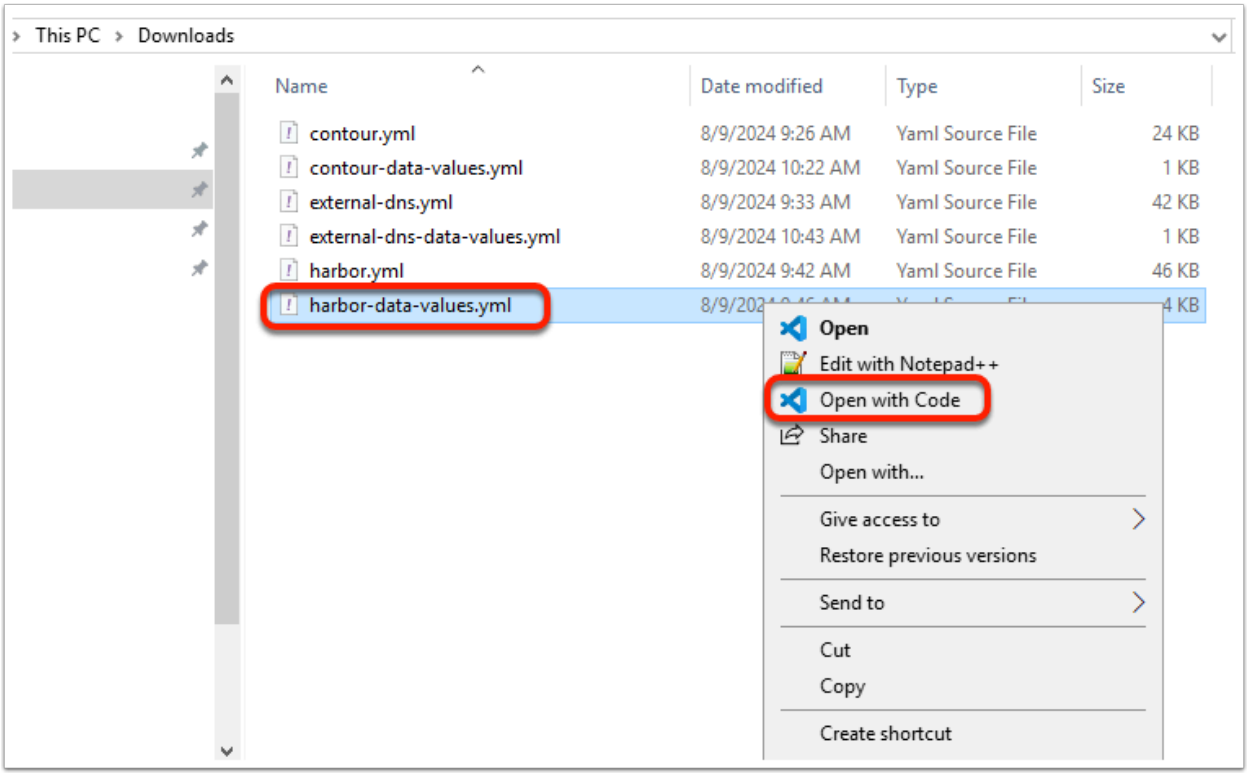


In a while you will see a displayed message for the **harbor** service successfully registered with your **Supervisor**.

You need to activate it with particular configuration settings.

You can close the message in green field.

Install Cloud Native Registry Service (Harbor)



Let's examine the particular file. You can review/edit it with any available editor

In Windows Explorer, navigate to your **Downloads** folder and locate the recently downloaded file **harbor-data-values.yml**.

Right click and **Open with Code**

For Harbor data values you need to edit a little bit.

Install Cloud Native Registry Service (Harbor)

C: > Users > Administrator > Downloads > harbor-data-values.yml > ...

```yaml
#! The FQDN for accessing Harbor admin UI and Registry service.
hostname: harbor.vcf.holo.lab
#! The network port of the Envoy service in Contour or other Ingress Controller.
port:
  https: 443

#! Do not change tlsSecretLabels. It is required for TKG  integration to work.
tlsCertificate:
  tlsSecretLabels: {"managed-by": "vmware-vRegistry"}

#! [Required] The initial password of Harbor admin.
harborAdminPassword: VMware1!

#! [Required] The secret key used for encryption. Must be a string of 16 chars.
secretKey: 0123456789ABCDEF

database:
  #! if external database is used, set "type" to "external"
  #! and fill the connection information in "external" section
  type: internal
  #! [Required] The initial password of the internal postgres database.
  #! if external database is used, please fill the external.password
  password: change-it

core:
  replicas: 1
  #! [Required] Secret is used when core server communicates with other components.
  secret: change-it
  #! [Required] The XSRF key. Must be a string of 32 chars.
  xsrfKey: 0123456789ABCDEF0123456789ABCDEF
jobservice:
  replicas: 1
  #! [Required] Secret is used when job service communicates with other components.
  secret: change-it
registry:
  replicas: 1
  #! [Required] Secret is used to secure the upload state from client
  #! and registry storage backend.
  #! See: https://github.com/docker/distribution/blob/master/docs/configuration.md#http
  secret: change-it

```

C: > Users > Administrator > Downloads > harbor-data-values.yml > ...

```yaml
#! The persistence is always enabled and a default StorageClass
#! is needed in the k8s cluster to provision volumes dynamically.
#! Specify another StorageClass in the "storageClass" or set "existingClaim"
#! if you have already existing persistent volumes to use
#!
#! For storing images and charts, you can also use "azure", "gcs", "s3",
#! "swift" or "oss". Set it in the "imageChartStorage" section
persistence:
  persistentVolumeClaim:
    registry:
      #! Use the existing PVC which must be created manually before bound,
      #! and specify the "subPath" if the PVC is shared with other components
      existingClaim: ""
      #! Specify the "storageClass" used to provision the volume. Or the default
      #! StorageClass will be used(the default).
      #! Set it to "-" to disable dynamic provisioning
      storageClass: "vsan-tanzu-storage"
      subPath: ""
      accessMode: ReadWriteOnce
      size: 10Gi
    jobservice:
      jobLog:
        existingClaim: ""
        storageClass: "vsan-tanzu-storage"
        subPath: ""
        accessMode: ReadWriteOnce
        size: 1Gi
    database:
      existingClaim: ""
      storageClass: "vsan-tanzu-storage"
      subPath: ""
      accessMode: ReadWriteOnce
      size: 1Gi
    redis:
      existingClaim: ""
      storageClass: "vsan-tanzu-storage"
      subPath: ""
      accessMode: ReadWriteOnce
      size: 1Gi
    trivy:
      existingClaim: ""
      storageClass: "vsan-tanzu-storage"
      subPath: ""
      accessMode: ReadWriteOnce
      size: 5Gi

```

You need to edit accordingly, from the default values:

```
hostname: yourdomain.com

harborAdminPassword: Harbor12345

storageClass: "insert-storage-class-name-here"
```

to new values as following:

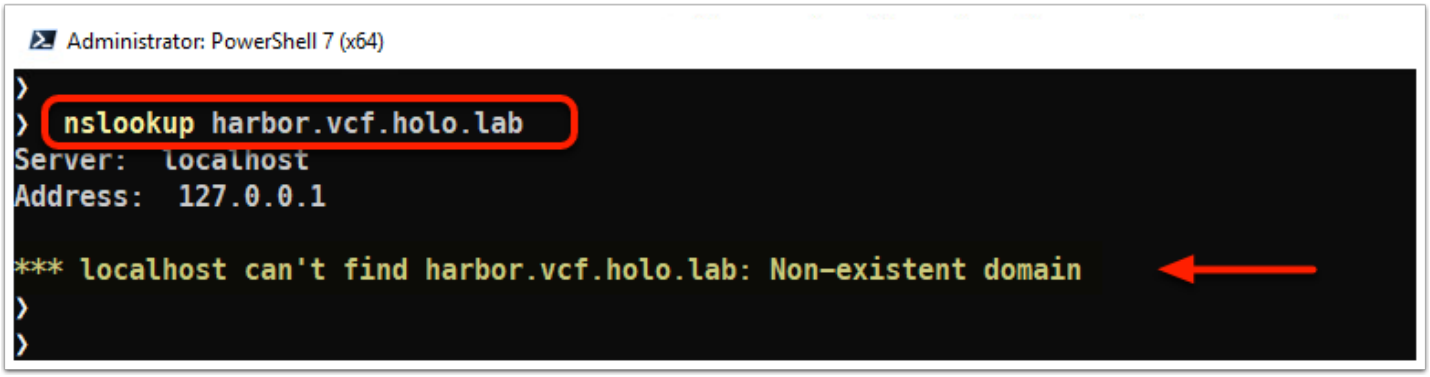
```
hostname: harbor.vcf.holo.lab

harborAdminPassword: VMware1!

storageClass: "vsan-tanzu-storage"
```

Save the file!

Don't close it, you will need the values shortly!

Install Cloud Native Registry Service (Harbor)

> OPTIONAL: Check there is no DNS record for harbor service, available before its install:



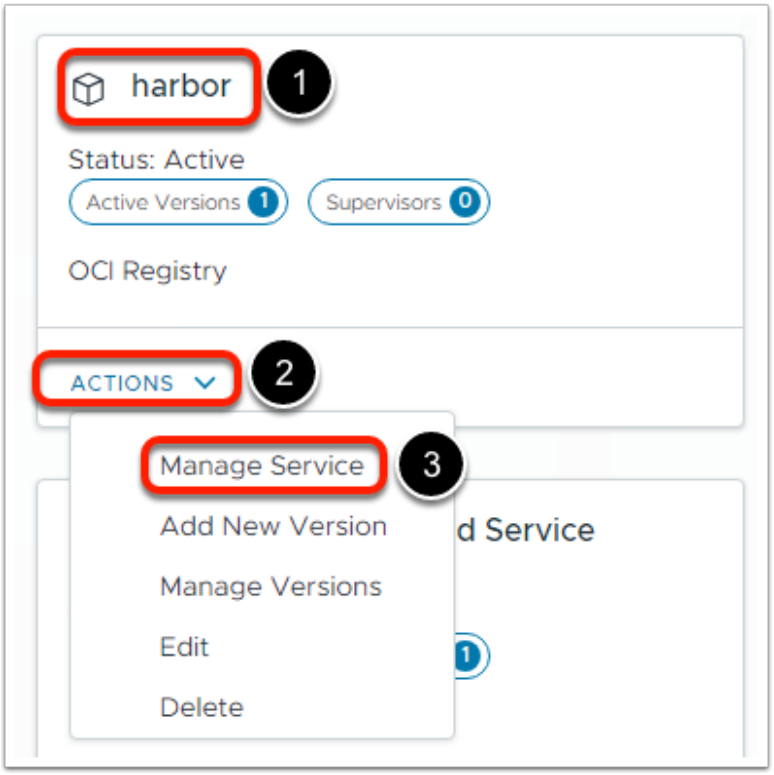
Open a Command Prompt in your Windows desktop

enter the check command

```
nslookup harbor.vcf.holo.lab
```

Once the harbor install is finished, you can check if external-dns has created the dynamic DNS update for it.

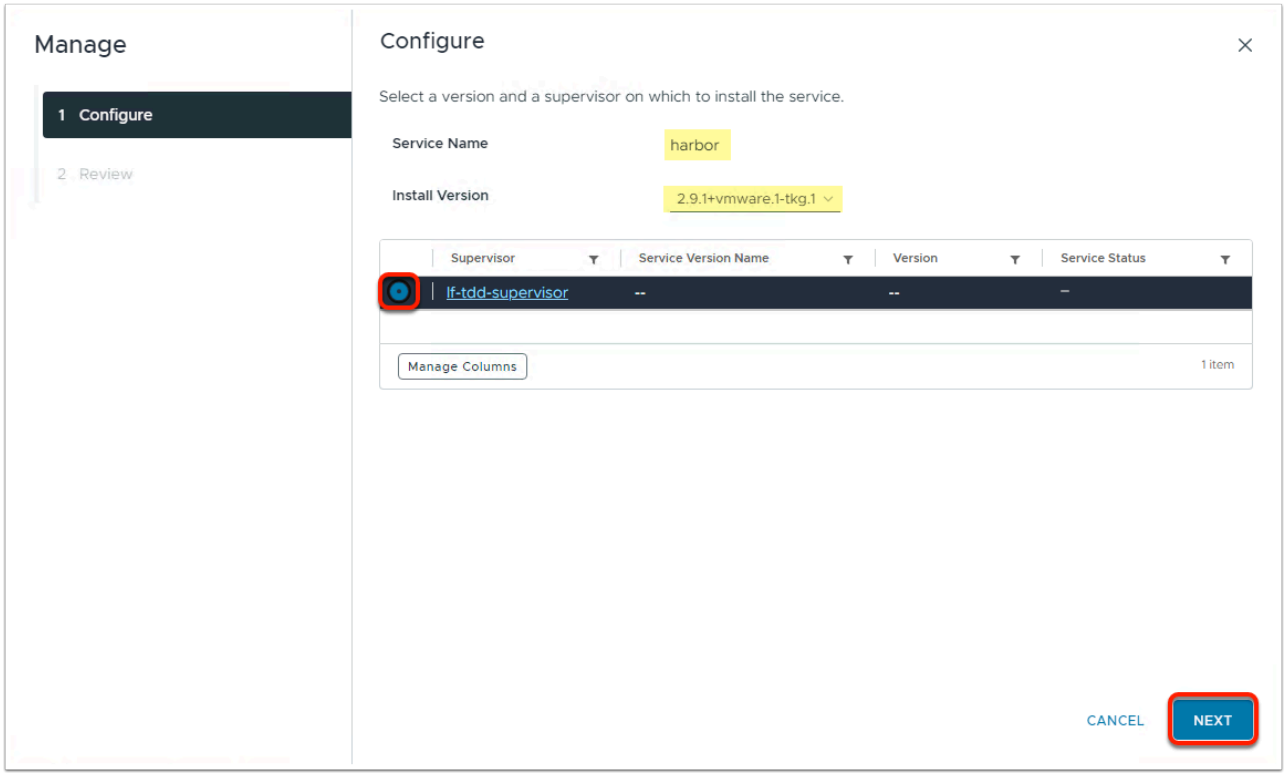
Install Cloud Native Registry Service (Harbor)



Go back to Chrome -> vSphere UI -> Workload Management -> Services

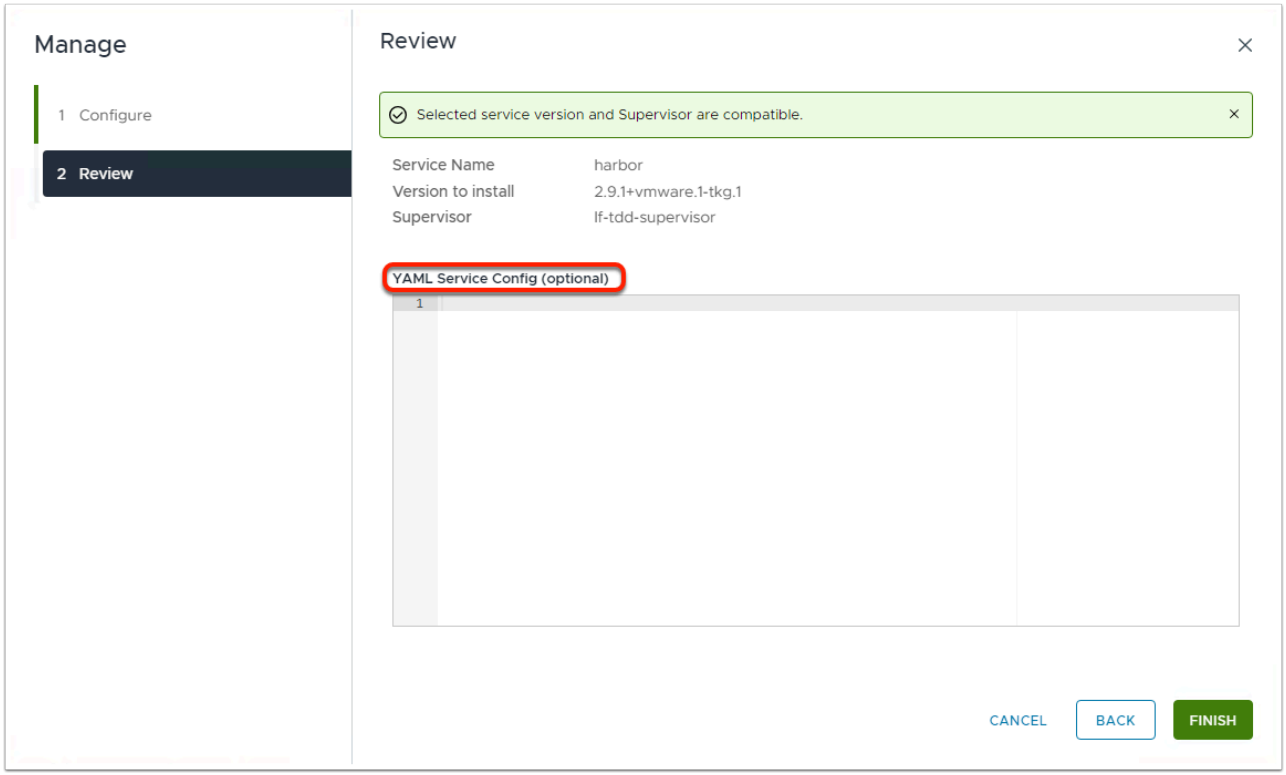
1. Locate your recently registered **harbor** service
2. From **Actions** drop down menu
3. select **Manage Service**

Install Cloud Native Registry Service (Harbor)



In the **Manage** wizard, from **Configure** select your **Supervisor** and confirm **NEXT**

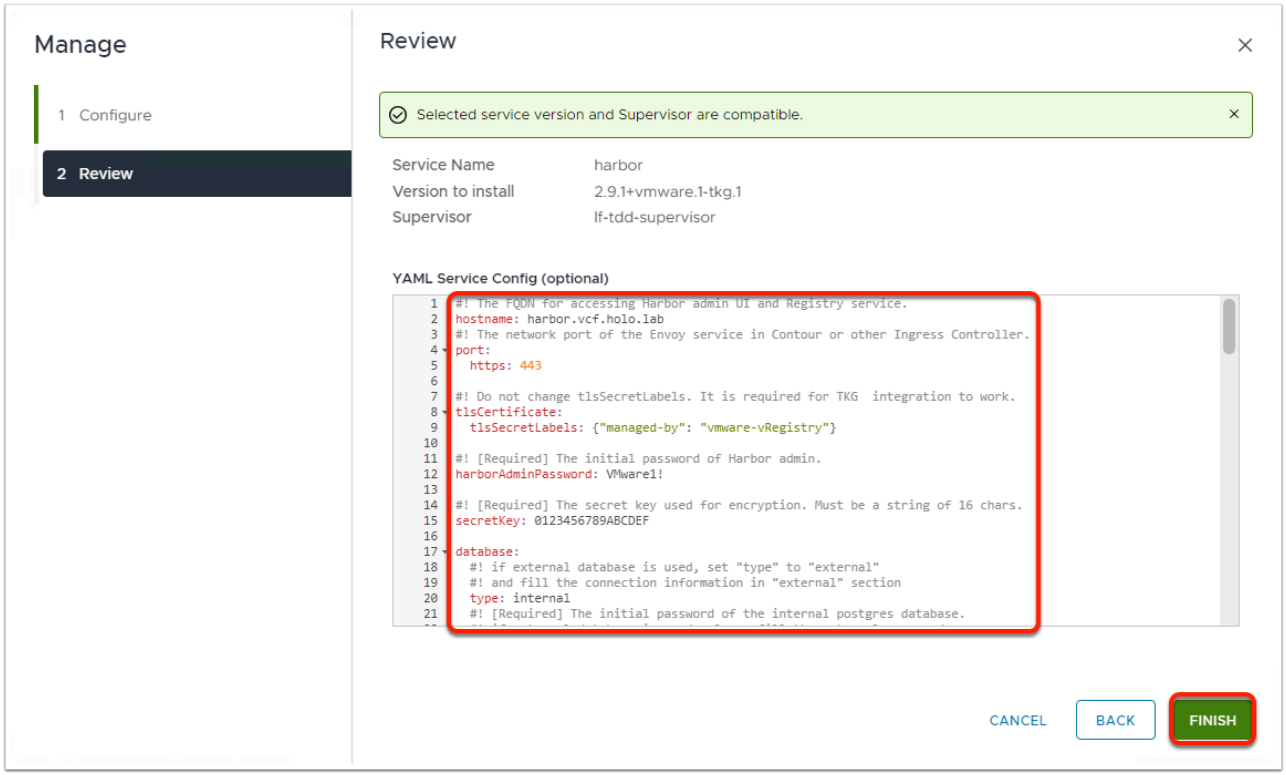
Install Cloud Native Registry Service (Harbor)



You need to provide the service configuration values.

Copy these from the Visual Studio Code, where you lastly reviewed **harbor-data-values.yml**

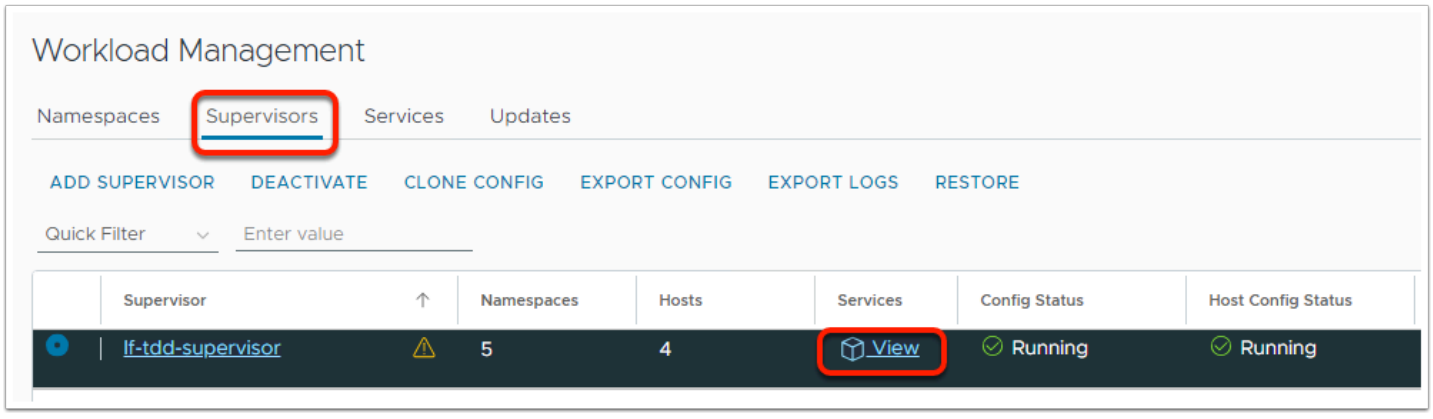
Install Cloud Native Registry Service (Harbor)



Select all and paste into the **YAML Service Config** section in the UI wizard.

Confirm with **FINISH**

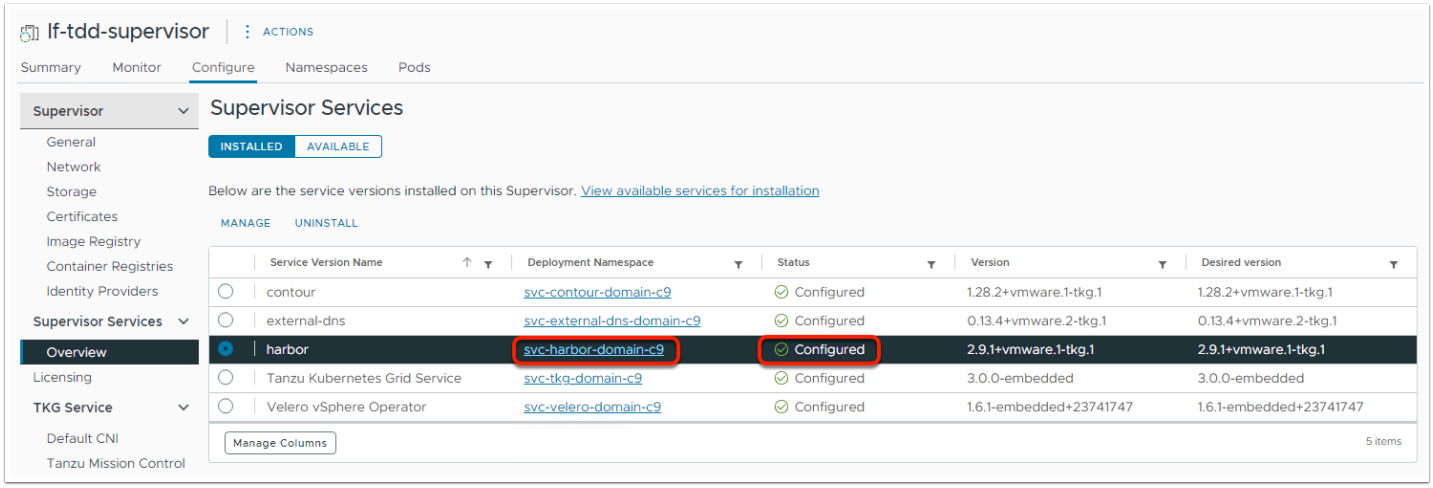
Install Cloud Native Registry Service (Harbor)



Go back in the **Workload Management** section

Select **Supervisors** and click on **View** under **Services**

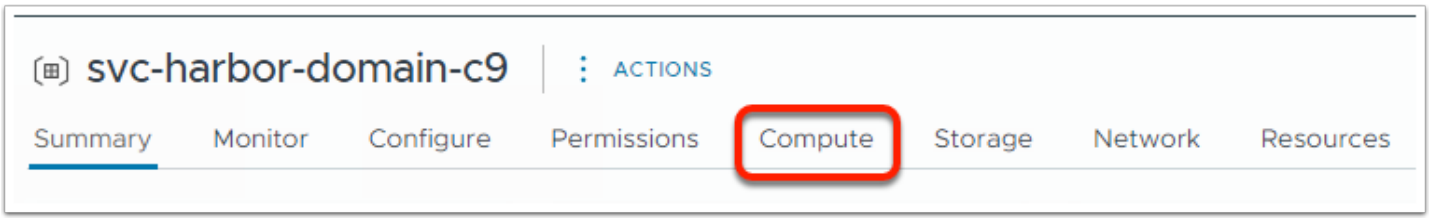
Install Cloud Native Registry Service (Harbor)



In a couple of minutes the **harbor** service Status will change from Configuring to **Configured**

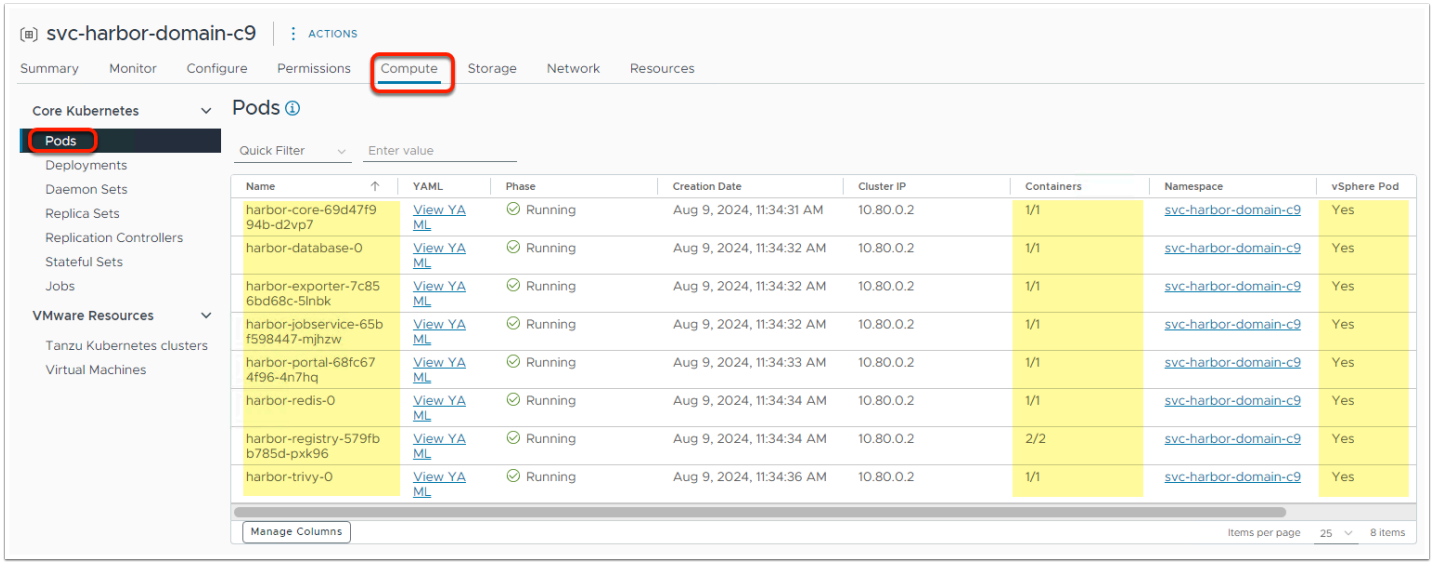
Click on the **svc-harbor-domain-c9** under **Deployment Namespace**

Install Cloud Native Registry Service (Harbor)



For the **svc-harbor-domain-c9** Namespace, click on **Compute**

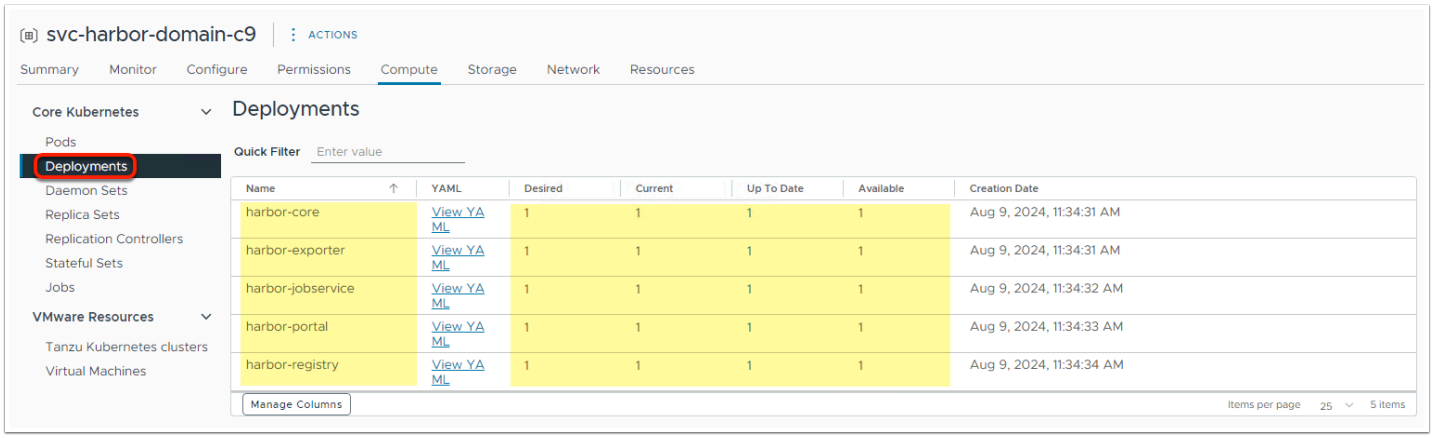
Install Cloud Native Registry Service (Harbor)



Under **Compute**, you can review all the **Kubernetes Core** configurations for harbor.

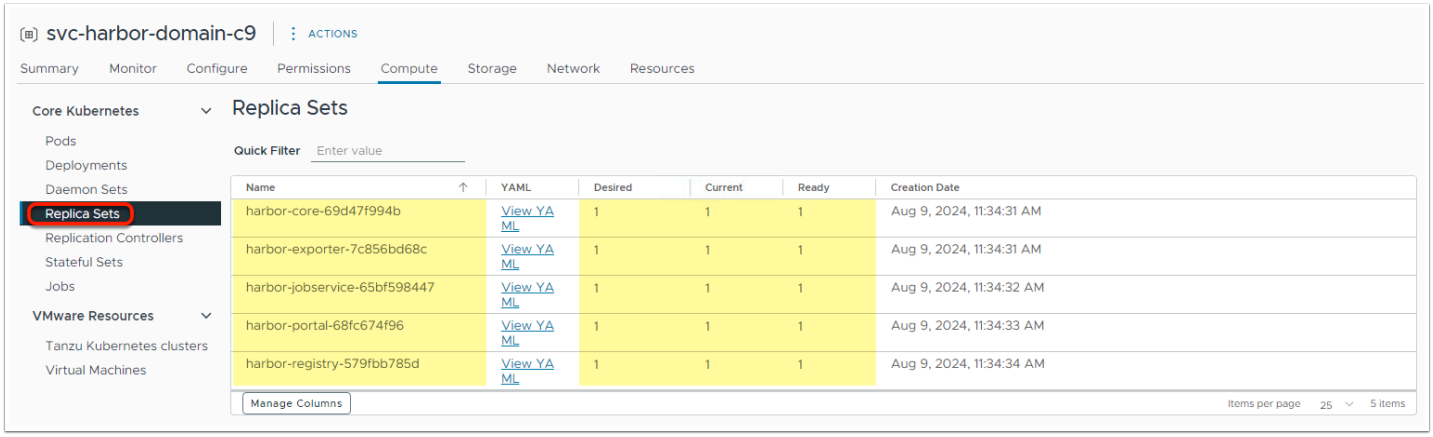
Select **Pods** and you will see all the **harbor** needed pods, running as **vSphere Pods**

Install Cloud Native Registry Service (Harbor)



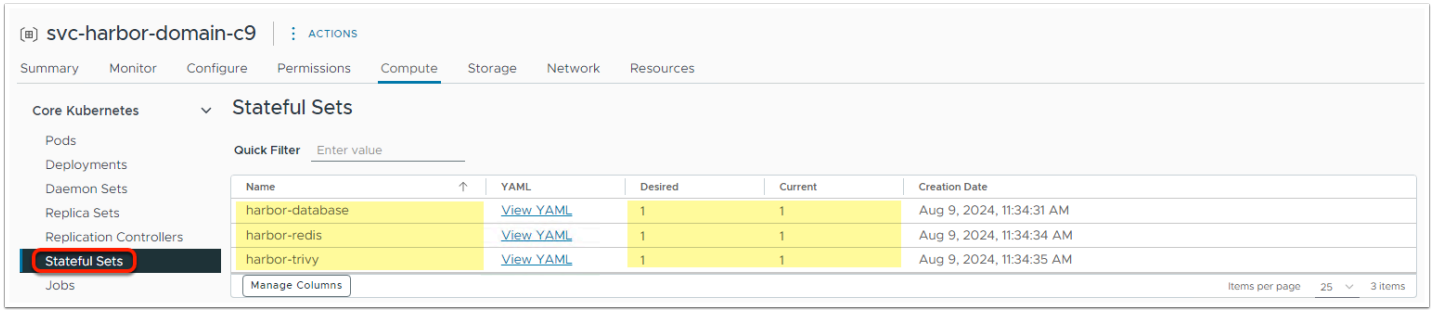
Select **Deployments** and you will see all the **harbor** needed deployments in place

Install Cloud Native Registry Service (Harbor)



Select **Replica Sets** and you will see all the **harbor** needed replica sets in place

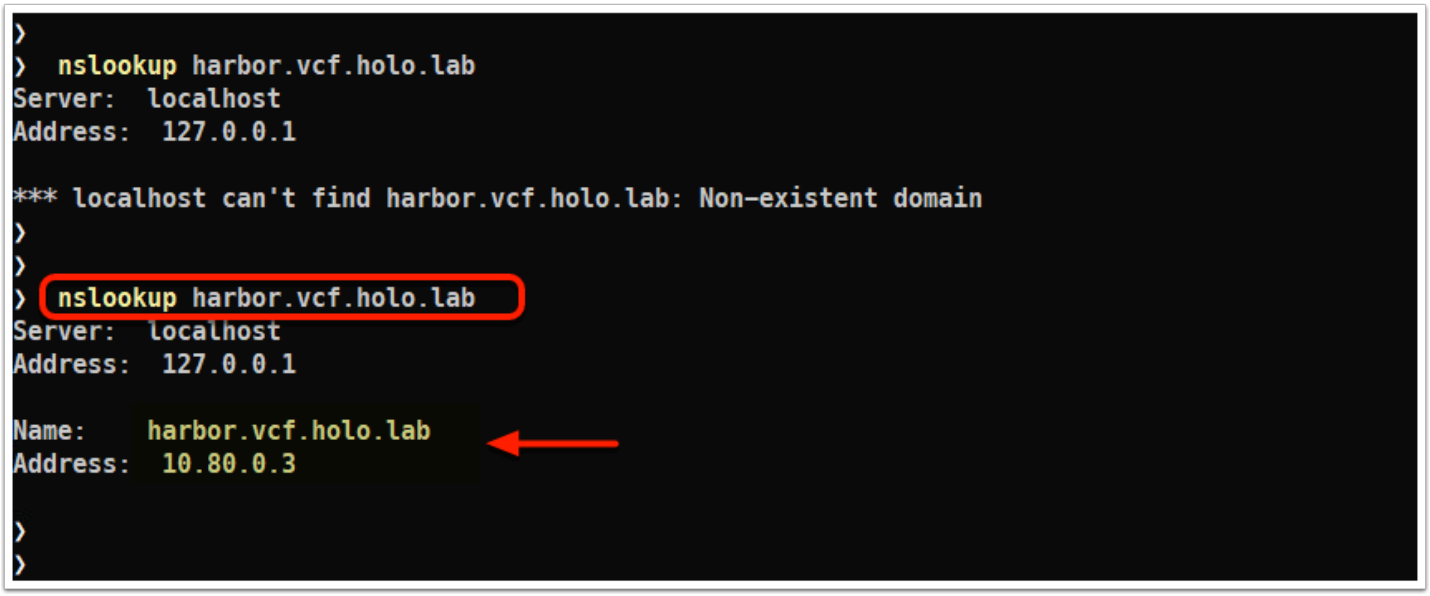
Install Cloud Native Registry Service (Harbor)



Select **Stateful Sets** and you will see all the **harbor** needed stateful sets in place: for harbor-database, harbor-redis and harbor-trivy

Since the harbor service got installed, let's check if external-dns had provided a dynamic update for it in DNS

Install Cloud Native Registry Service (Harbor)



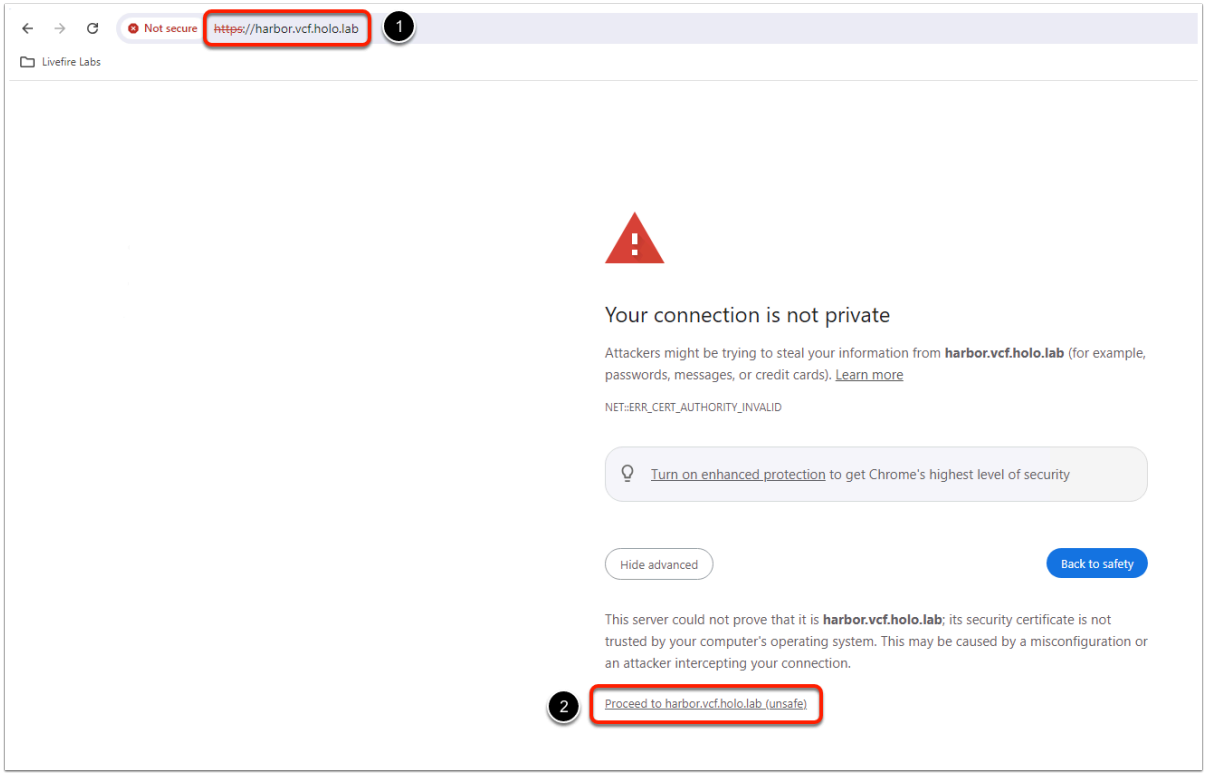
Go back to your Windows command prompt box and check again the resolution

```
nslookup harbor.vcf.holo.lab
```

Looks good!

Now, let's get to the harbor UI

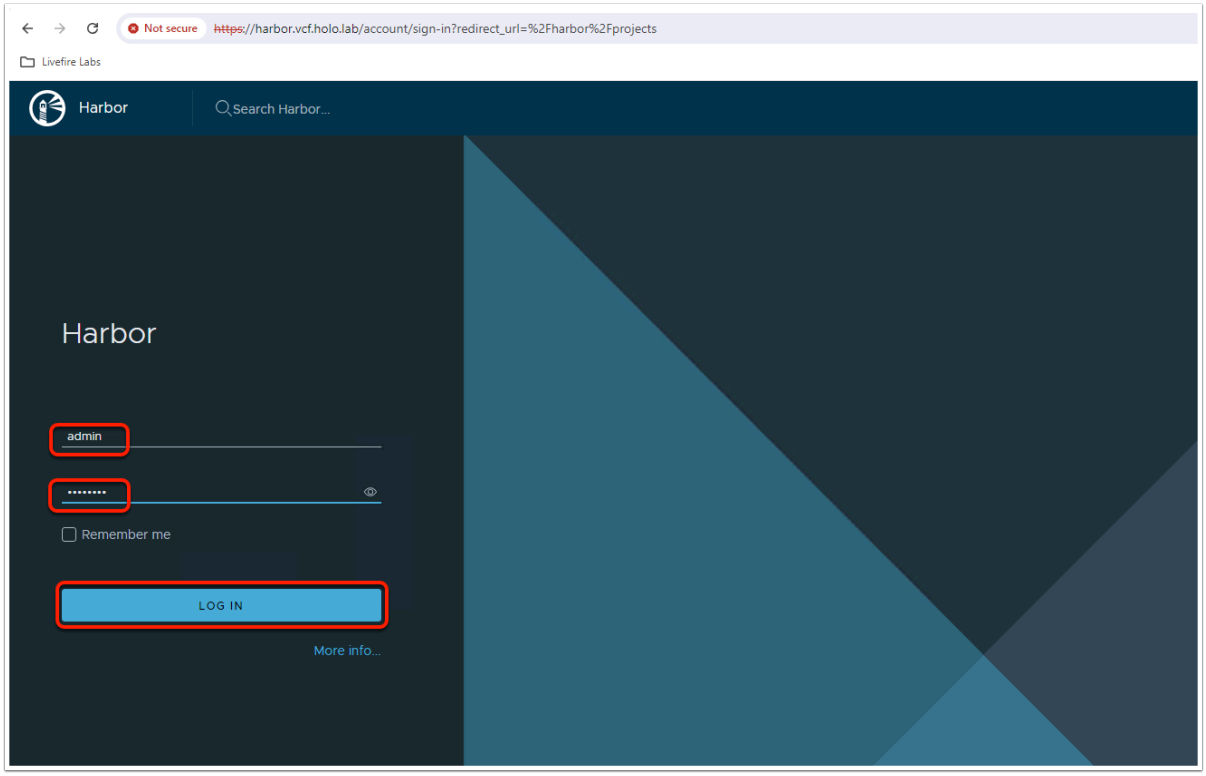
Install Cloud Native Registry Service (Harbor)



Open a **new** tab in **Chrome** and:

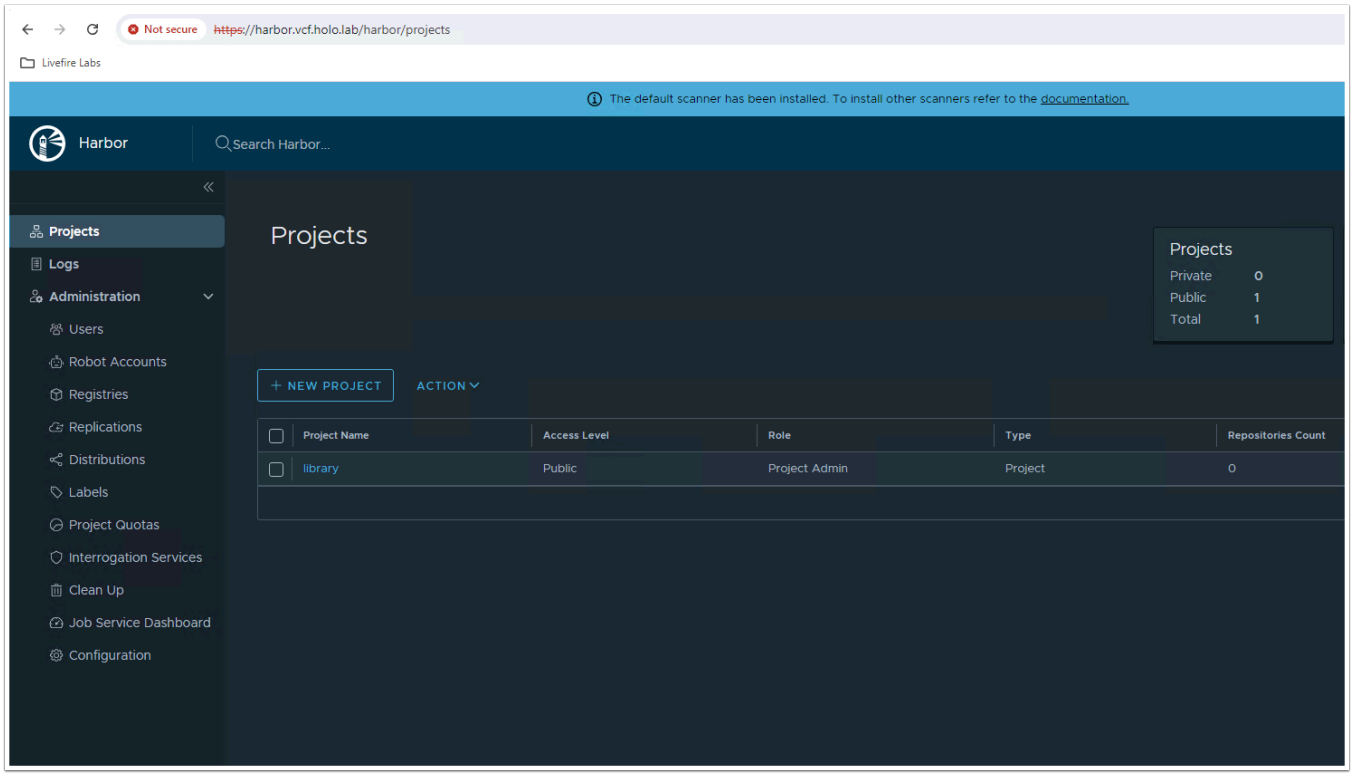
1. Go to https://harbor.vcf.holo.lab
2. Click on **Proceed**

Install Cloud Native Registry Service (Harbor)



In Harbor Log In page, Populate the credentials and click on **LOG IN**

```
user: admin
password: VMware1!
```

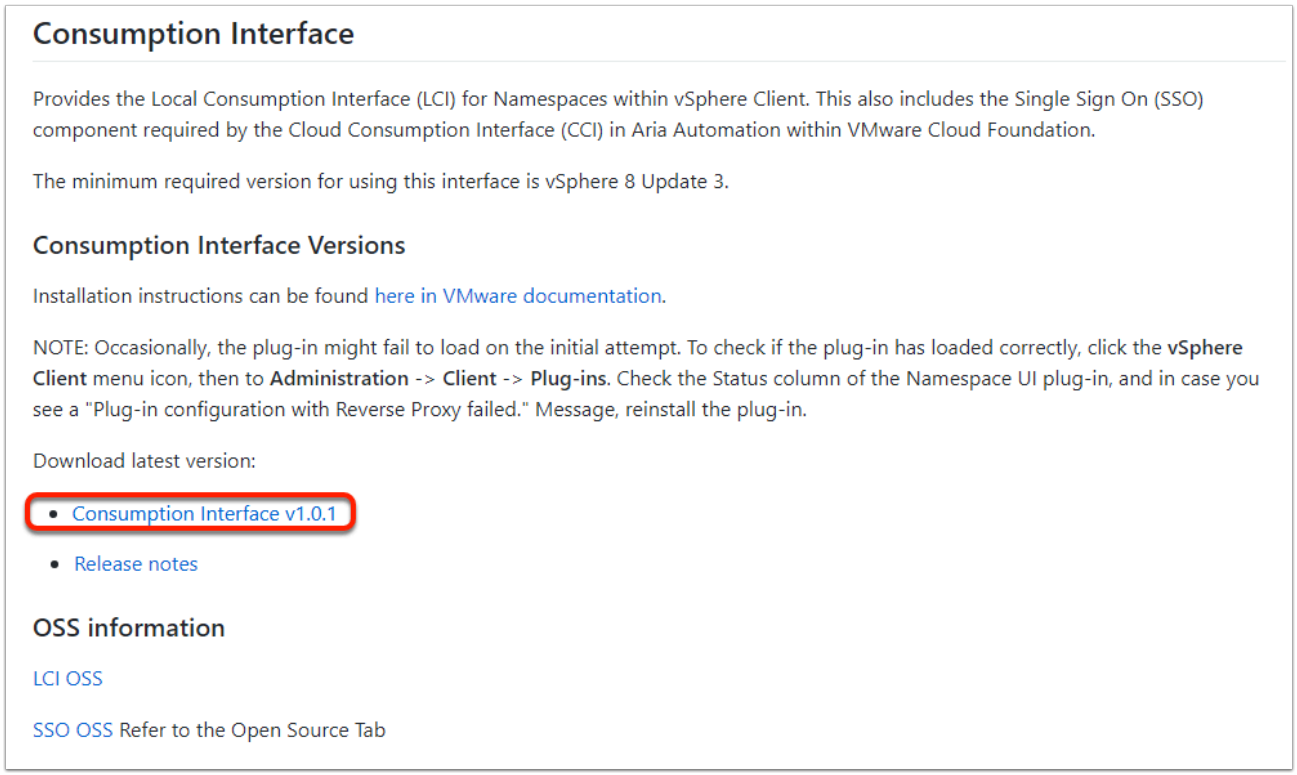


The Harbor is ready!

Install (Local) Consumption Interface

Provides the Local Consumption Interface (LCI) for Namespaces within vSphere Client. This also includes the Single Sign On (SSO) component required by the Cloud Consumption Interface (CCI) in Aria Automation within VMware Cloud Foundation.

The minimum required version for using this interface is vSphere 8 Update 3.

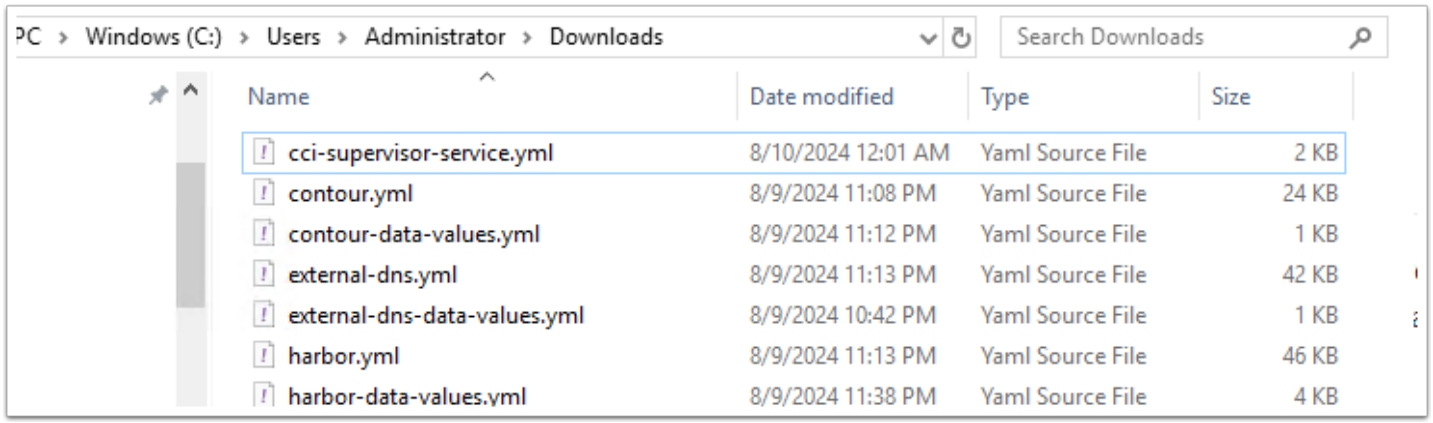


Download the latest available manifests for Consumption Interface to your **Downloads** folder

cci-supervisor-service.yml

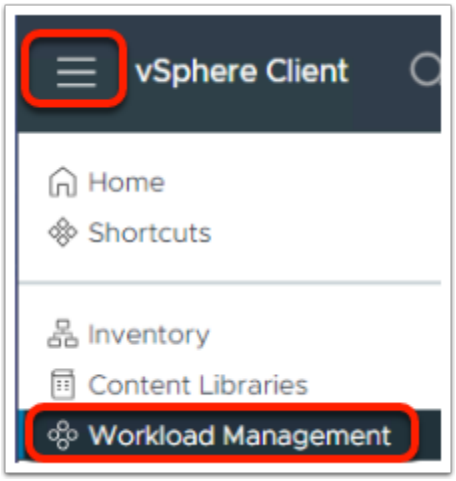
https://vsphere-tmm.github.io/Supervisor-Services/#consumption-interface

Install (Local) Consumption Interface



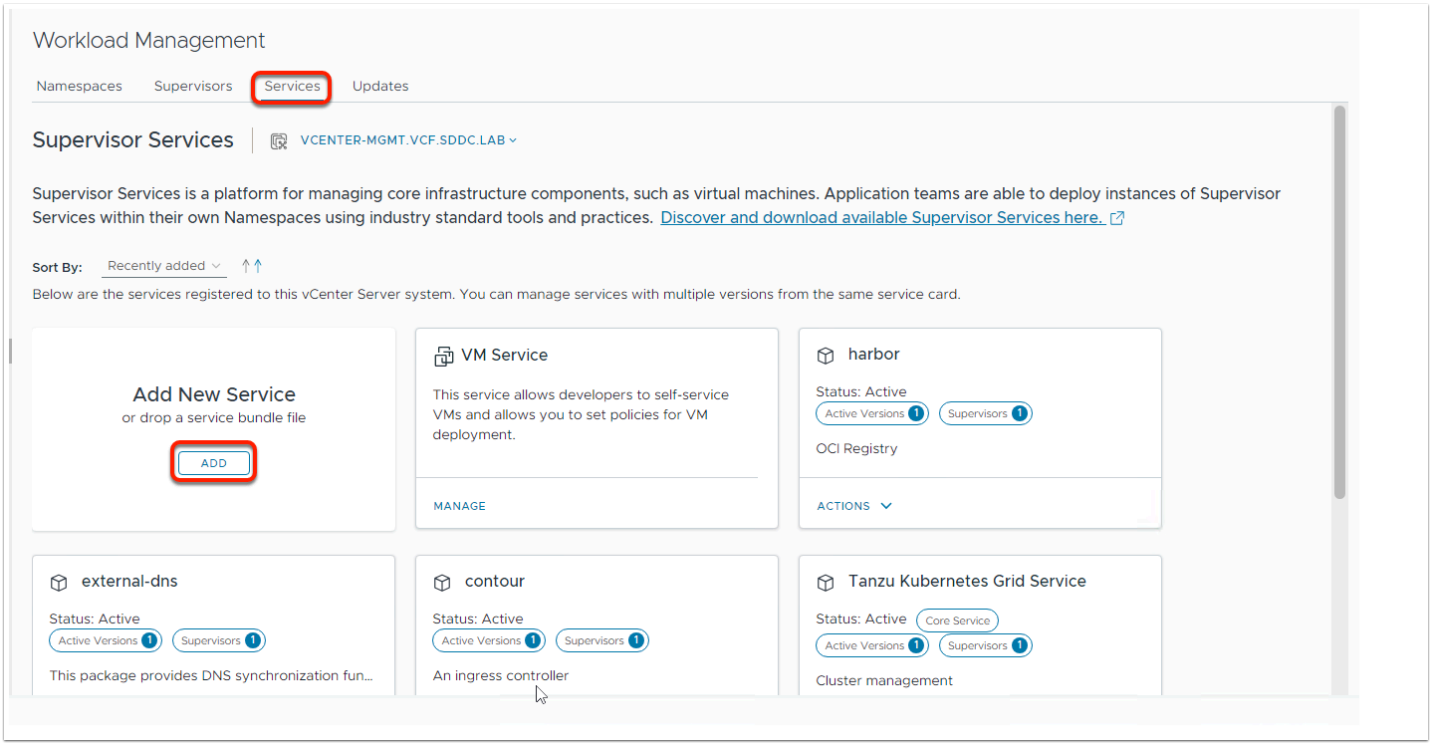
You are ready to deploy Local Consumption Interface as a Supervisor service!

Install (Local) Consumption Interface



From the vSphere Client home menu, select **Workload Management**.

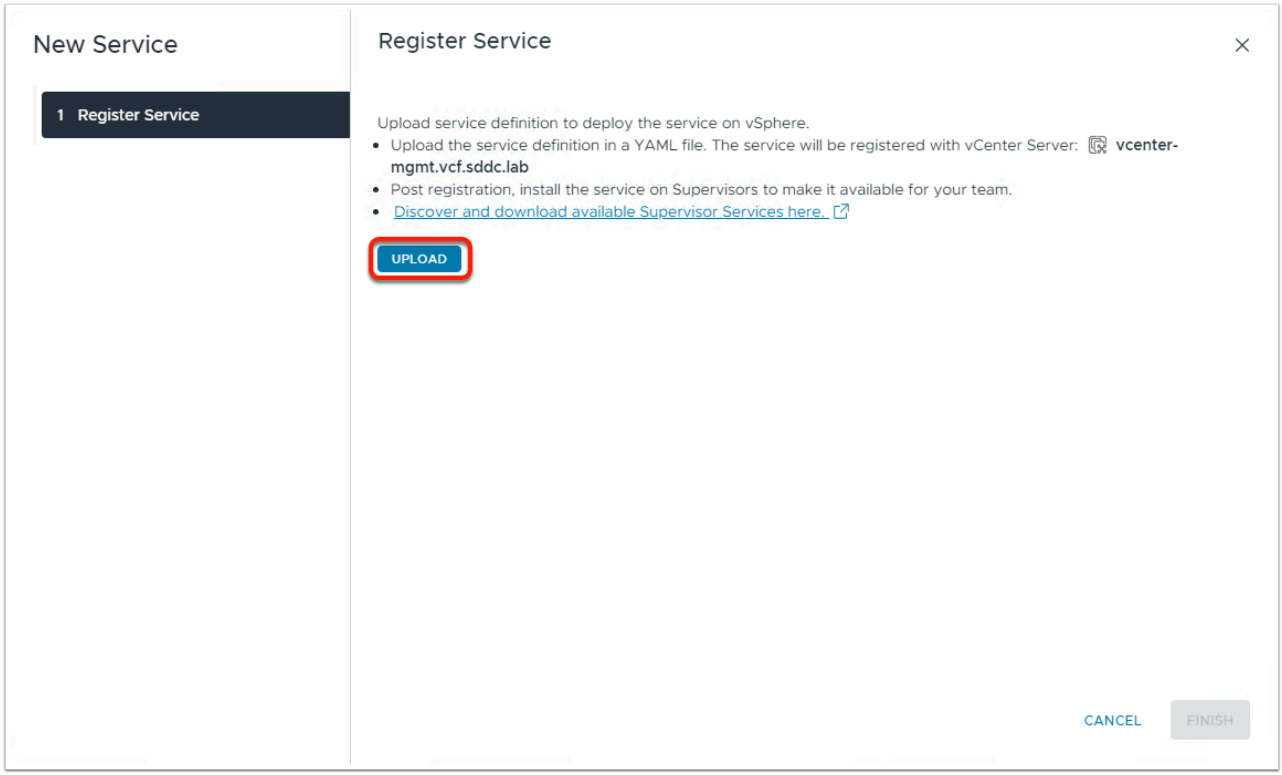
Install (Local) Consumption Interface



In the **Workload Management** section, select **Services**.

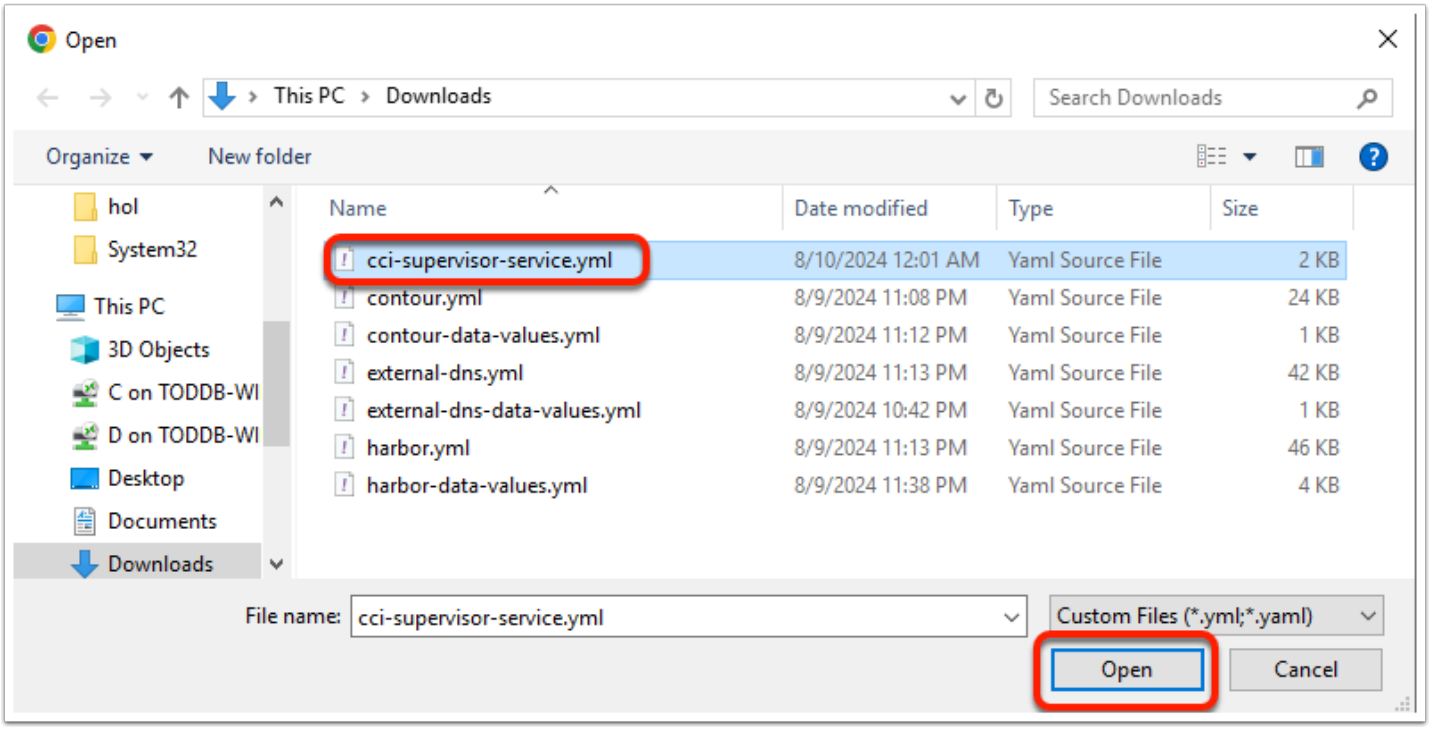
Locate the **Add New Service** card and click on **ADD**

Install (Local) Consumption Interface



In the **New Service** wizard, for **Register Service** select **UPLOAD**

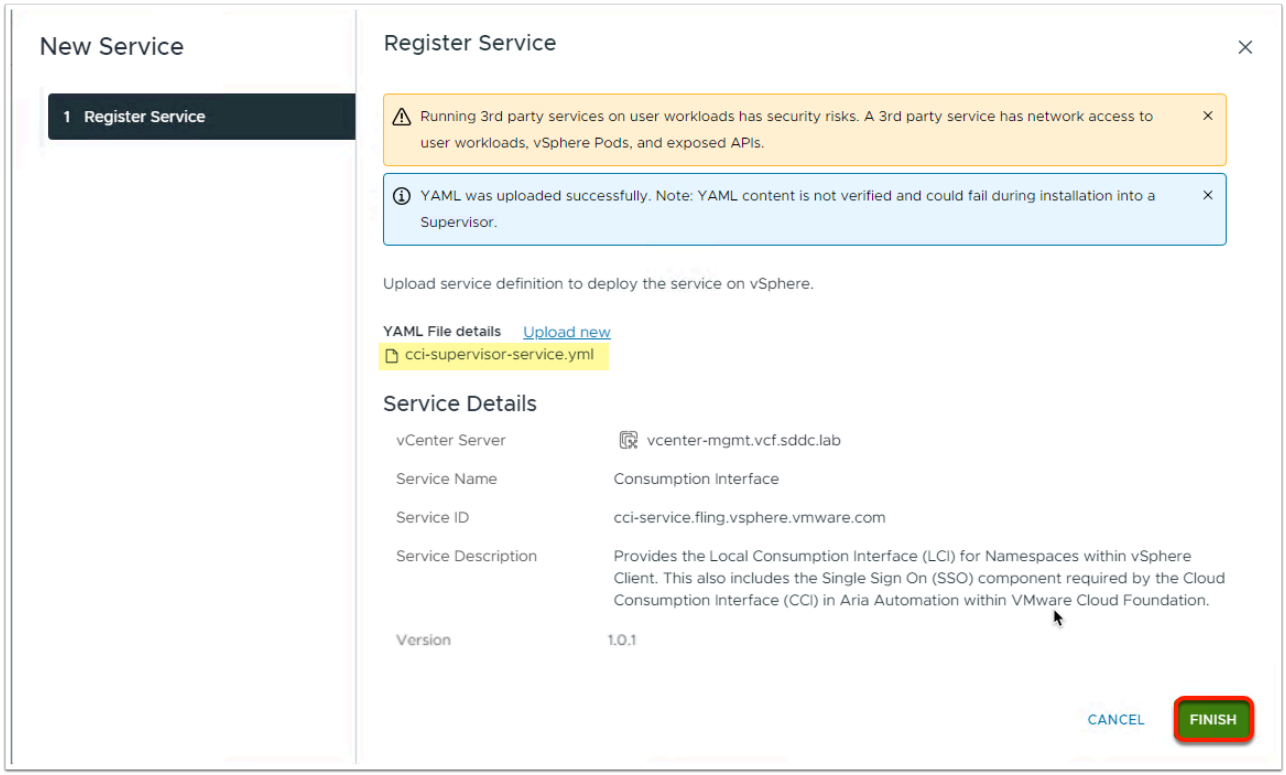
Install (Local) Consumption Interface



In the Chrome **Open** menu go to **Downloads** folder and select the **cci-supervisor-service.yml** you downloaded recently.

Confirm with clicking on **Open** button

Install (Local) Consumption Interface

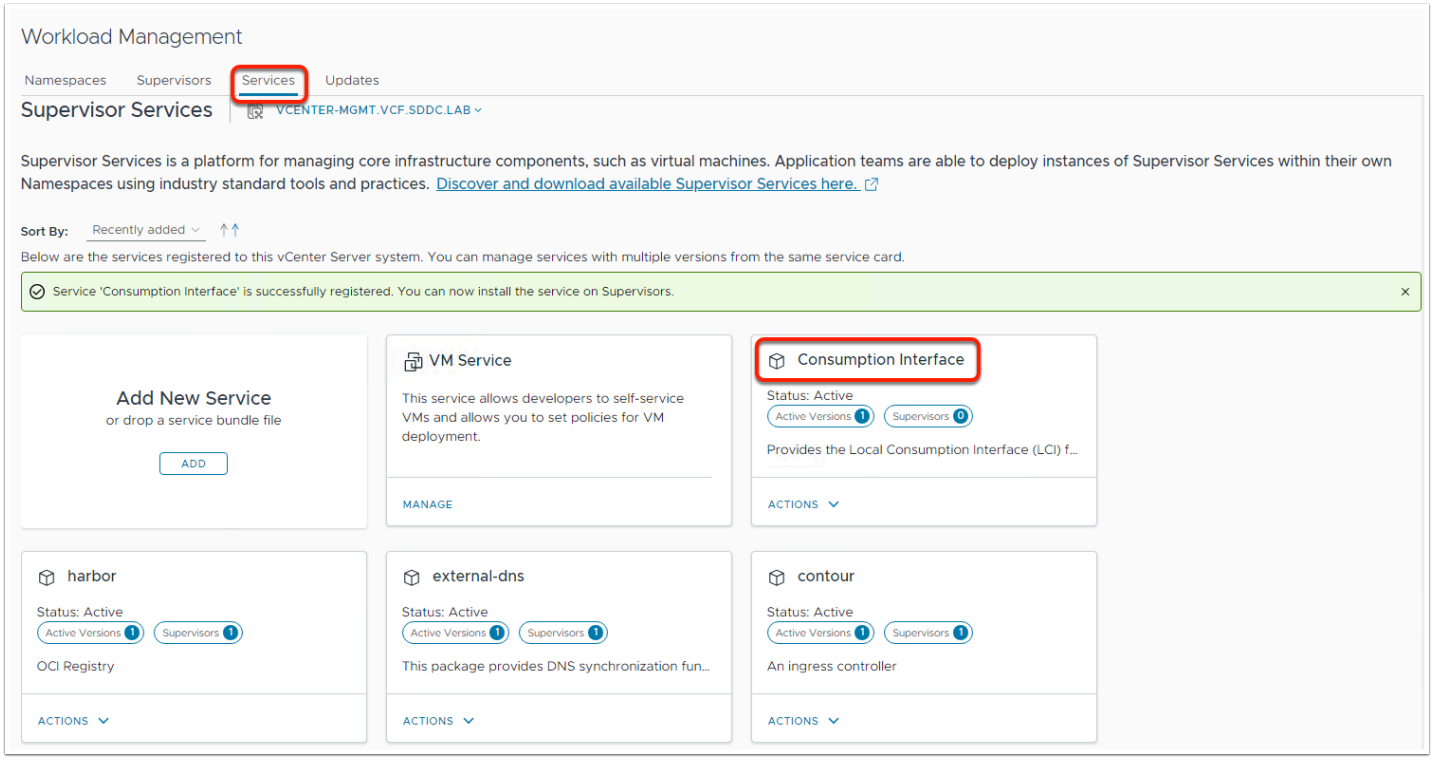


In the **New Service** wizard, for **Register Service** make sure you can see the aforementioned **cci-supervisor-service.yml** file.

Its displayed version should match the one you downloaded.

Confirm with **FINISH**

Install (Local) Consumption Interface



In a while you will see a displayed message for the **Consumption Interface** service successfully registered with your **Supervisor**.

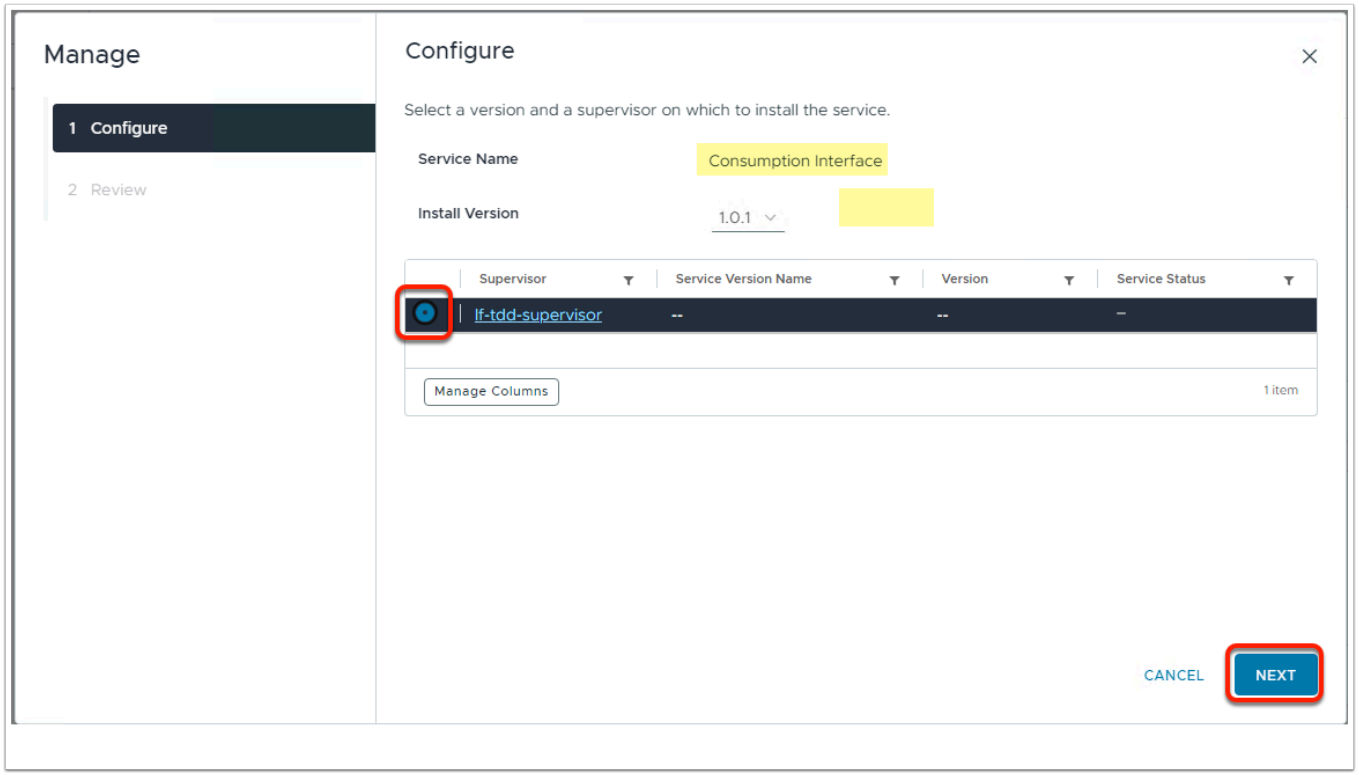
You need to activate it with particular configuration settings.

You can close the message in green field.

Install (Local) Consumption Interface

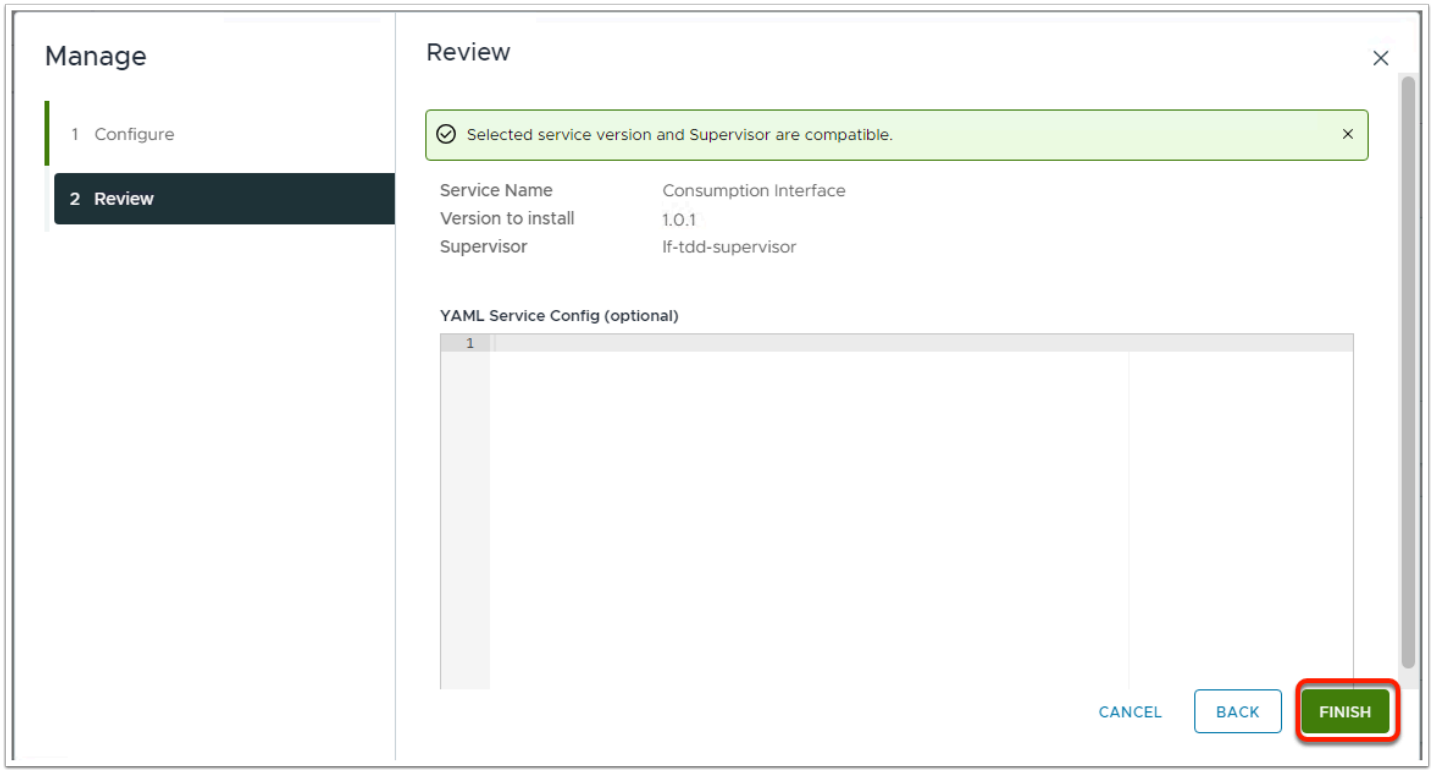
1. Locate your recently registered **Consumption Interface** service
2. From **Actions** drop down menu
3. select **Manage Service**

Install (Local) Consumption Interface



In the **Manage** wizard, from **Configure** select your **Supervisor** and confirm **NEXT**

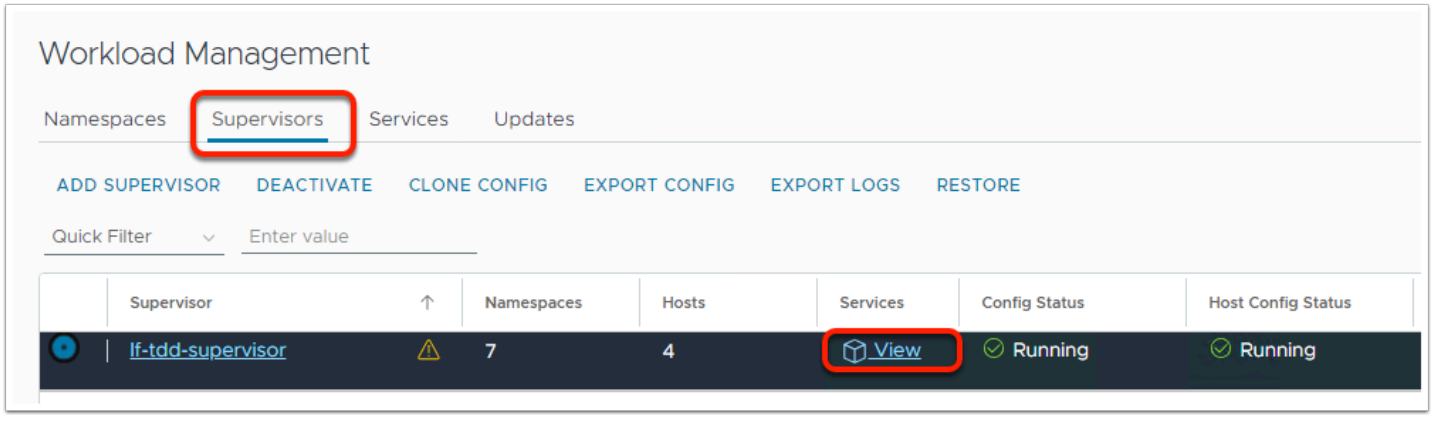
Install (Local) Consumption Interface



No data file to use

Click **FINISH**

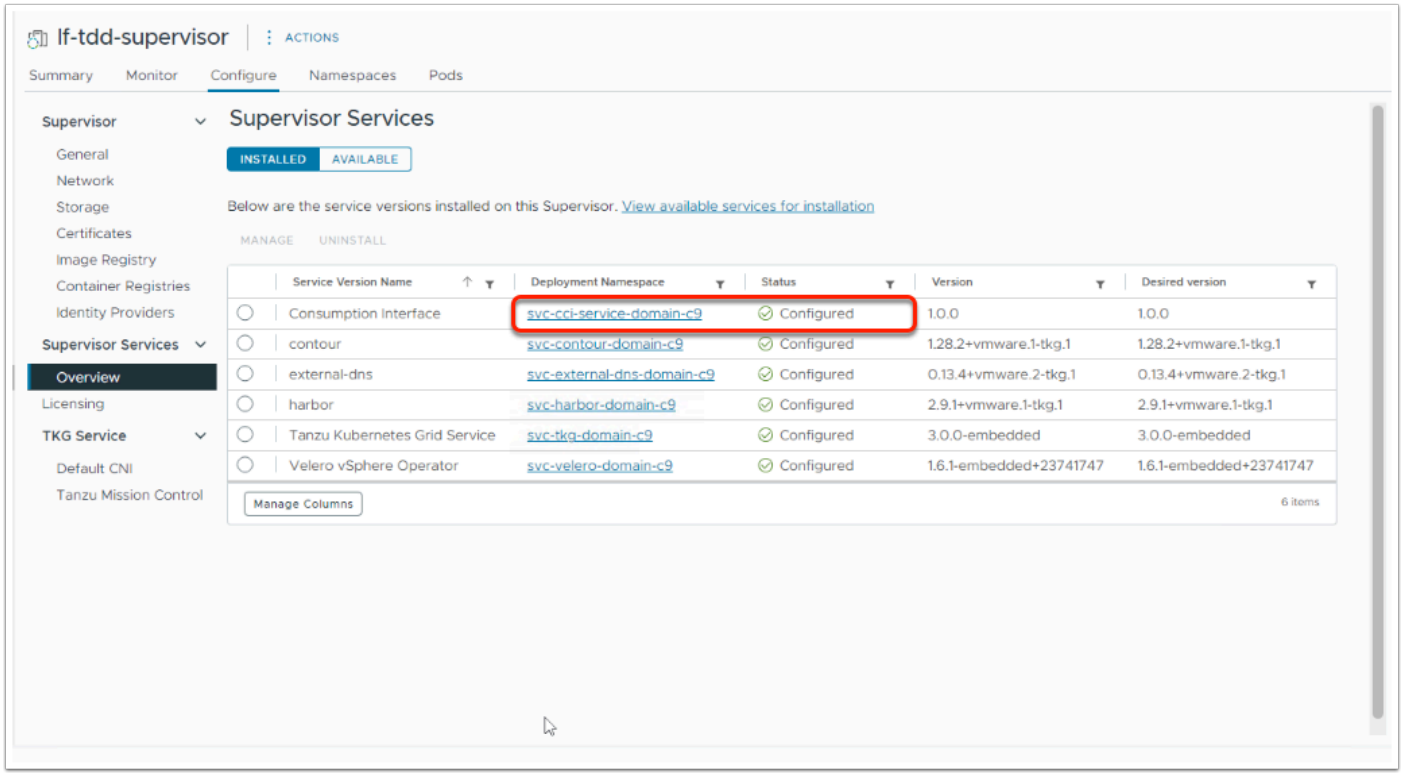
Install (Local) Consumption Interface



Go back in the **Workload Management** section

Select **Supervisors** and click on **View** under **Services**

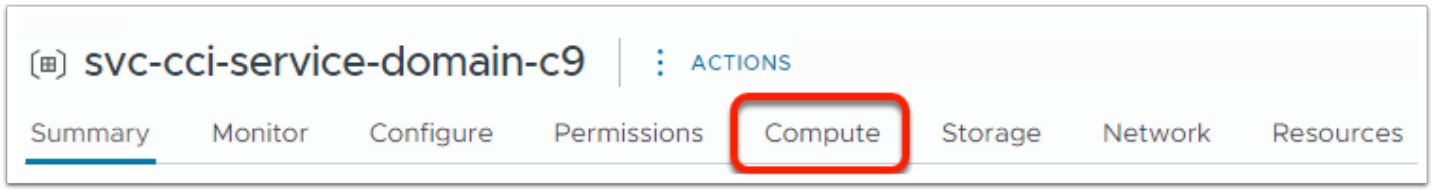
Install (Local) Consumption Interface



In a couple of minutes the **Consumption Interface** service Status will change from Configuring to **Configured**

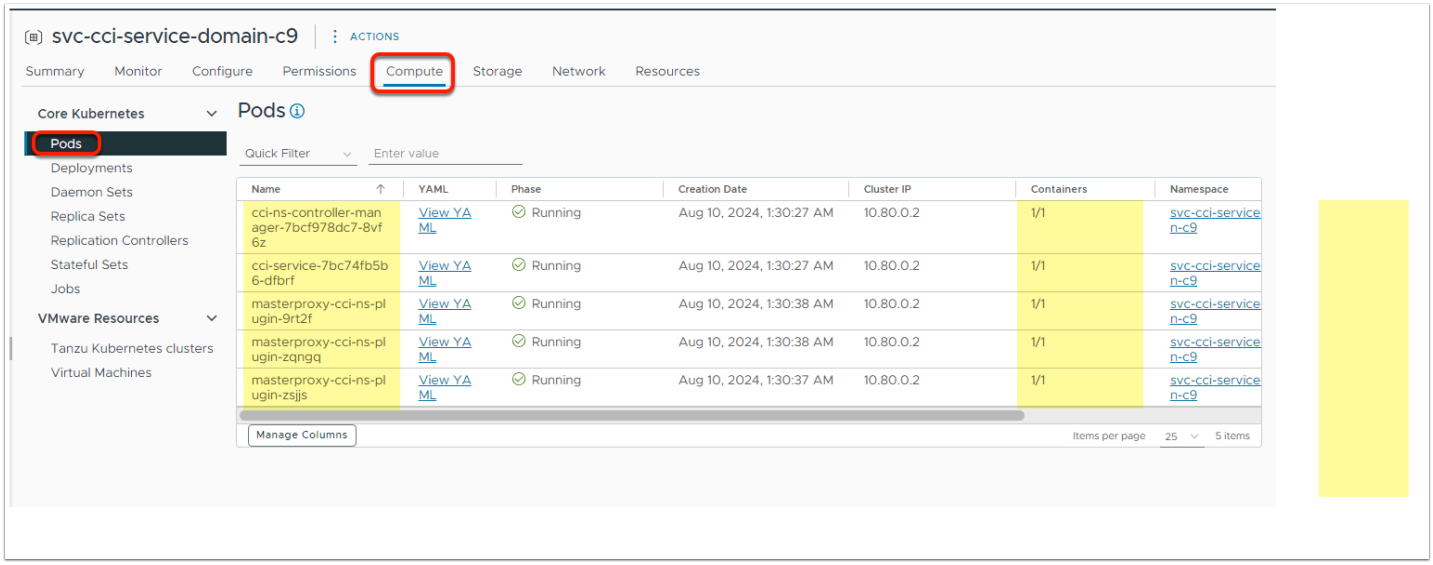
Click on the **svc-cci-service-domain-c9** under **Deployment Namespace**

Install (Local) Consumption Interface



For the **svc-cci-service-domain-c9** Namespace, click on **Compute**

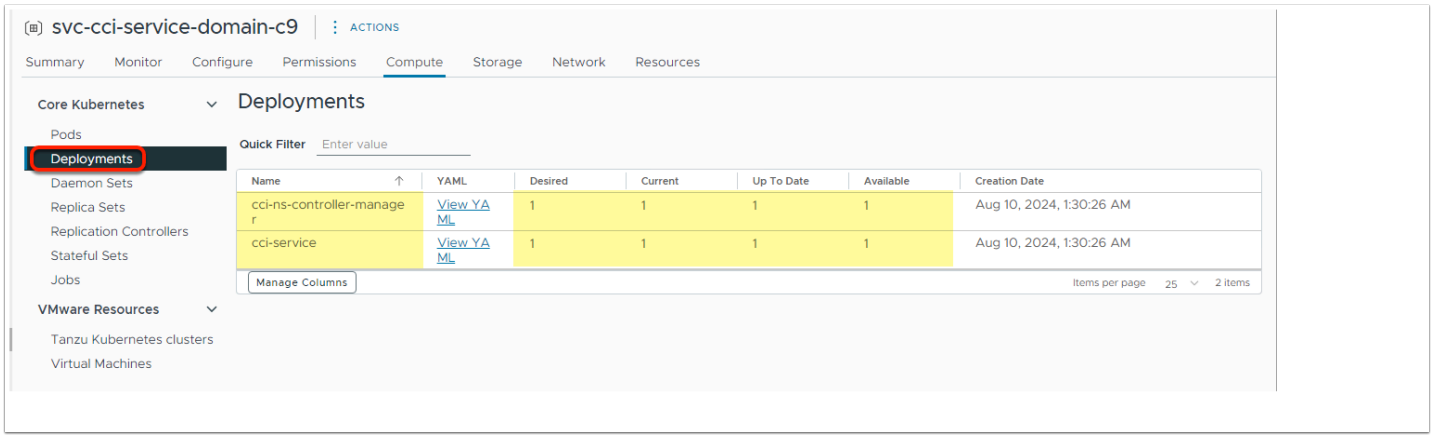
Install (Local) Consumption Interface



Under **Compute**, you can review all the **Kubernetes Core** configurations for **Consumption Interface**.

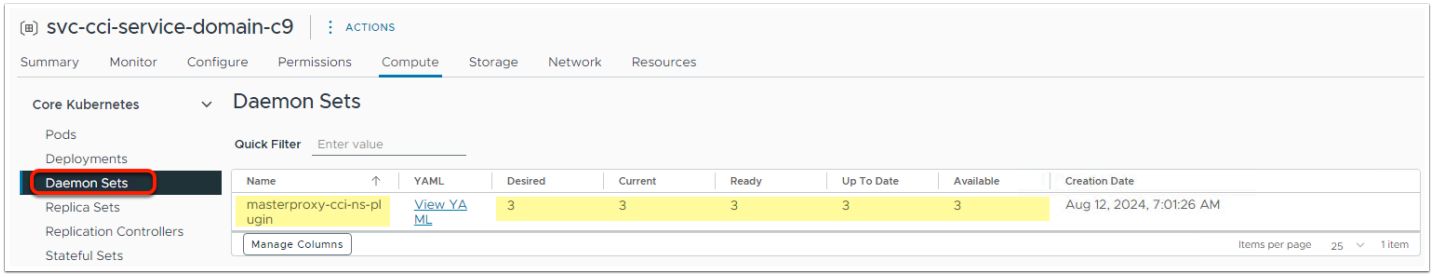
Select **Pods** and you will see all the **Consumption Interface** needed pods, running as **vSphere Pods**

Install (Local) Consumption Interface



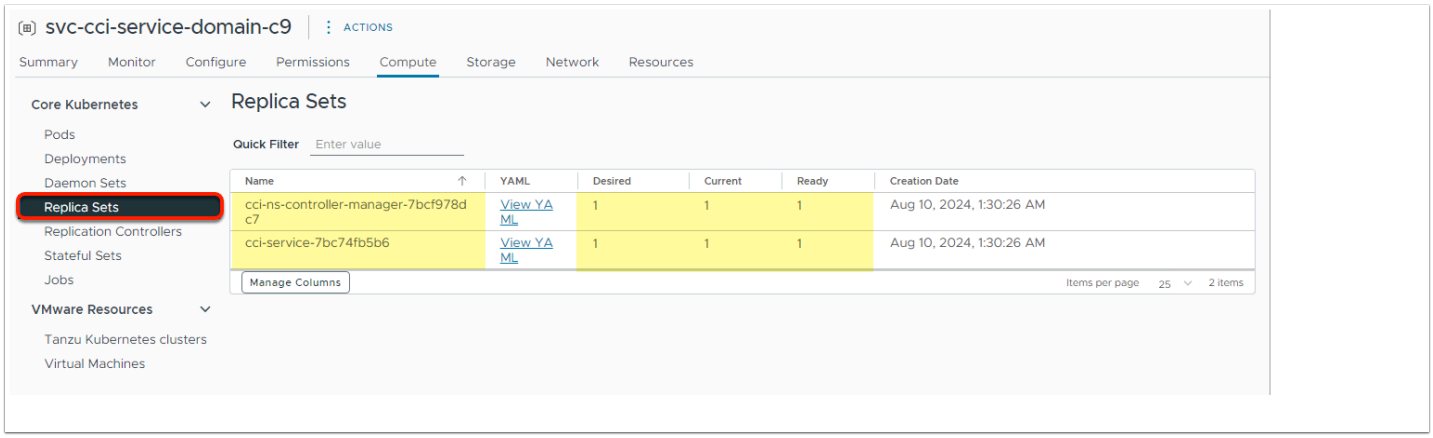
Select **Deployments** and you will see all the **Consumption Interface** needed deployments in place

Install (Local) Consumption Interface



Select **Daemon Sets** and you will see all the **Consumption Interface** needed replica sets in place

Install (Local) Consumption Interface



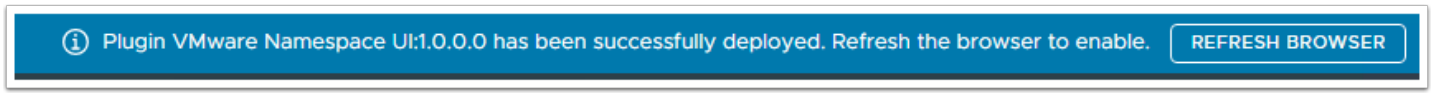
Select **Replica Sets** and you will see all the **Consumption Interface** needed replica sets in place

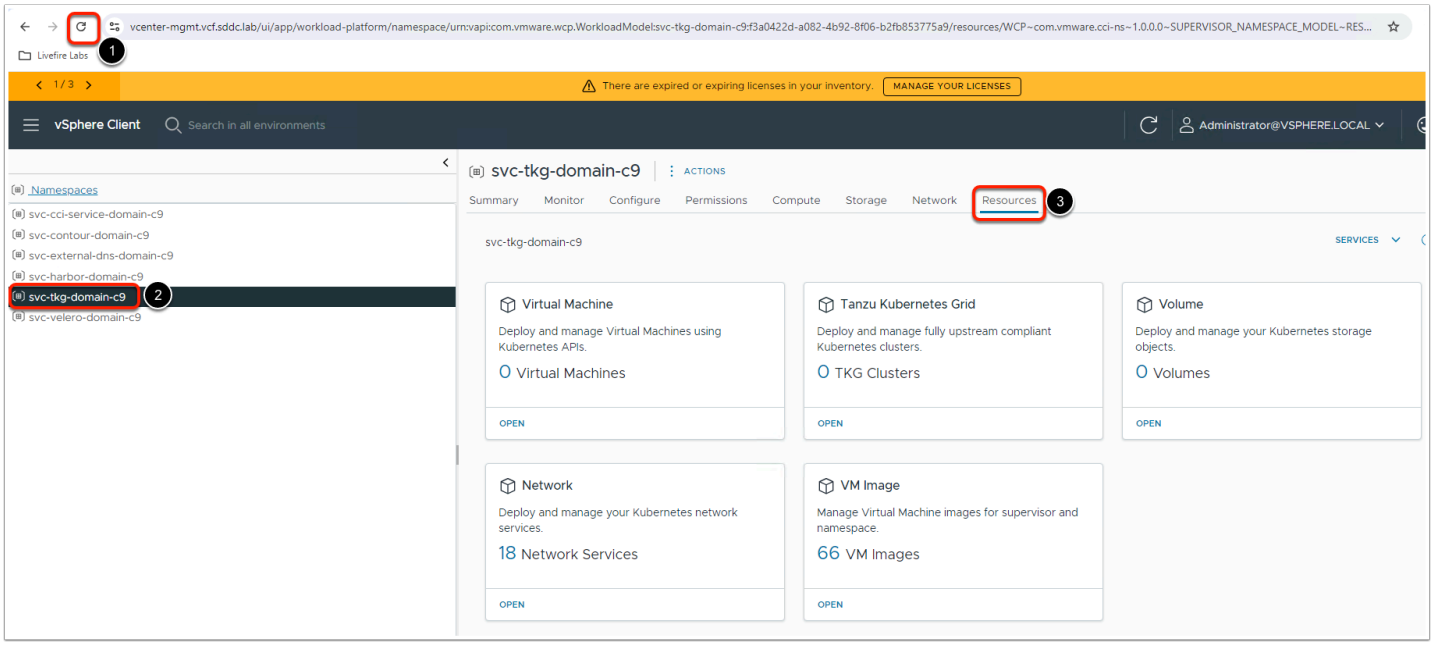
Install (Local) Consumption Interface

IMPORTANT for LCI : **Local Consumption Interface**

- LCI is not supported on Multi-Zone Supervisors.
- After installing the service into a supervisor, you should expect a delay while the plugin is installed. If you do not see the plugin listed in the Resources tab under Workload Management -> Namespaces -> your namespace, please wait and then refresh the page.

- After installing the service into a supervisor, **you will need to reload the web page or login in a new window to allow the plugin to load properly.**





1. **Reload/Refresh** the vSphere client in your browser
2. Under **Namespaces** section go to **svc-tkg-domain-c9**
3. Notice the new plugin installed under the **Resources** tab

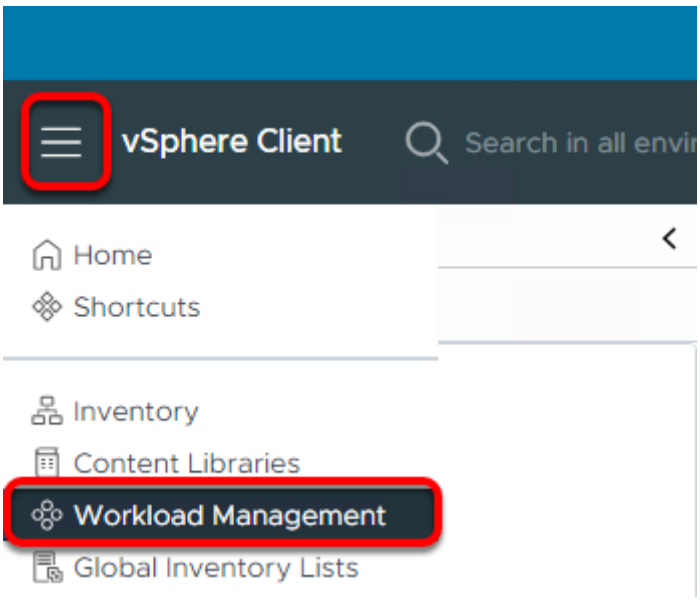
You are now ready to proceed to the next section!

Create and Configure a Namespace

A vSphere Namespace sets the resource boundaries where vSphere Pods, VMs, and TKG clusters can run. As a vSphere administrator, you create and configure vSphere Namespaces through the vSphere Client.

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

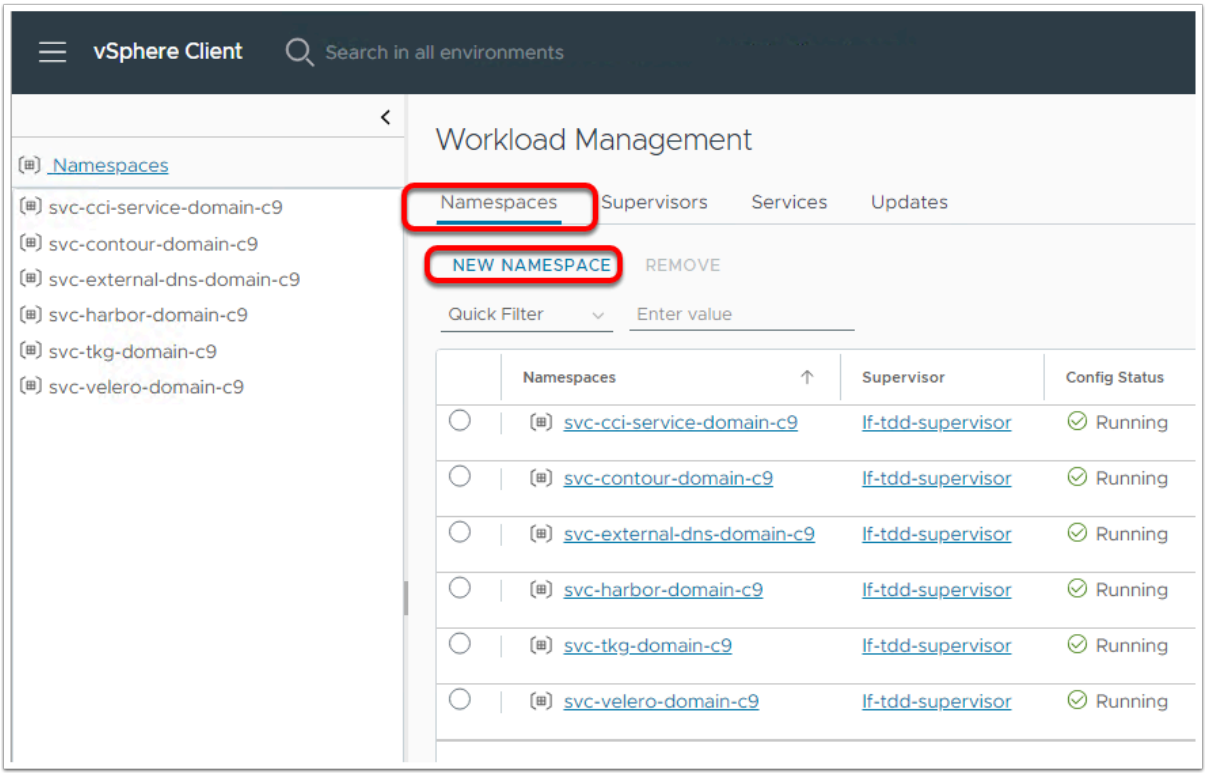
Use the vSphere client to Create a Namespace



Click the Hamburger Menu in vCenter

Click Workload Management

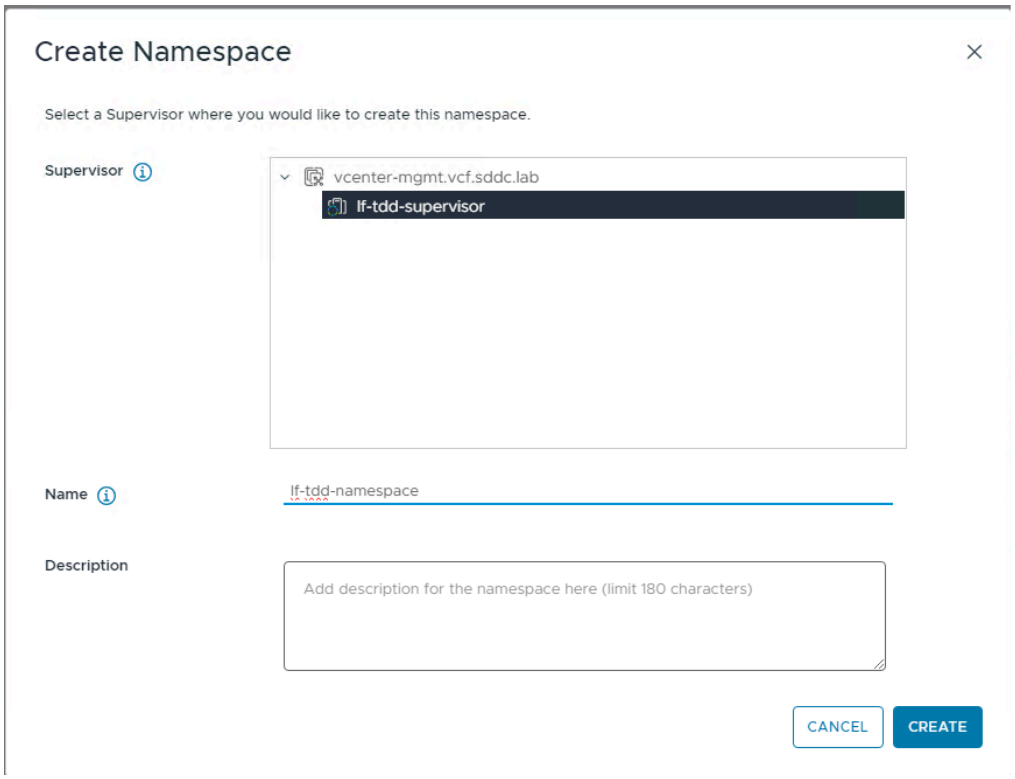
Use the vSphere client to Create a Namespace



Click the **Namespaces** tab

Click **NEW NAMESPACE**

Use the vSphere client to Create a Namespace



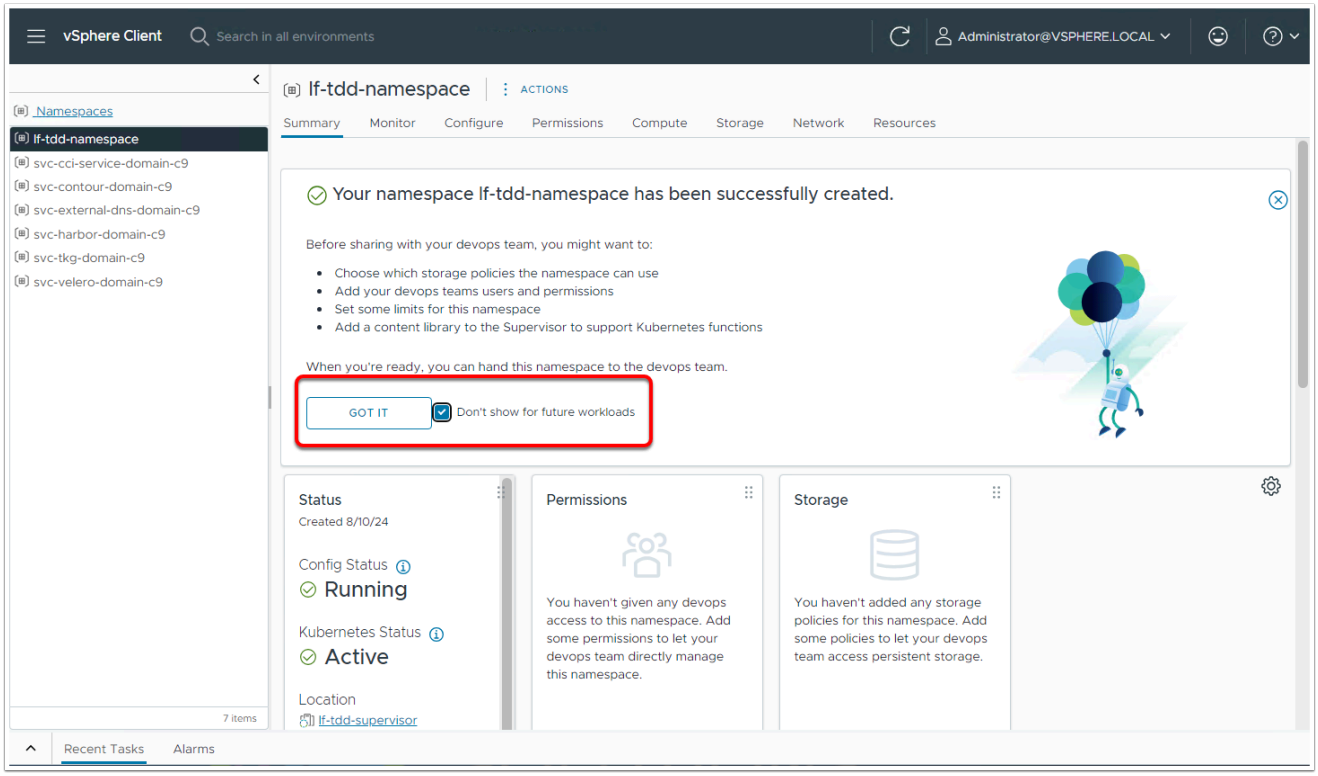
Choose your current Supervisor Cluster

Give your new **NAMESPACE** a name, i.e.

```
lf-tdd-namespace
```

Click **CREATE**

Configure New Namespace

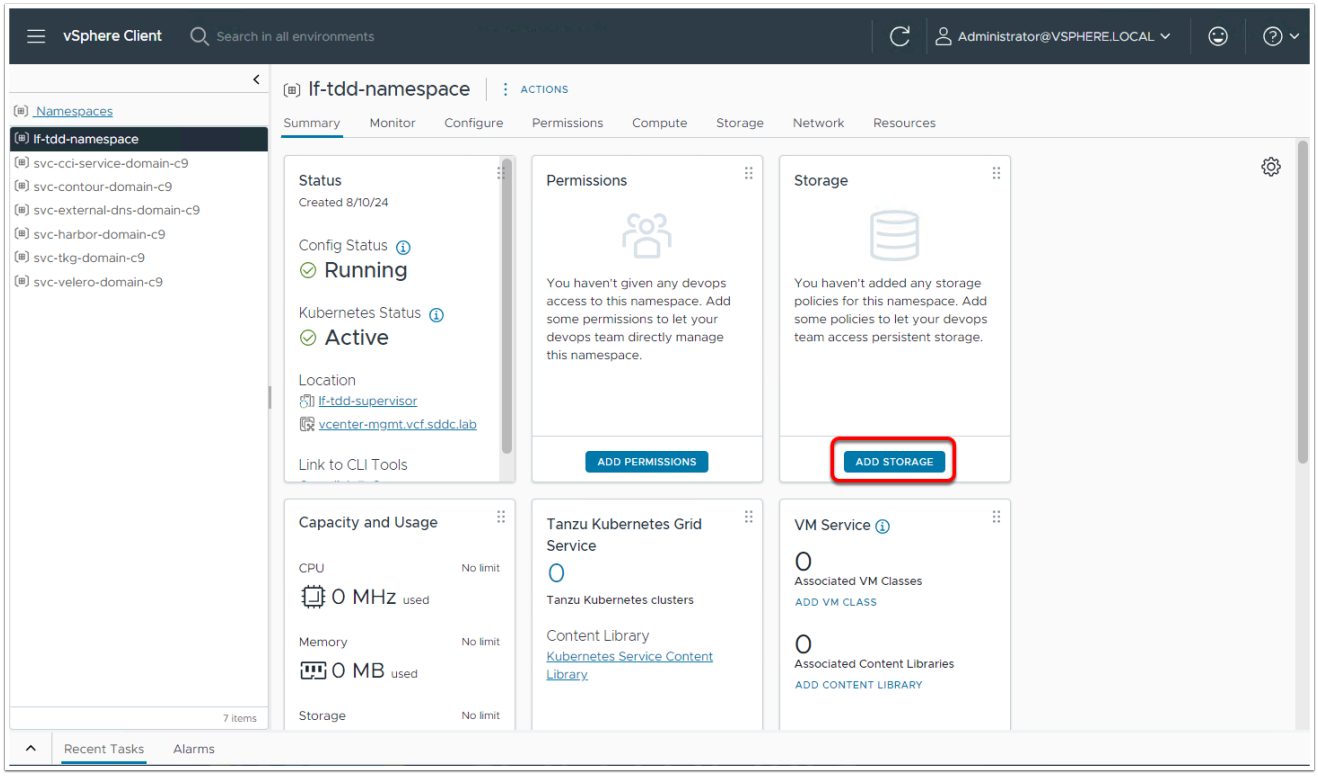


Click the **Don't Show for future workloads**

Click **GOT IT**

This will save you some screen real estate

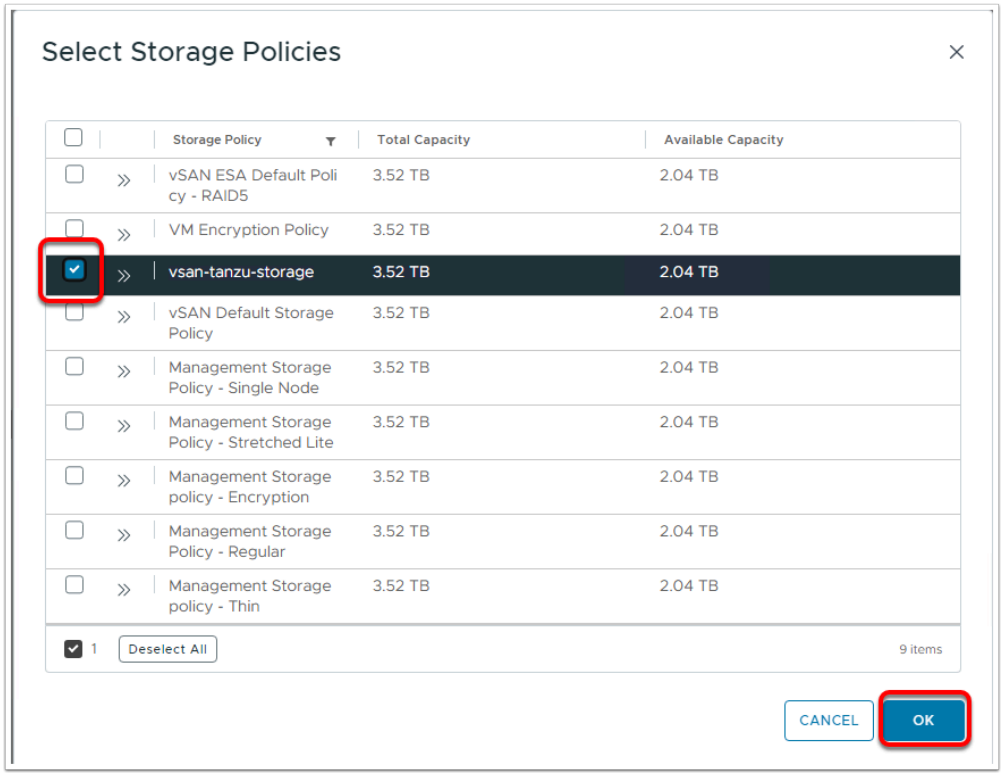
Configure New Namespace



Add the tanzu storage policy to this namespace.

Click **ADD STORAGE**

Configure New Namespace



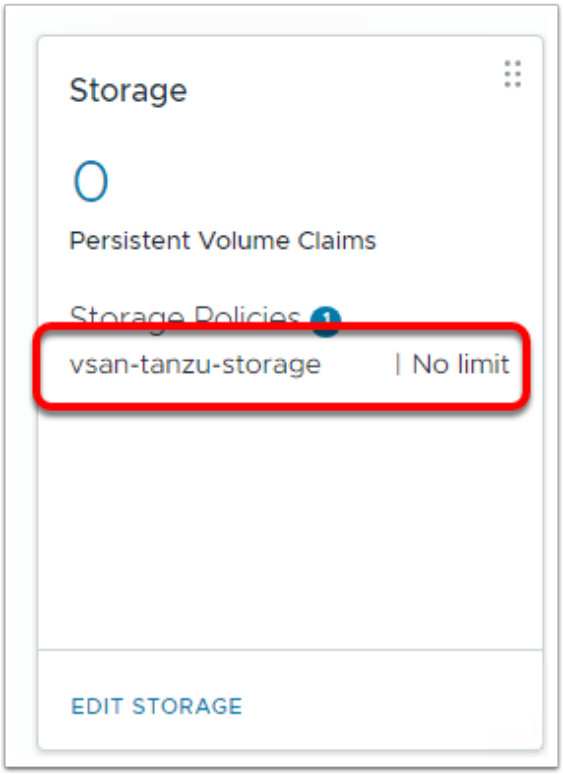
To deploy VMs, vSphere PODs, TKG Clusters, etc to this namespace some settings need to be set..

Start with Storage

Click the selection box for **vsan-tanzu-storage** storage policy

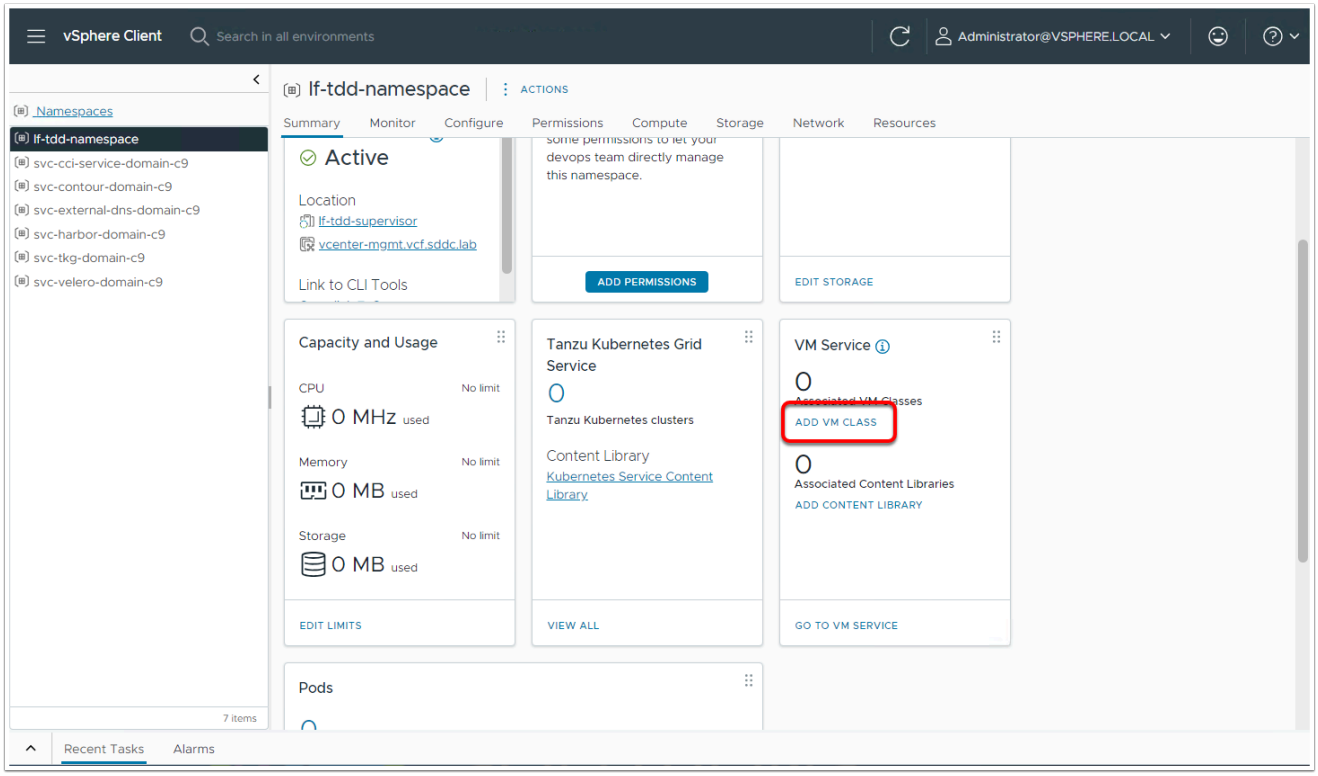
Click **OK**

Configure New Namespace



Verify the tanzu storage policy is now attached to the namespace

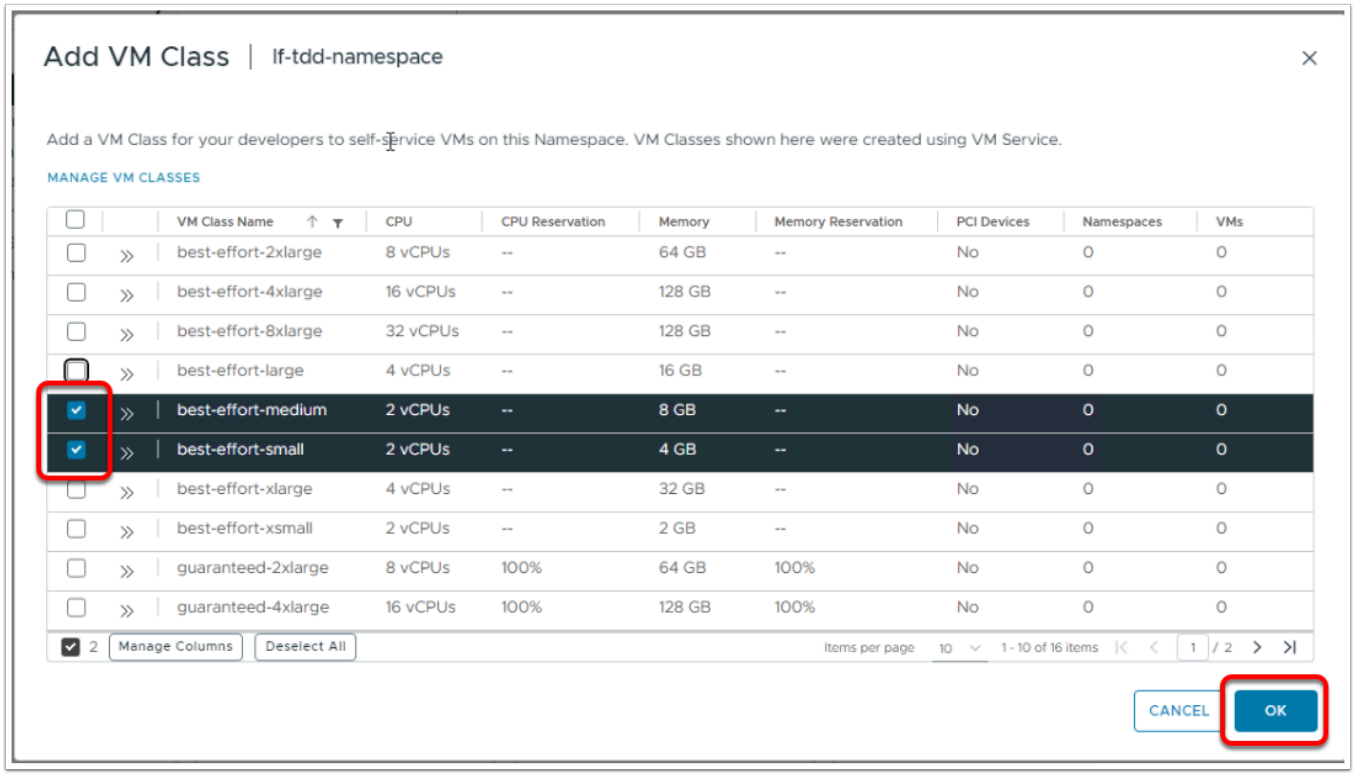
Configure New Namespace



Next you have to give the namespace permission to deploy certain VM Classes.

Click **ADD VM CLASS**

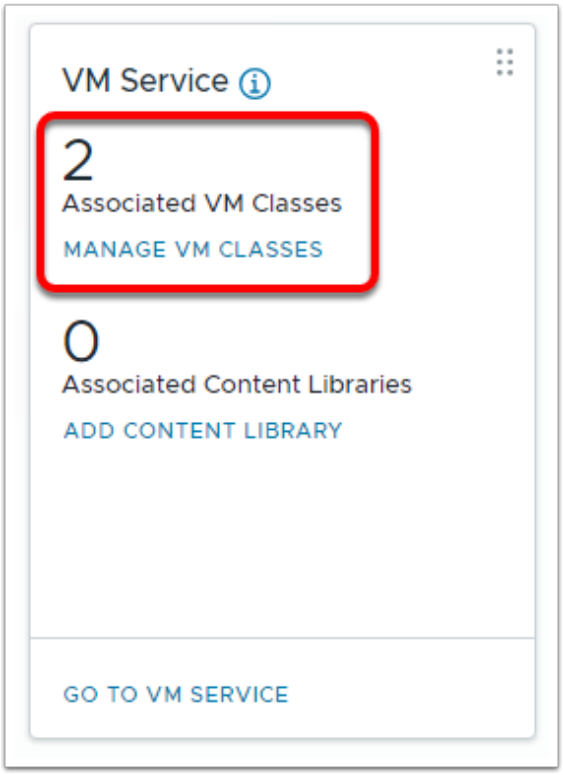
Configure New Namespace



For a lab environment like this one we will use **best-effort small** and **medium** VM Classes, and **best-effort** since this is a lab and wil not be deploying any large enterprise applications. **Best Effort** since we do not need to worry about resource contention.

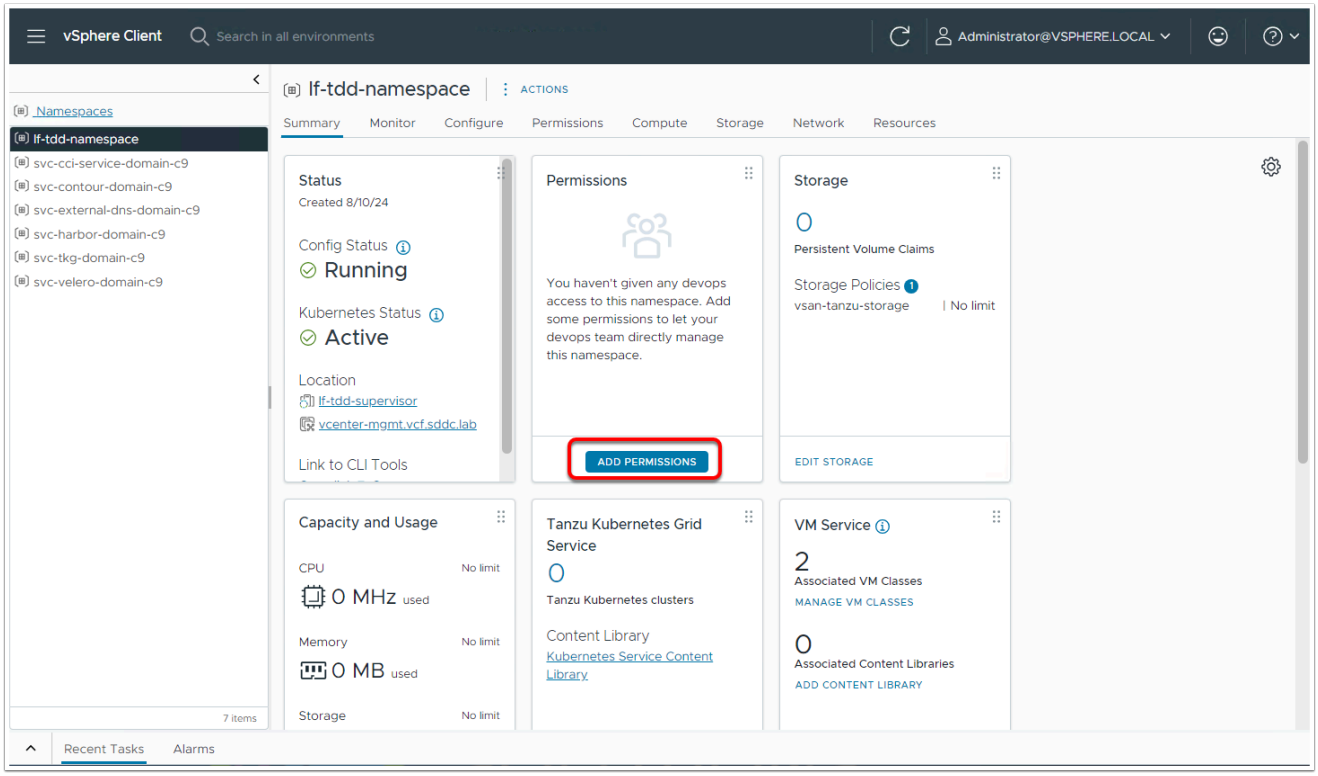
Click **OK**

Configure New Namespace



Verify that you now have two **VM CLASSES** assigned for use to the namespace

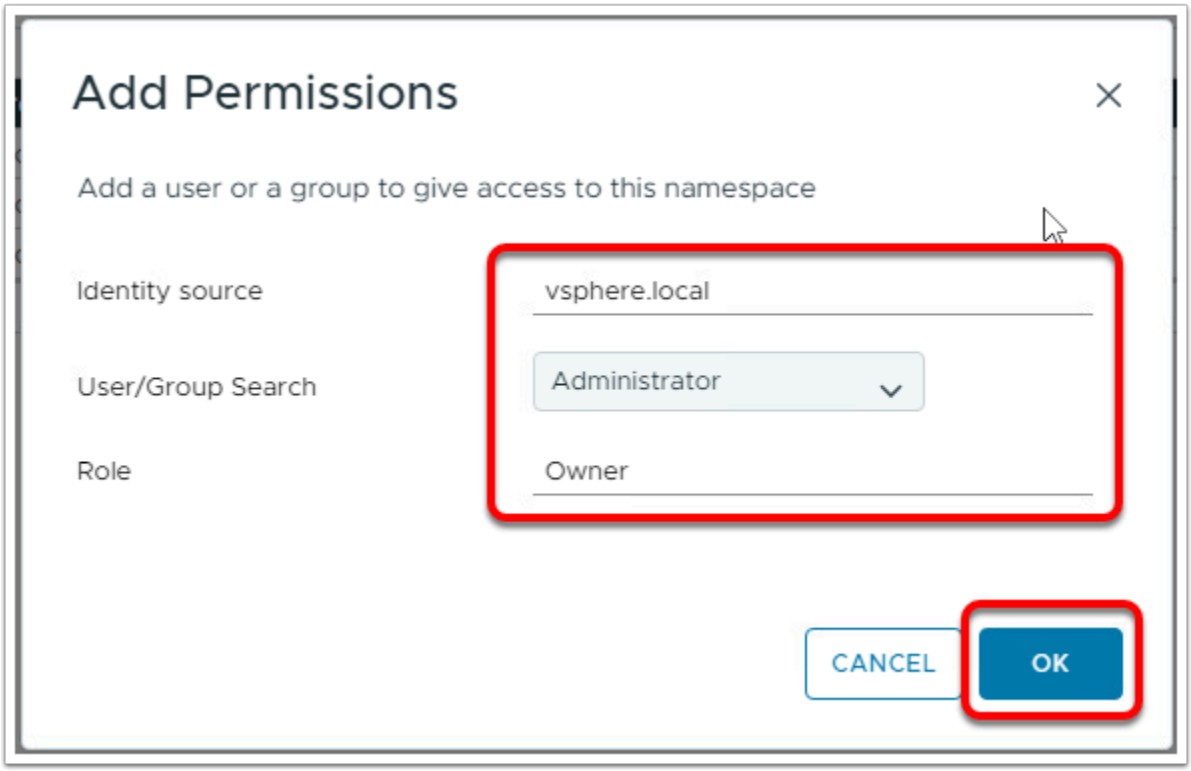
Configure New Namespace



Now you need to give team members permission to use the namespace and the attached services.

Click **ADD PERSMISIONS**

Configure New Namespace

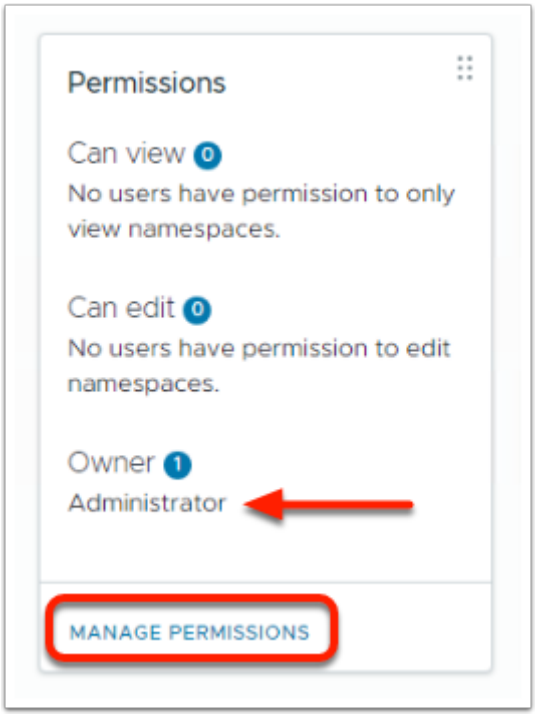


You are going to add groups from **SSO** to the three different Roles for this namespace

- Identity Source: vsphere.local
- User/Group Search: Administrator
- Role: Owner

Click **OK**

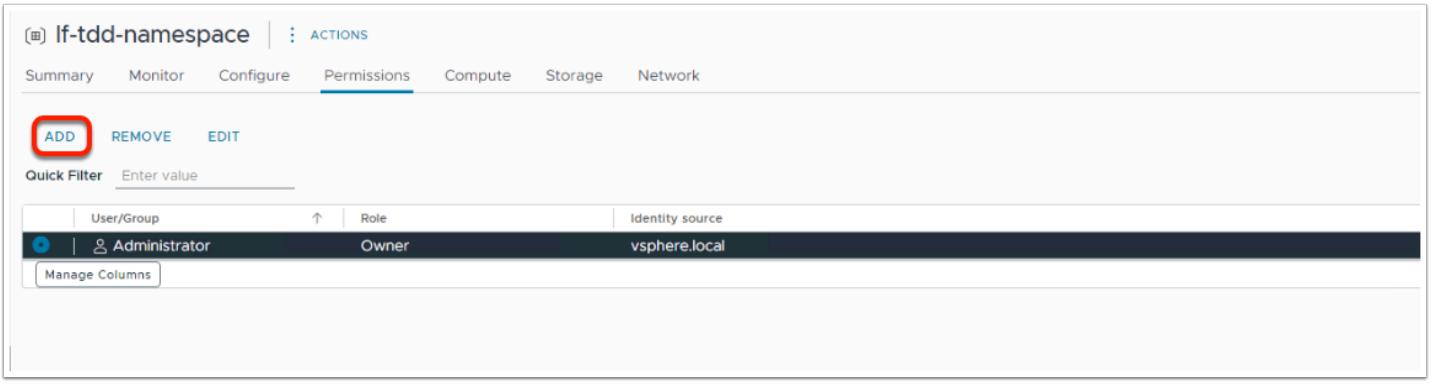
Configure New Namespace



Two more groups/roles

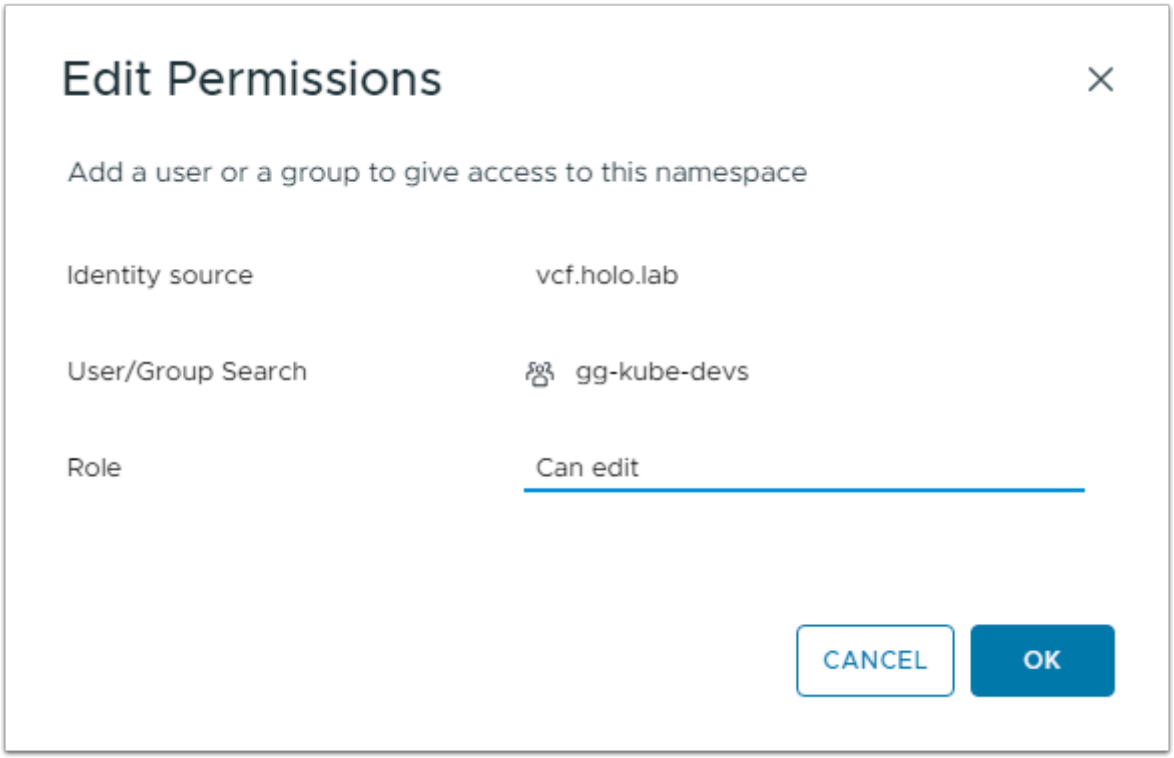
Click **MANAGE PERMISSIONS**

Configure New Namespace



Click **ADD**

Configure New Namespace

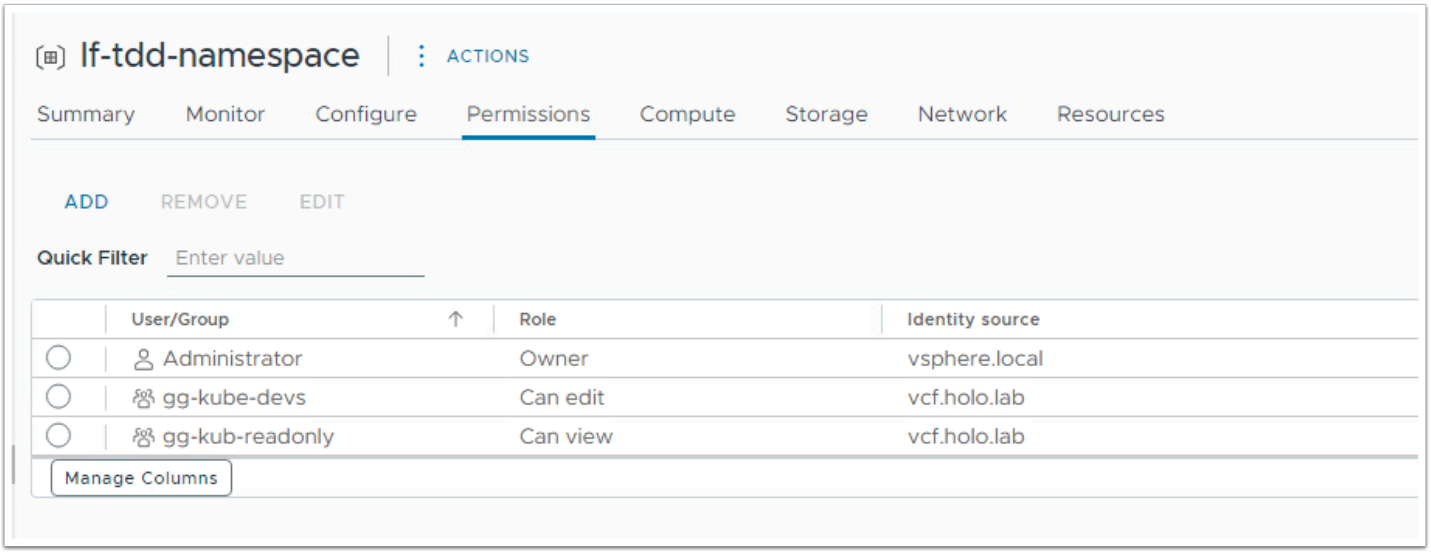


Follow the same workflow and add two more **AD** groups from **SSO** and their roles as follows

- Identity Source: vcf.holo.lab
- User/Group Search: gg-kube-devs
- Role: Can Edit

- Identity Source: vcf.holo.lab
- User/Group Search: gg-kub-readonly
- Role: Can view

Configure New Namespace



When done your permissions should look like this.

There are a few more Namespace configurations you could address, but for the sake of time and the scope of this lab you can move on.

Deploy a TKG Workload Cluster

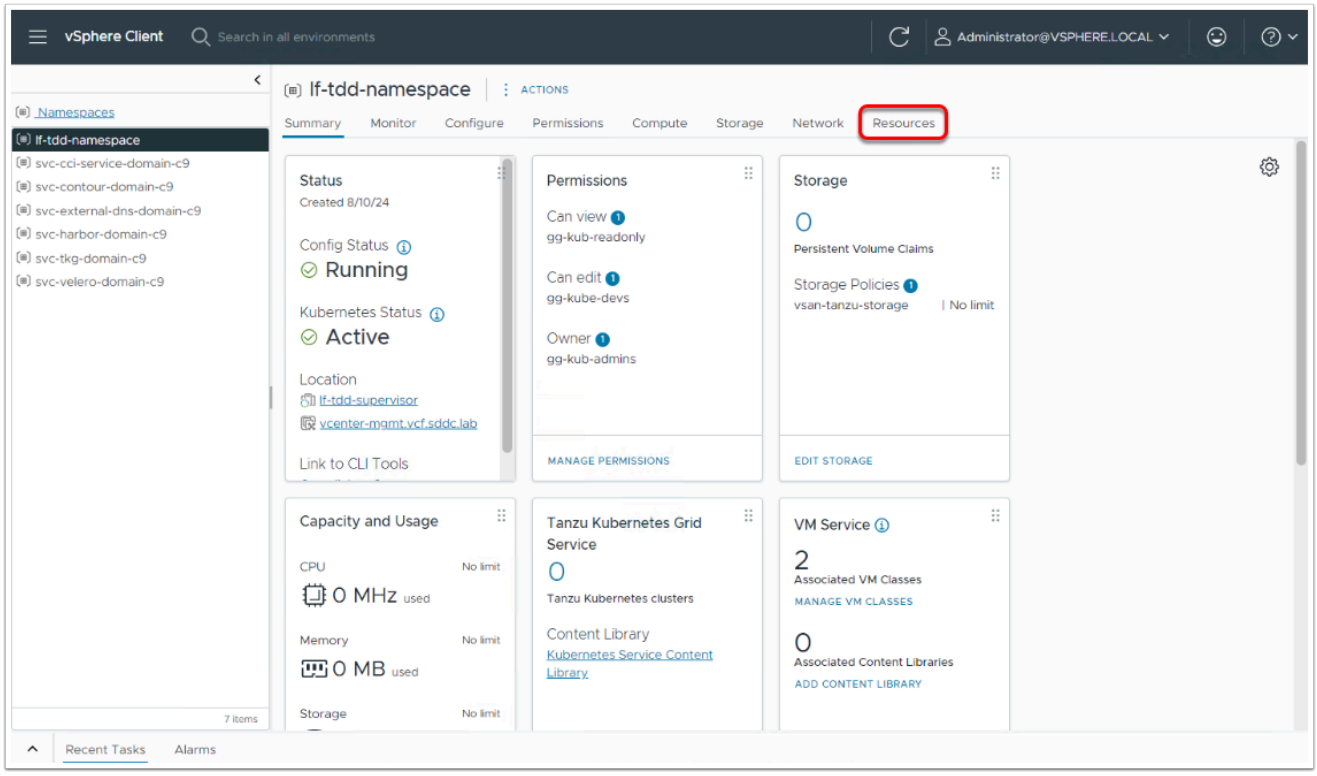
Task description and objectives

A Tanzu Kubernetes cluster is **a full distribution of the open-source Kubernetes container orchestration platform that is built, signed, and supported by VMware**. You can provision and operate Tanzu Kubernetes clusters on the Supervisor Cluster by using the Tanzu Kubernetes Grid Service.

In this lab you are going to use the previously deployed Supervisor Service the Local Consumption Interface (LCI). The LCI is a requirement of each Supervisor Cluster that will be used as an endpoint by the **VCF Automation** Feature, the **Cloud Consumption Interface**.

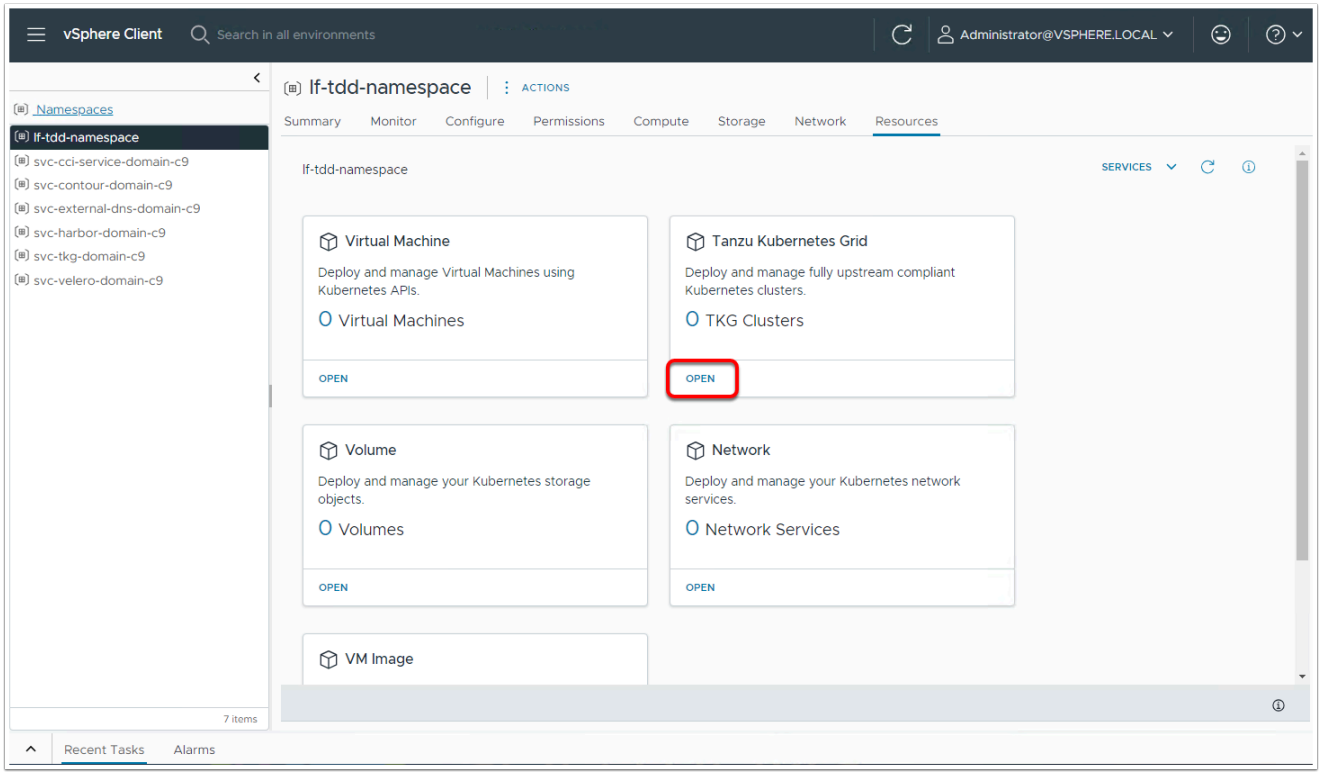
VCF Automation is out of scope and focus of this Technical Deep Dive and we only have a single Supervisor Cluster so you will be working in LCI.

Deploy a TKG Workload Cluster using LCI



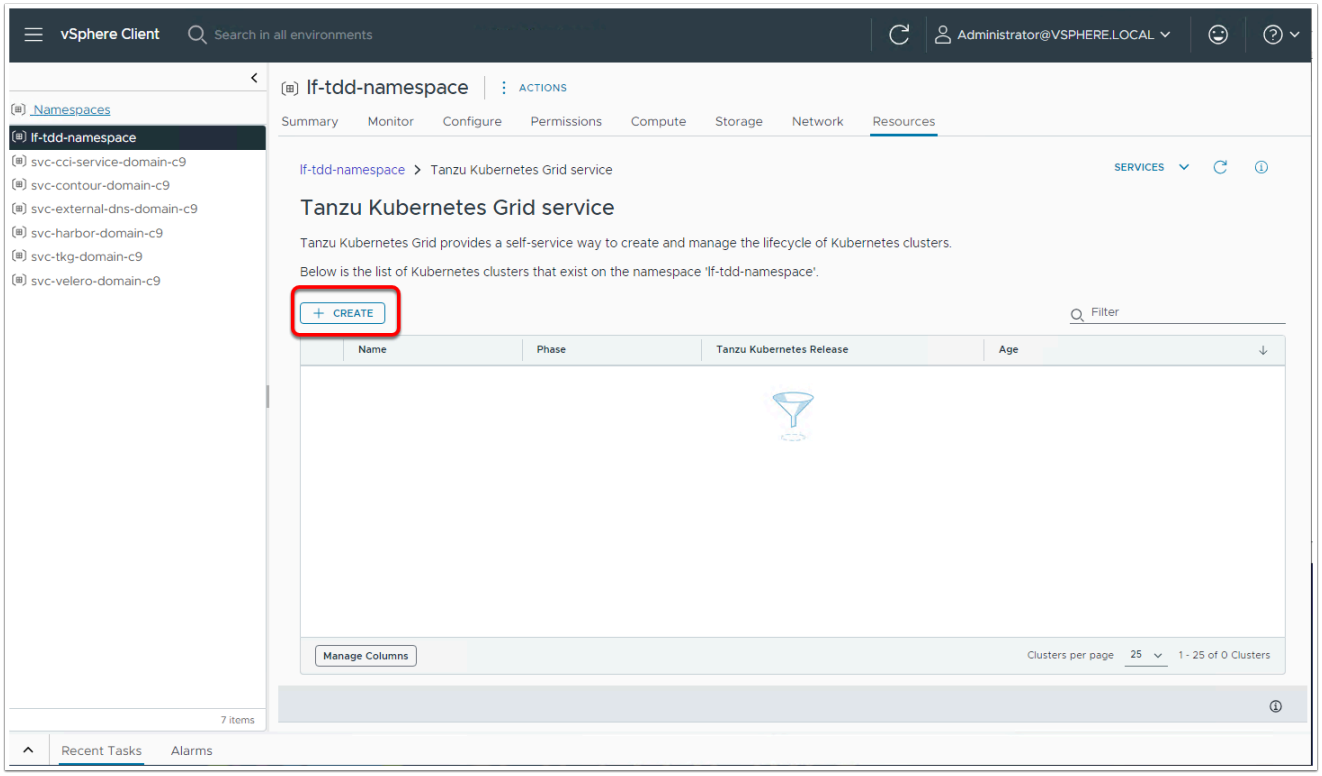
If when you click the **Resources** Tab under the ***lf-tdd-namespace*** **NAMESPACE**, you get no activity, or a form that tells you to install the **Local Consumption Interface** on this Supervisor, Log out and back into the vCenter.

Deploy a TKG Workload Cluster using LCI



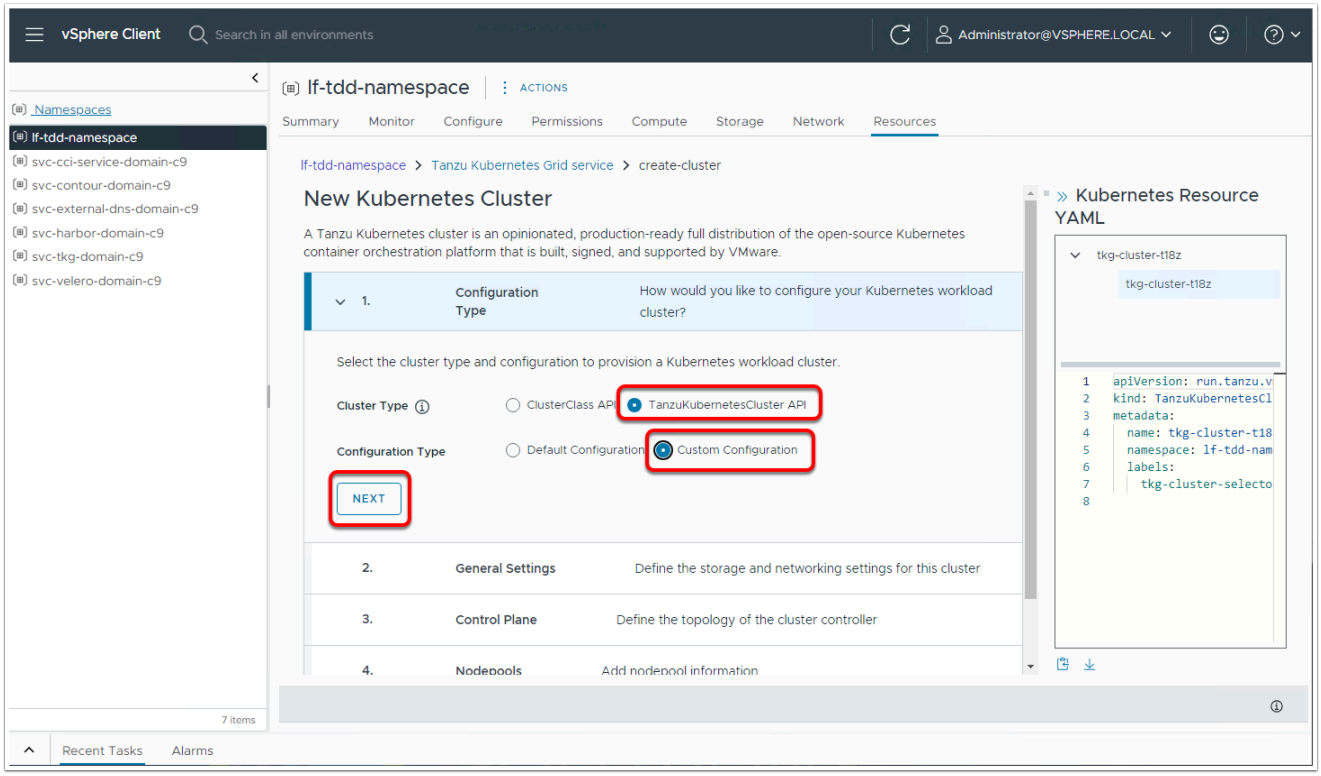
Click **OPEN** on the **Tanzu Kubernetes Grid** Box

Deploy a TKG Workload Cluster using LCI



Click **CREATE**

Deploy a TKG Workload Cluster using LCI

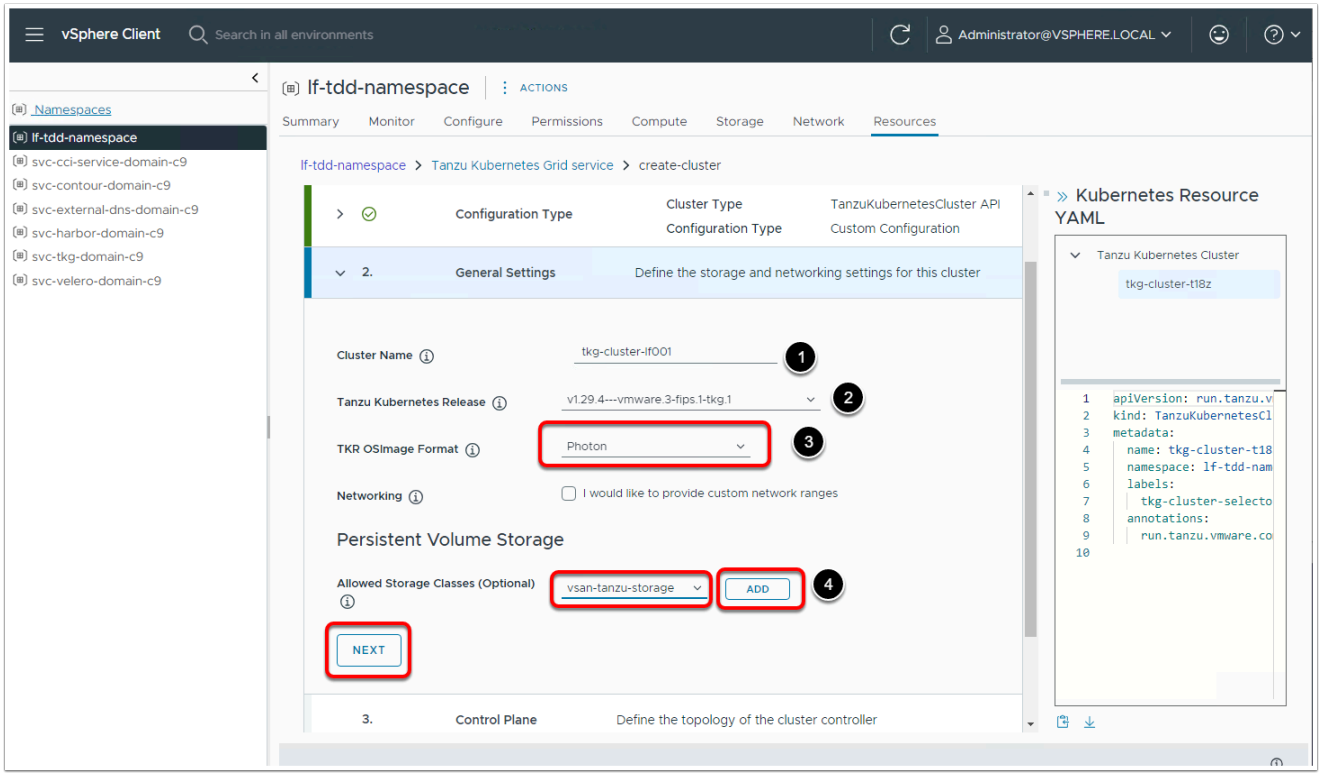


Select the **TanzuKubernetesCluster API** Radio button

Select the **Custom Configuration** Radio button

Click **NEXT**

Deploy a TKG Workload Cluster using LCI



1. Give the cluster a name, i.e.

```
tkg-cluster-lf001
```

2. Leave Tanzu Kubernetes Release as default

3. Select **Photon** from the **TKR OSImage Format** dropdown

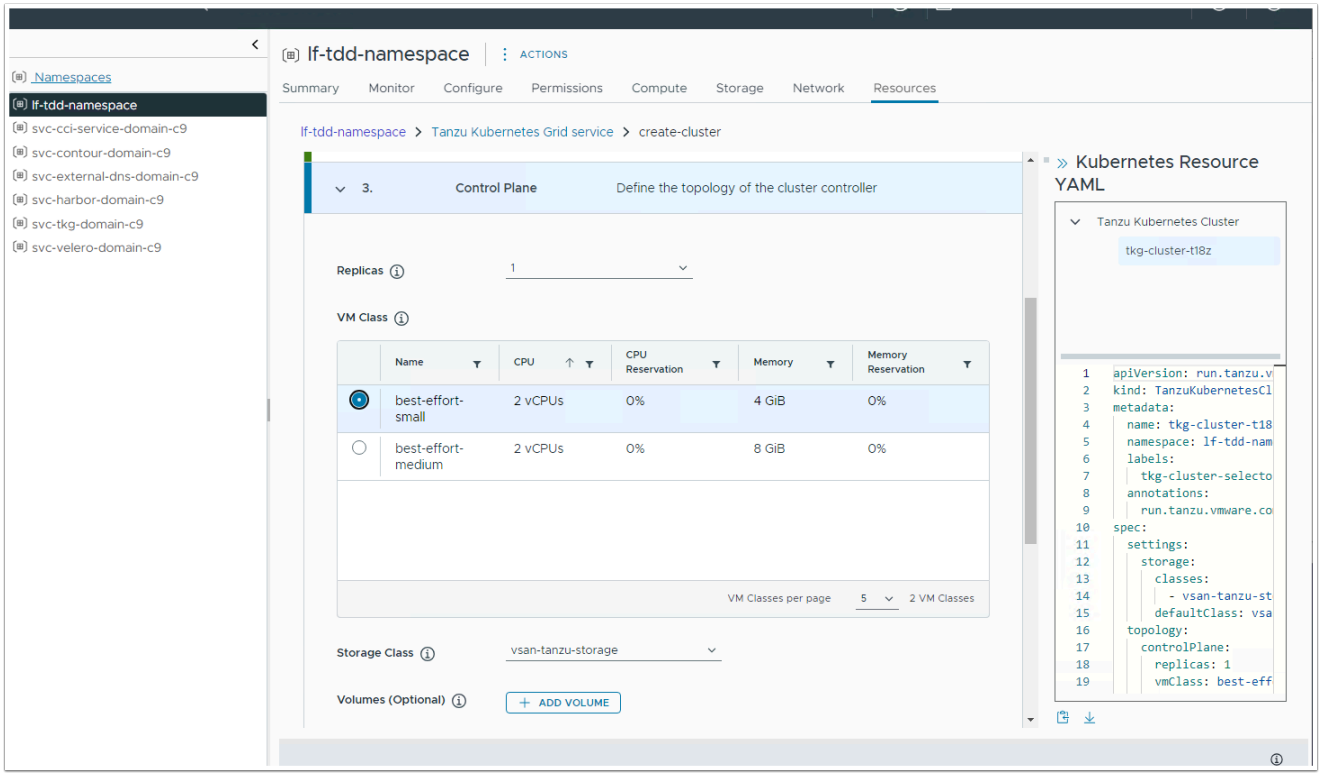
For the **Persistant Storage Classes (optional)** section, select

```
vsan-tanzu-storage
```

Click **ADD**

Click **NEXT**

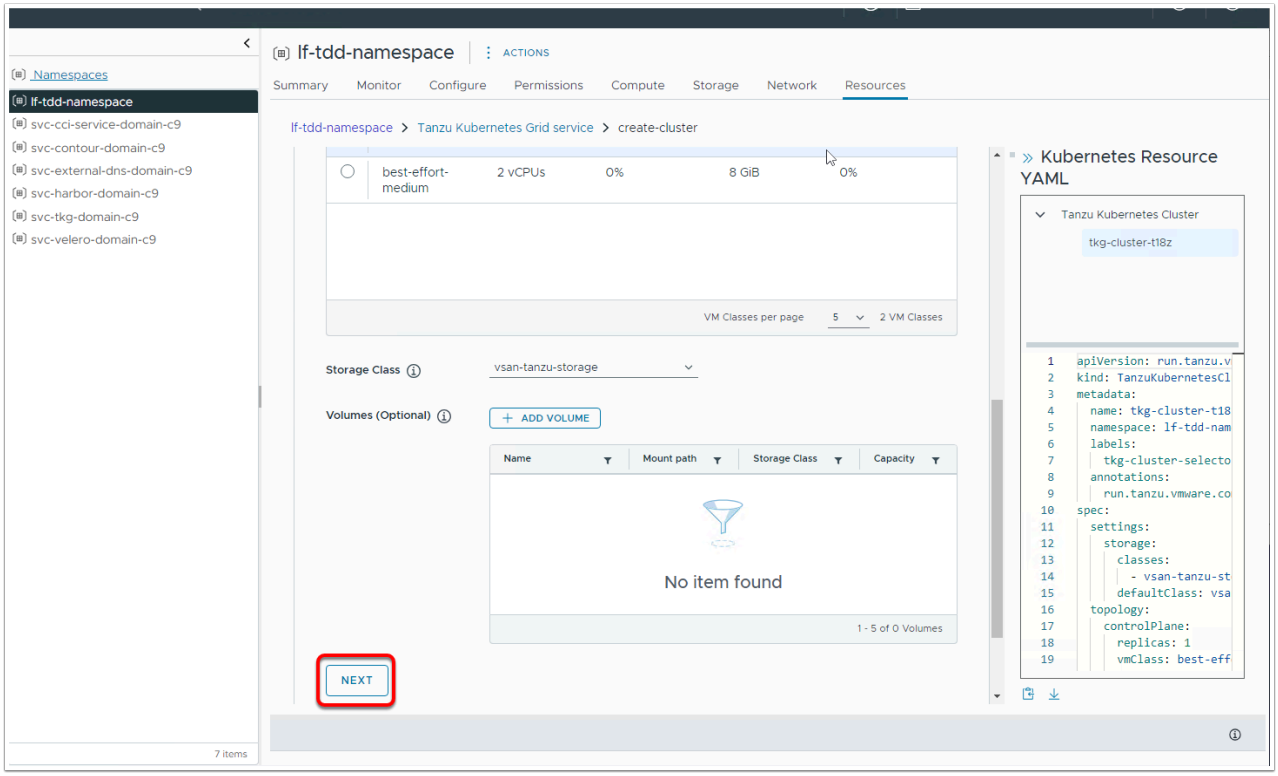
Deploy a TKG Workload Cluster using LCI



Leave the **Control Plane Replica** count at 1

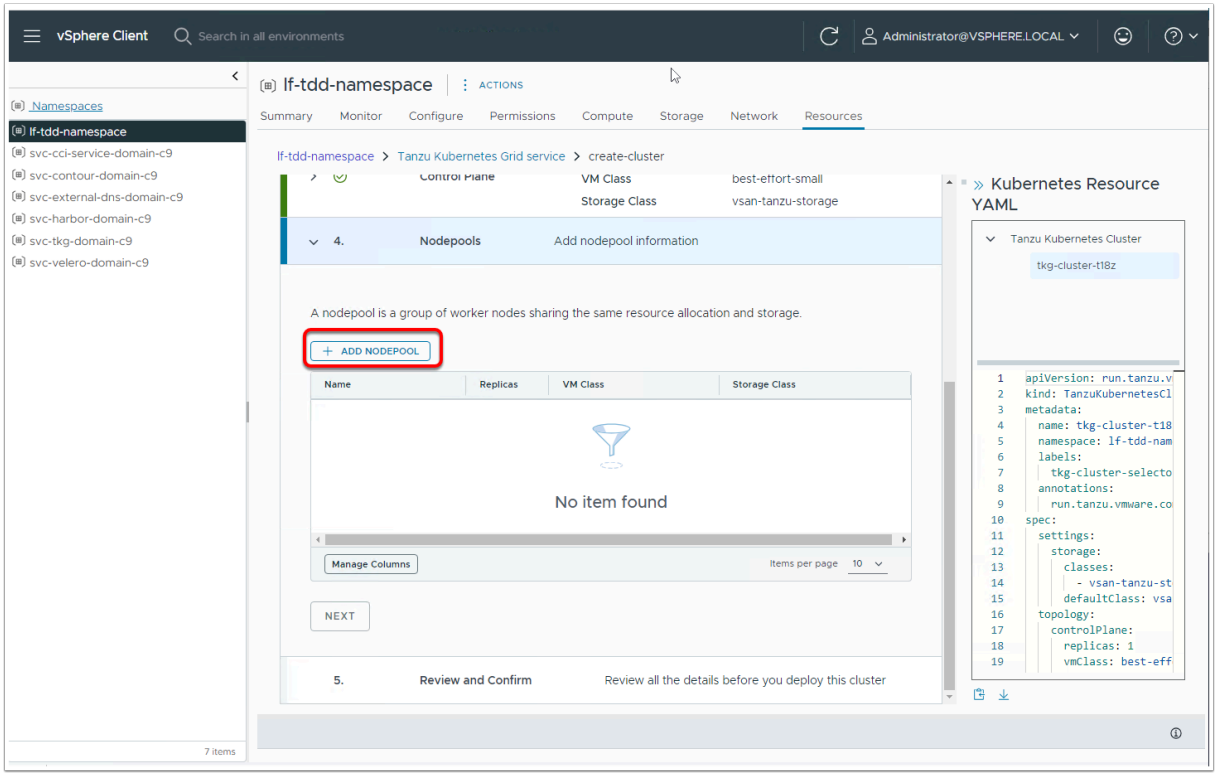
Check the Radio button for **best-effort-small**

Deploy a TKG Workload Cluster using LCI



Scroll down and click **NEXT**

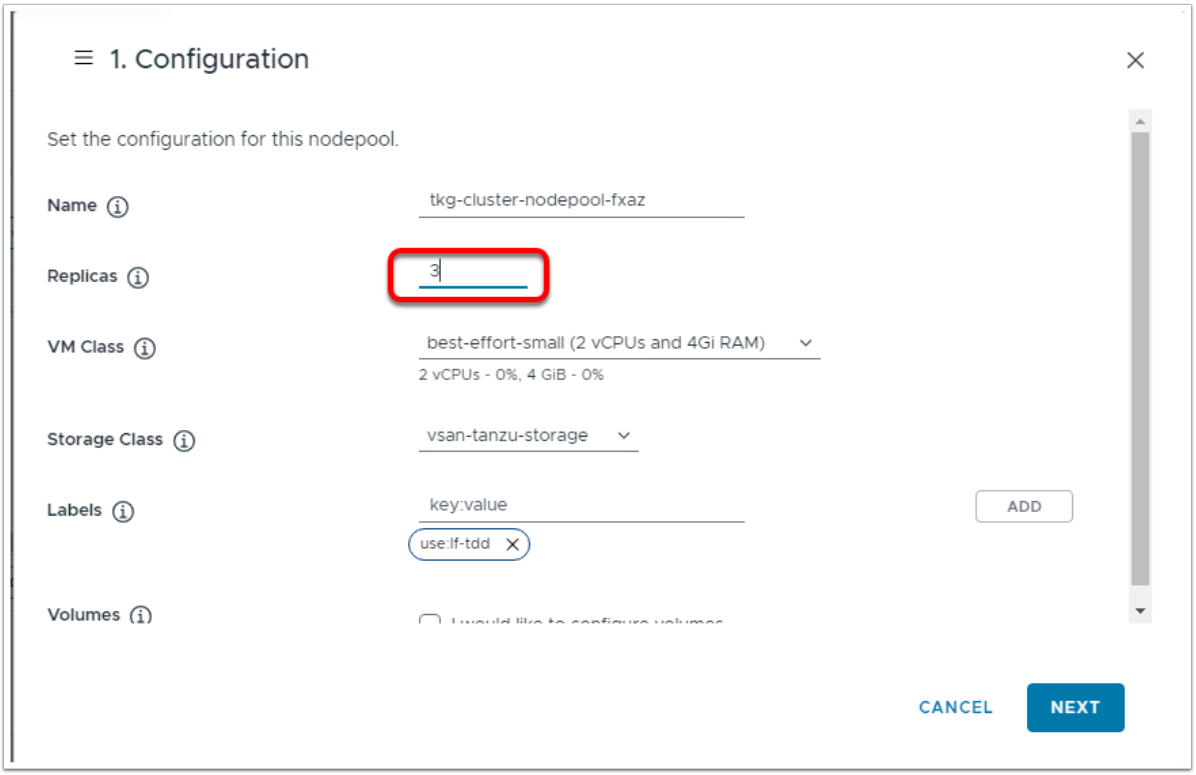
Deploy a TKG Workload Cluster using LCI



Worker nodes are grouped in NODEPOOLS

Click **+ ADD NODEPOOL**

Deploy a TKG Workload Cluster using LCI



Set the Replicas count to 3

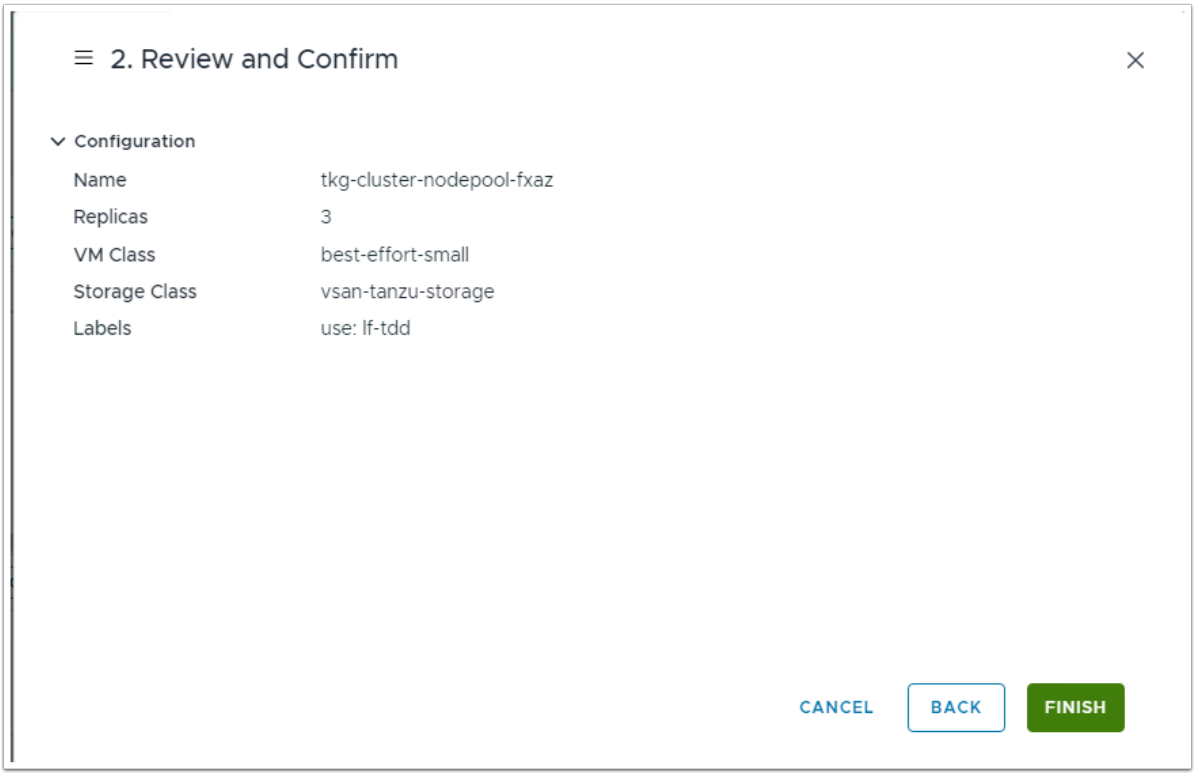
In Labels enter

```
use:lf-tdd
```

Click ADD to add this key:value pare as a label to the **NODEPOOL**

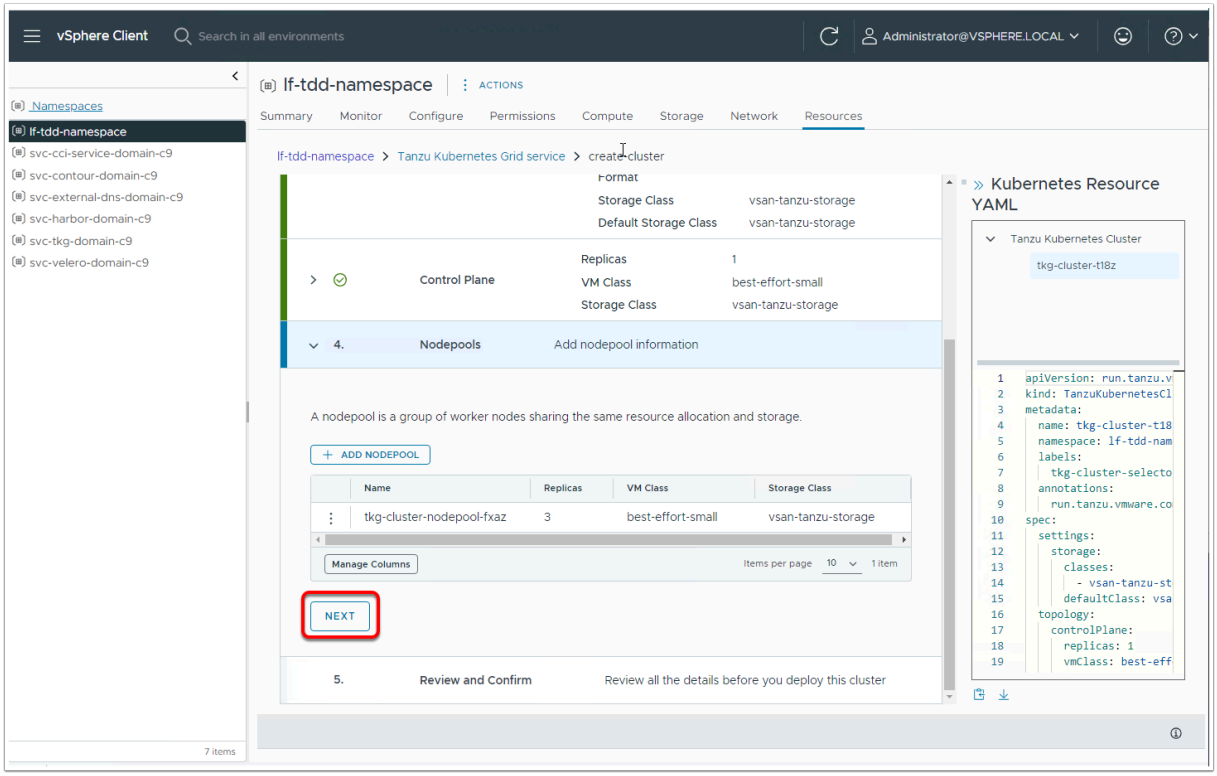
Click **NEXT**

Deploy a TKG Workload Cluster using LCI



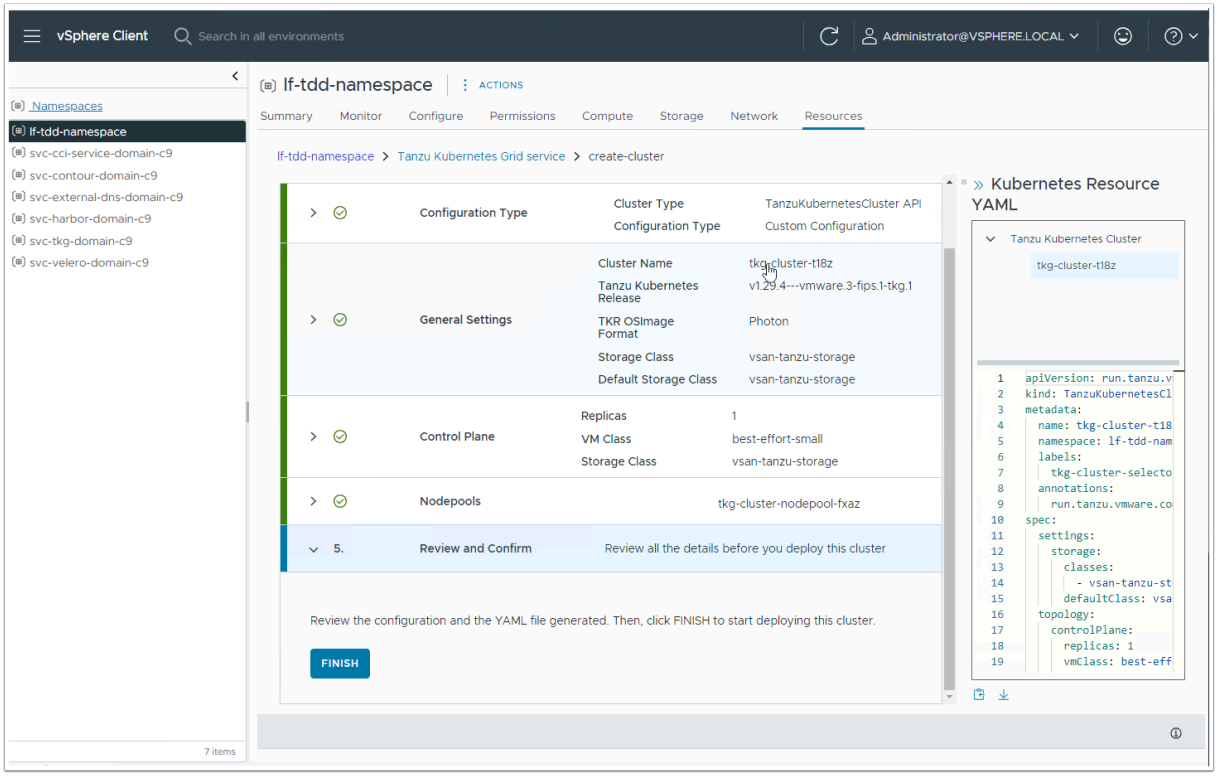
Click **FINISH**

Deploy a TKG Workload Cluster using LCI



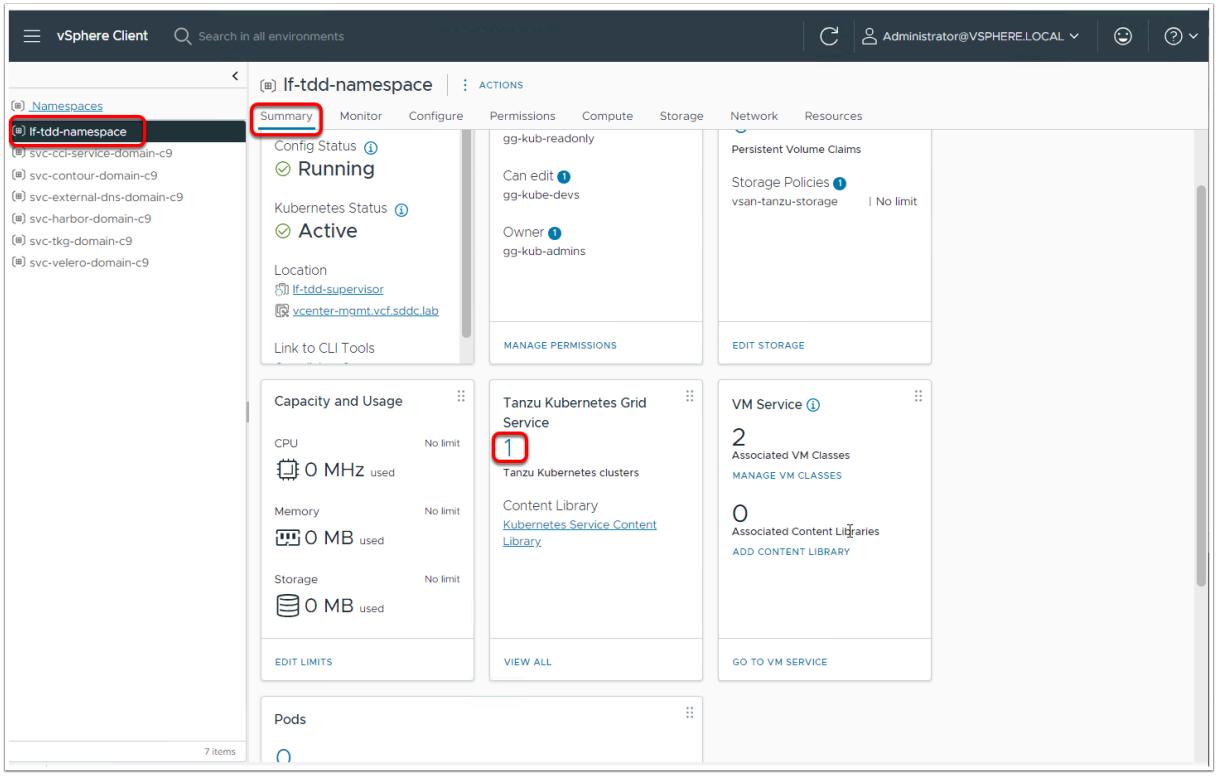
Click **NEXT**

Deploy a TKG Workload Cluster using LCI



Click **FINISH**

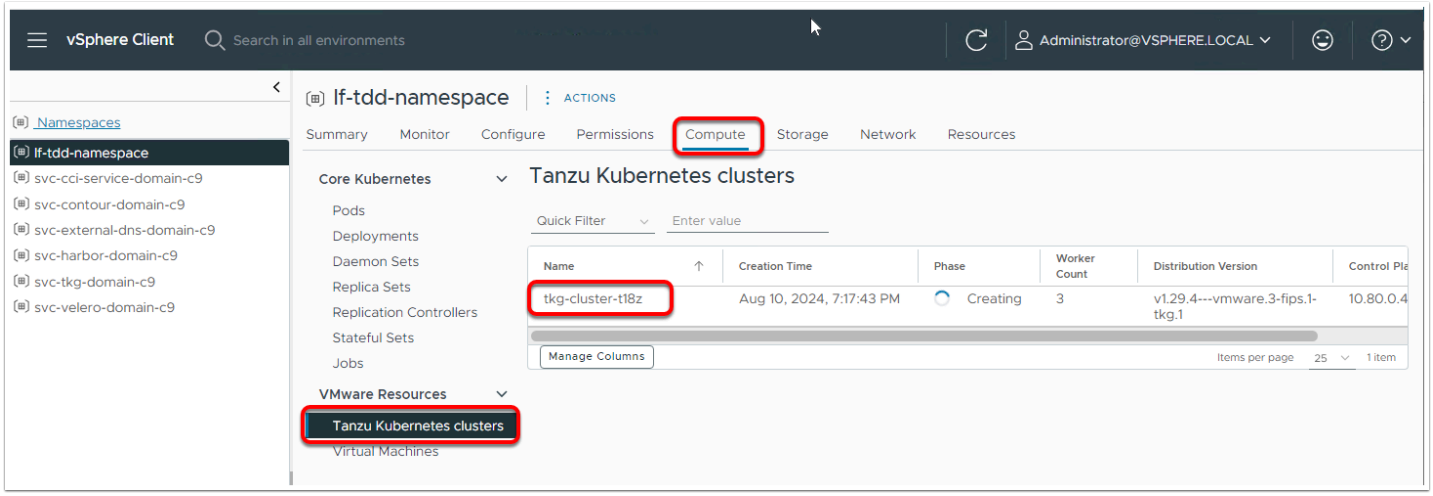
Deploy a TKG Workload Cluster using LCI



From here you can make sure you are on the correct Namespace

Then Confirm under **Summary** --> **Tanzu Kubernetes Grid** shows 1 new cluster

Deploy a TKG Workload Cluster using LCI

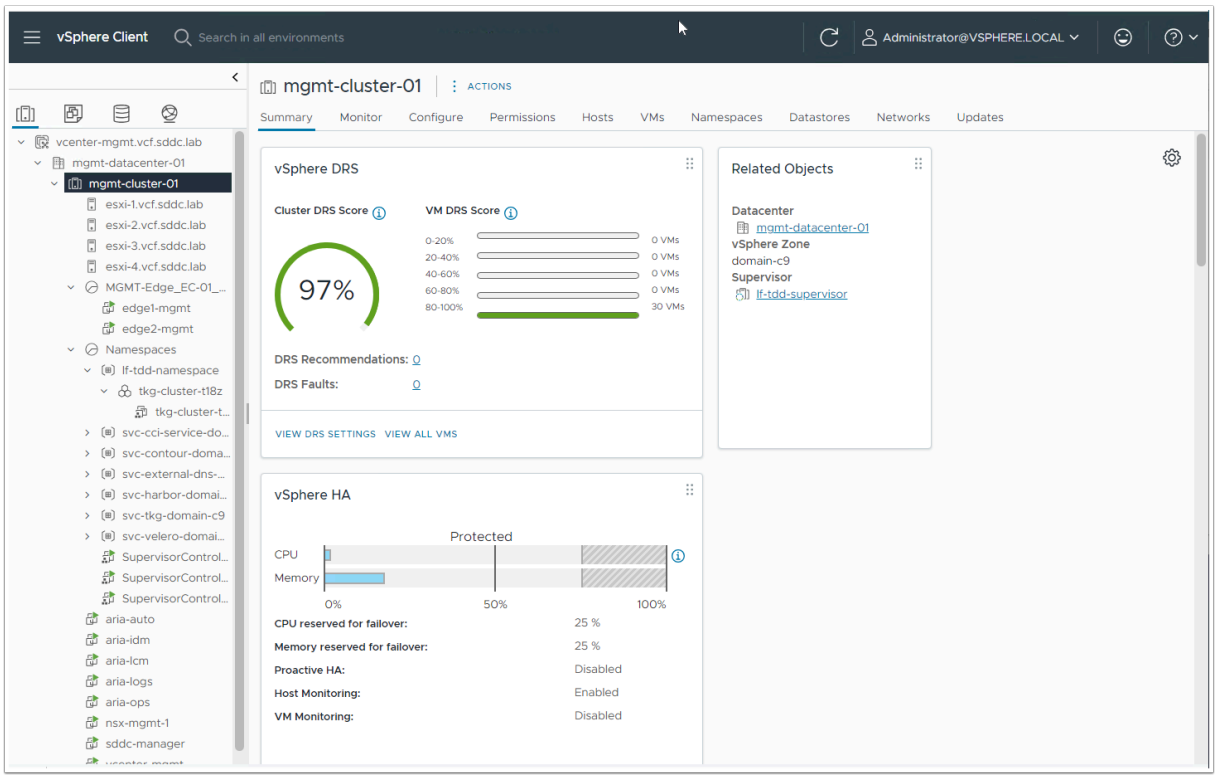


Click **Compute**

Click **Tanzu Kubernetes** clusters

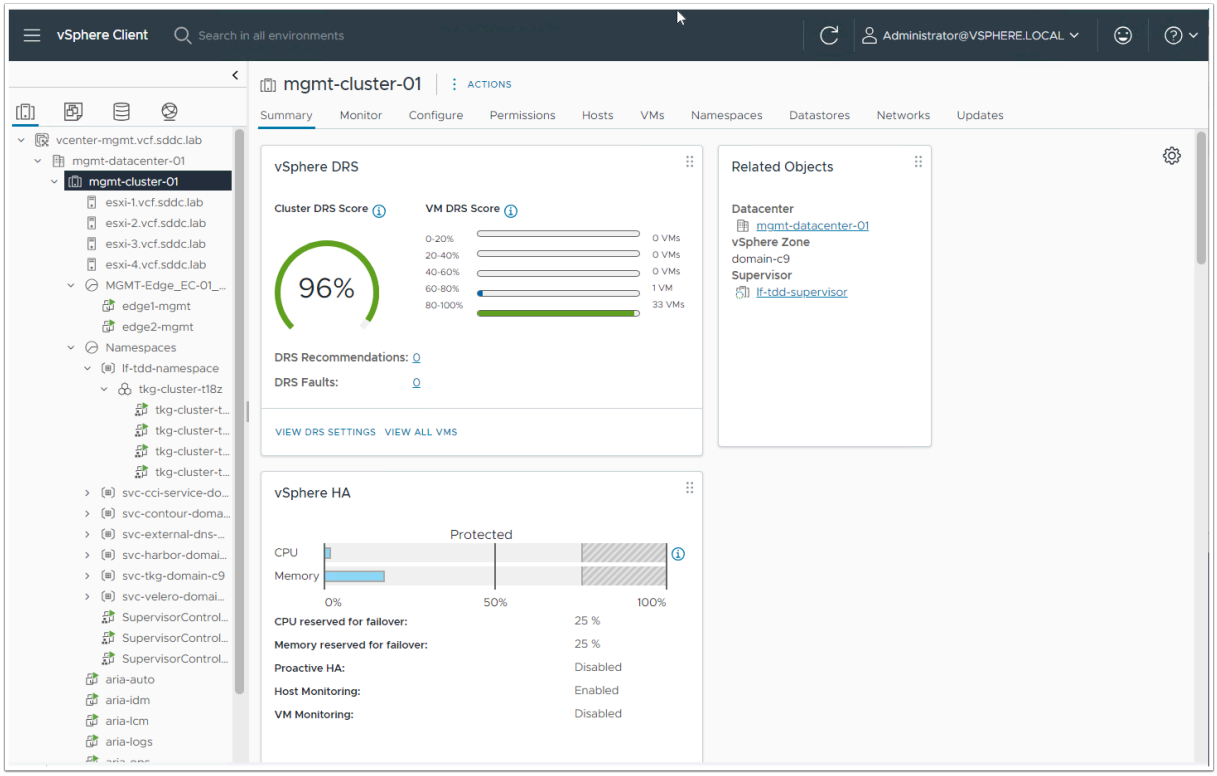
you can see your new TKGs cluster being Created

Deploy a TKG Workload Cluster using LCI

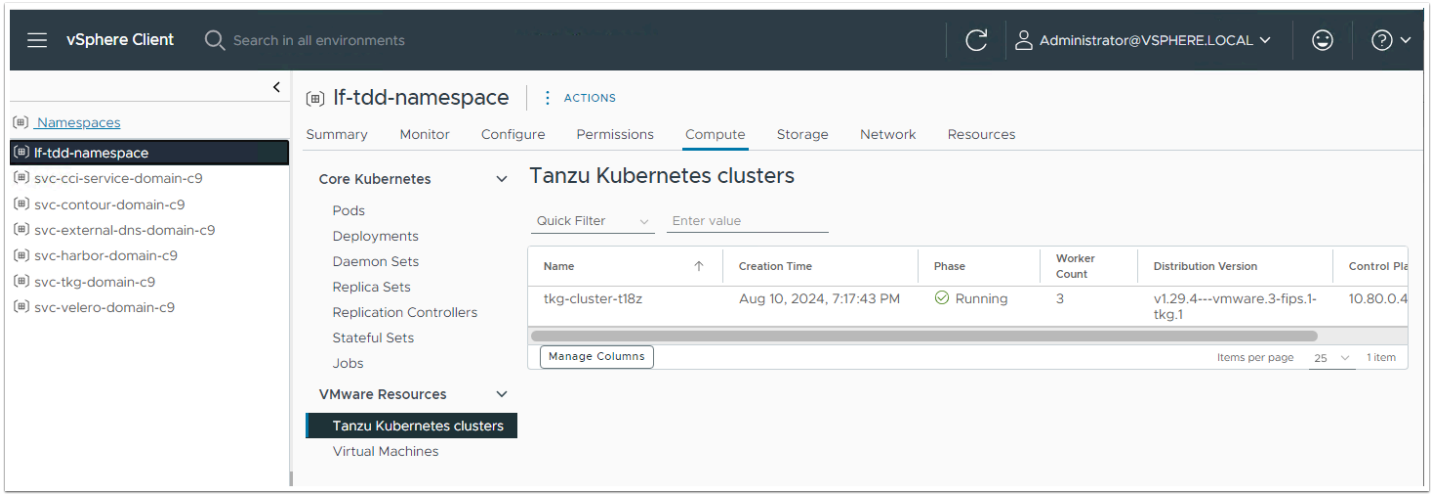


Switch to **Inventory** --> **Hosts and Clusters** view and drill down into **Namespaces** --> ***your namespace*** --> ***Your new tkgs cluster***

Deploy a TKG Workload Cluster using LCI



You can follow the deployment of the **controlplane** and **workload** cluster nodes



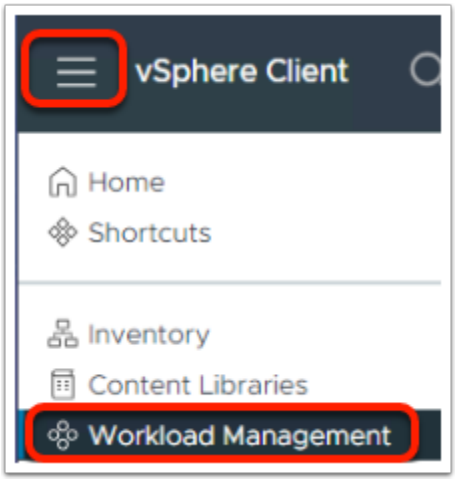
And in a few minutes (30 or so in this lab) you should have a TKGs cluster running

K8s Tools for IaaS Control Pane Management - CLI

Task description and objectives

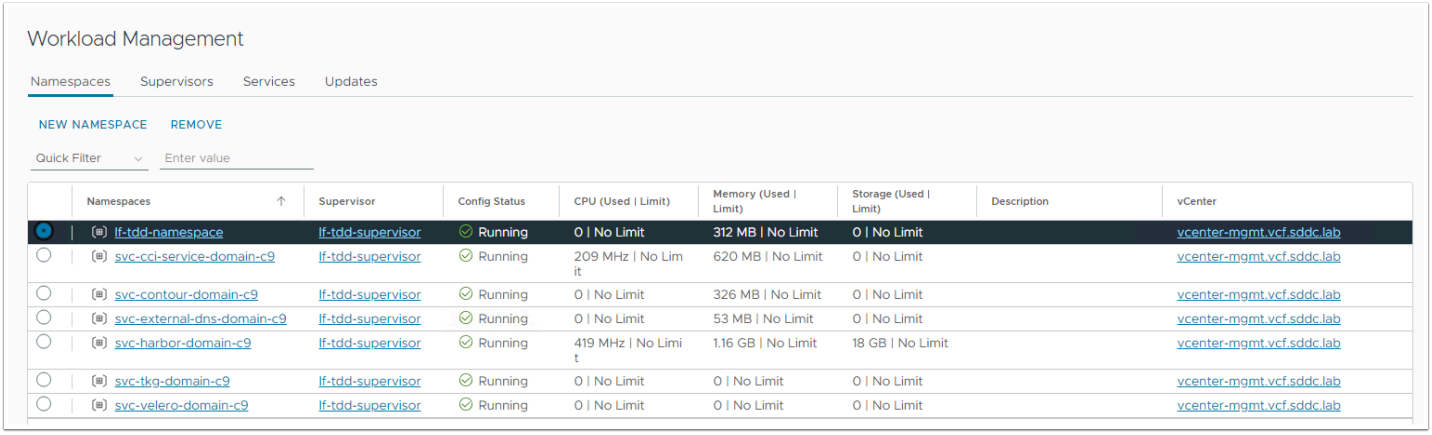
Now that you have the Supervisor Cluster and a TKG Workload cluster deployed, it's time to deploy the tools needed to connect, manage, and make use of the eco system.

From the vSphere client



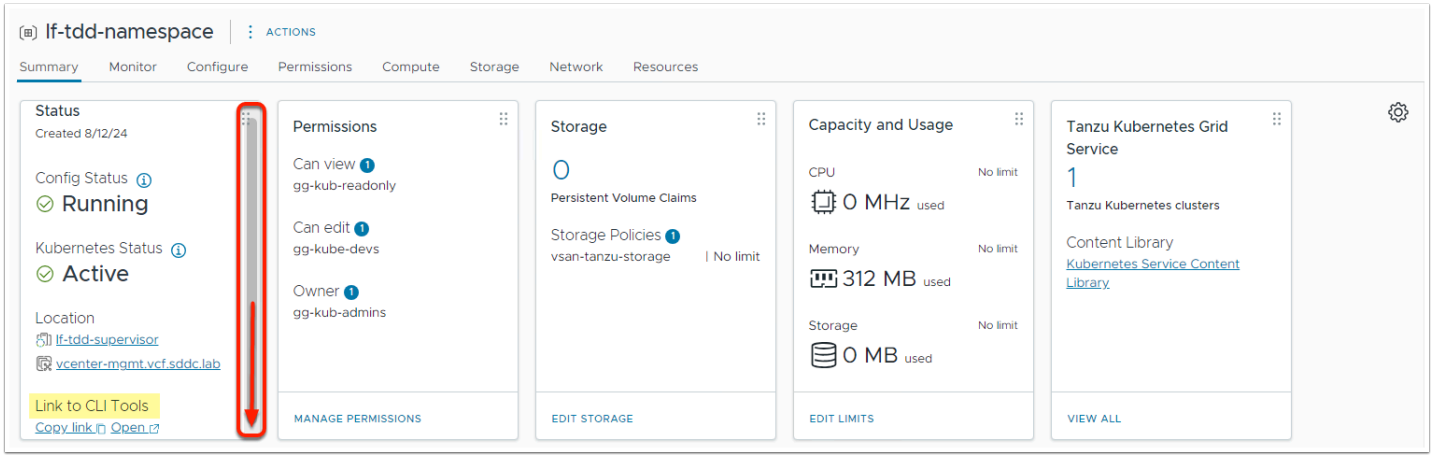
From the vSphere Client home menu, select **Workload Management**.

Download the K8s Tools for IaaS Management Service



In the Workload Management window click on the **lf-tdd-namespace** namespace

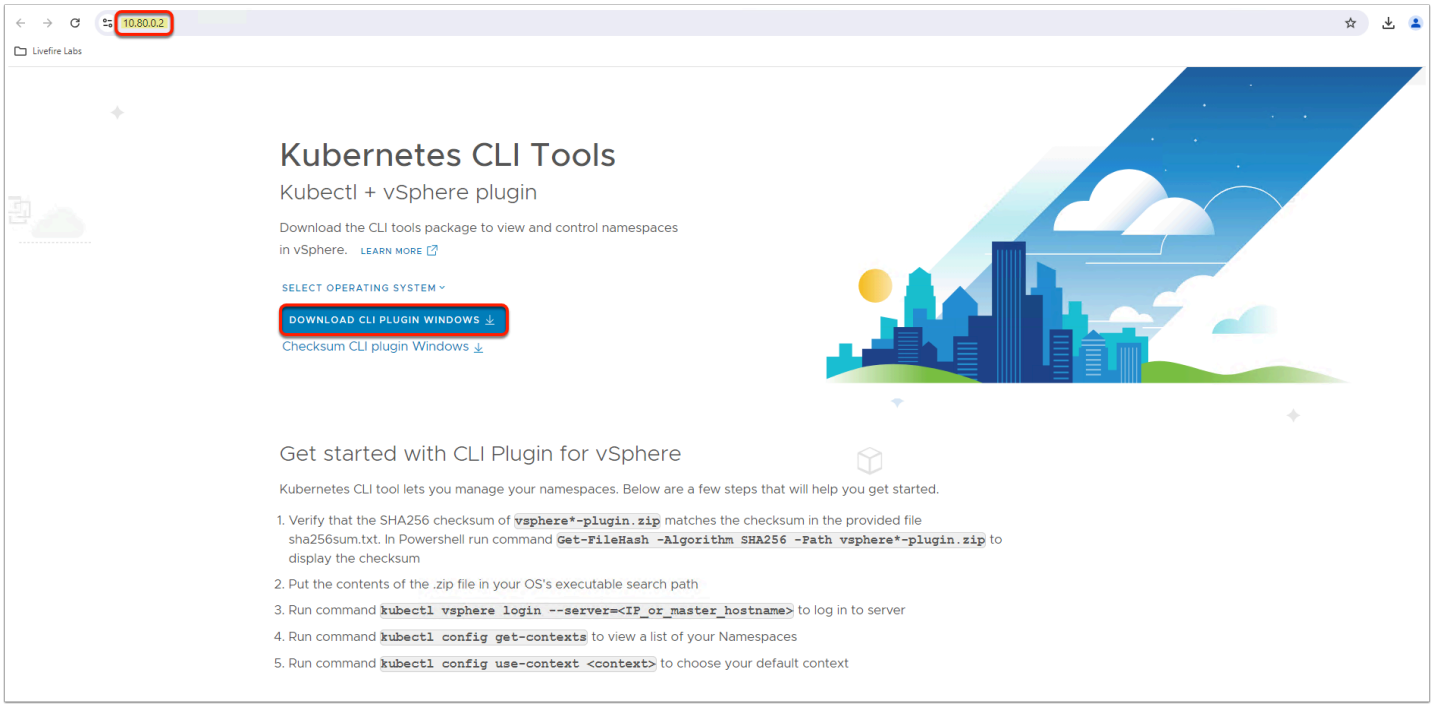
Download the K8s Tools for IaaS Management Service



In the namespace window make sure to scroll down to the bottom in the Satus section.

Locate the **Link to CLI Tools** and click on **Open** (opens in a new tab)

Download the K8s Tools for IaaS Management Service

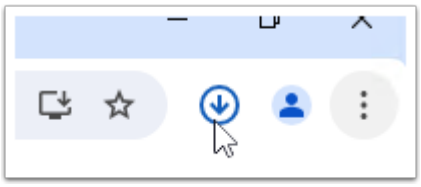


If the Your connection is not private message appears, click Advanced and then click Proceed to 10.80.0.2 (unsafe).

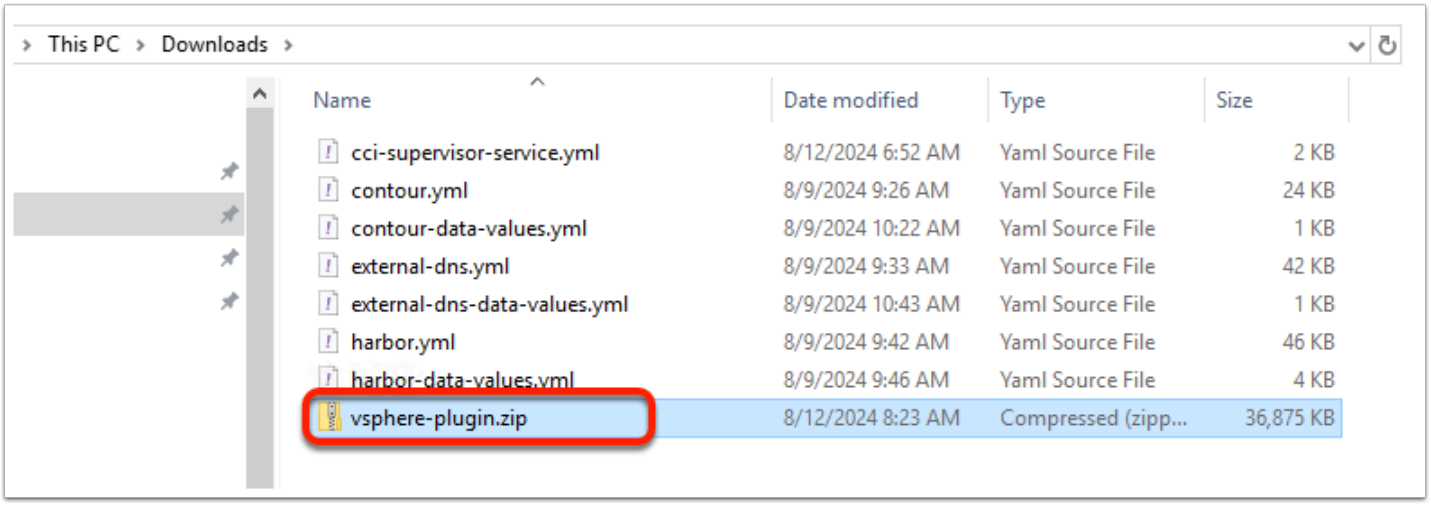
The browser redirects to the Kubernetes CLI Tools for vSphere download page. The version of your OS is automatically recognized.

Click on **DOWNLOAD CLI PLUGIN WINDOWS**

> ⚠ NOTE: A new browser tab may, open. Or you may not notice any change to the browser at all. Look at the right top corner, for the download icon to show that the download has started.

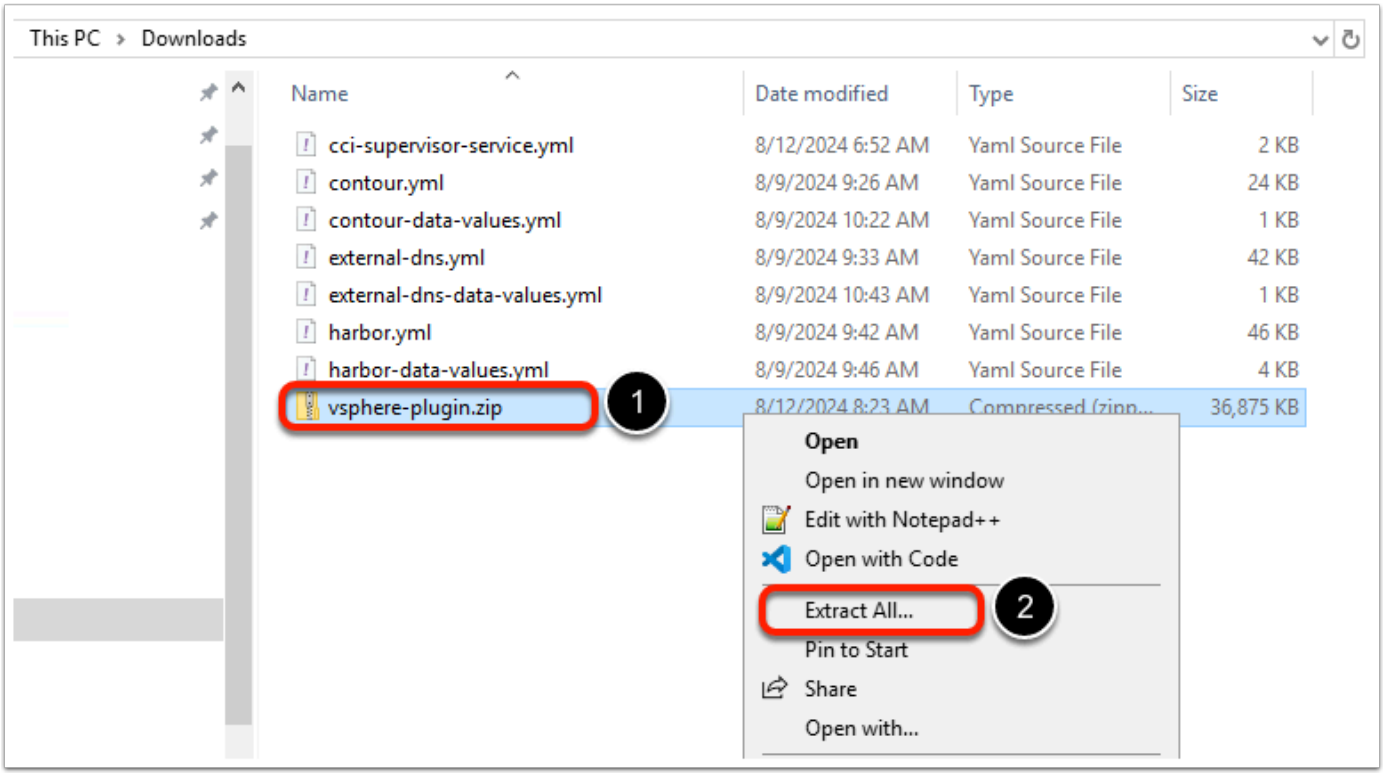


> Please note the **IP address of your Supervisor.** You will use it shortly in the following activities!
>
> **<SUPERVISOR_IP> = 10.80.0.2**



This will download the needed CLI tools in your Downloads folder.

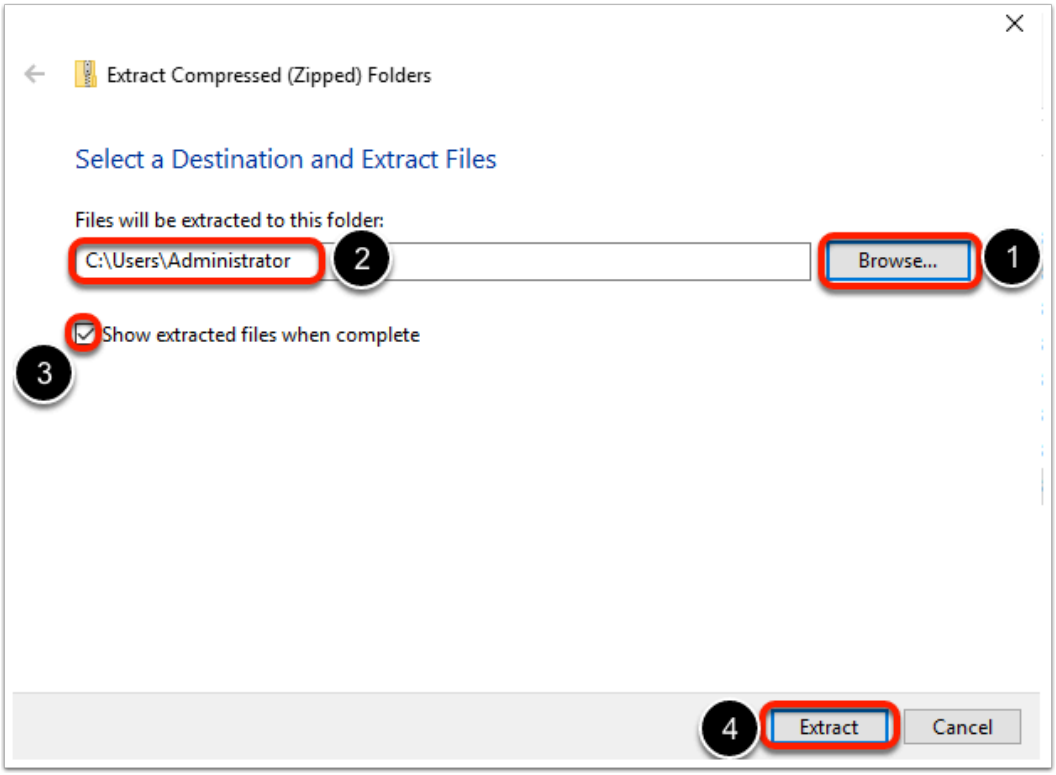
Install the K8s Tools for IaaS Management Service



In your Downloads folder

1. Right-click on the **vsphere-plugin.zip** file
2. Choose the **Extract All** option

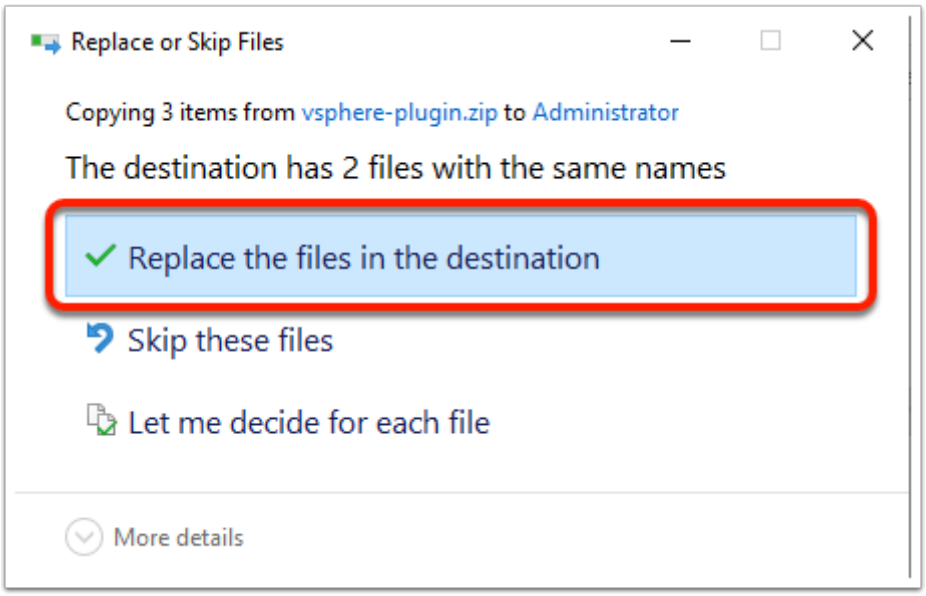
Install the K8s Tools for IaaS Management Service



In the **Extract Compressed (Zipped) Folders** window

1. Click on Browse, select C:\Users\Administrator folder
2. Mark Show extracted files when complete
3. Click Extract

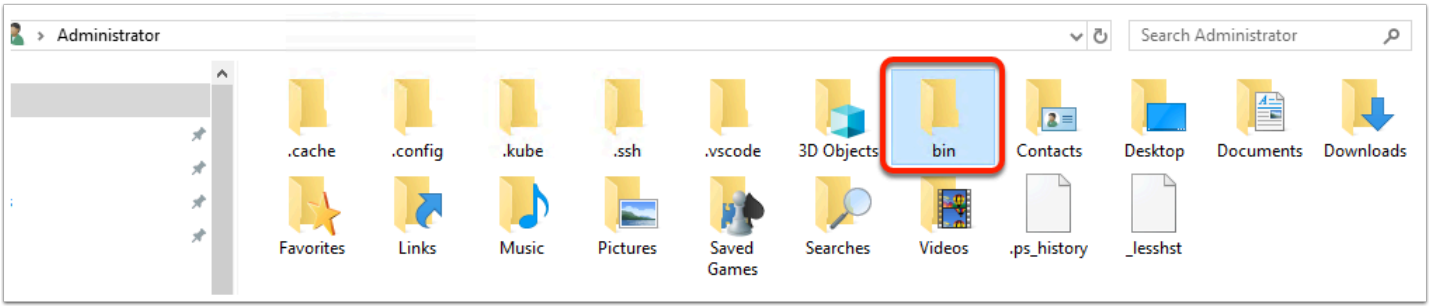
```
C:\Users\Administrator
```



Confirm Replace the files in the destination

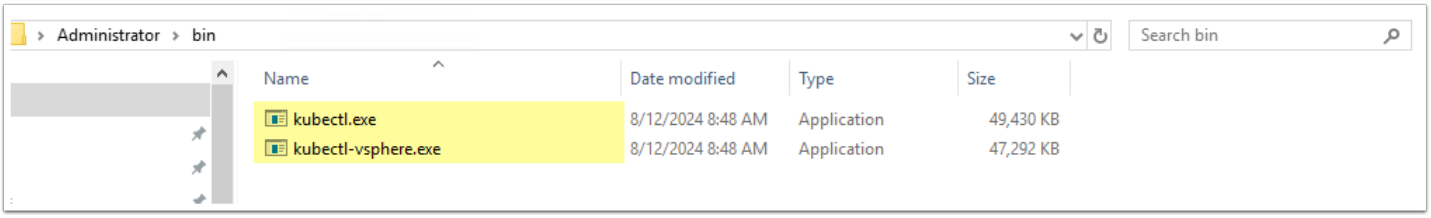
Install the K8s Tools for IaaS Management Service

Replacing will guarantee up-to date version of the cli binaries on par with your Supervisor



Double click on the **bin** folder

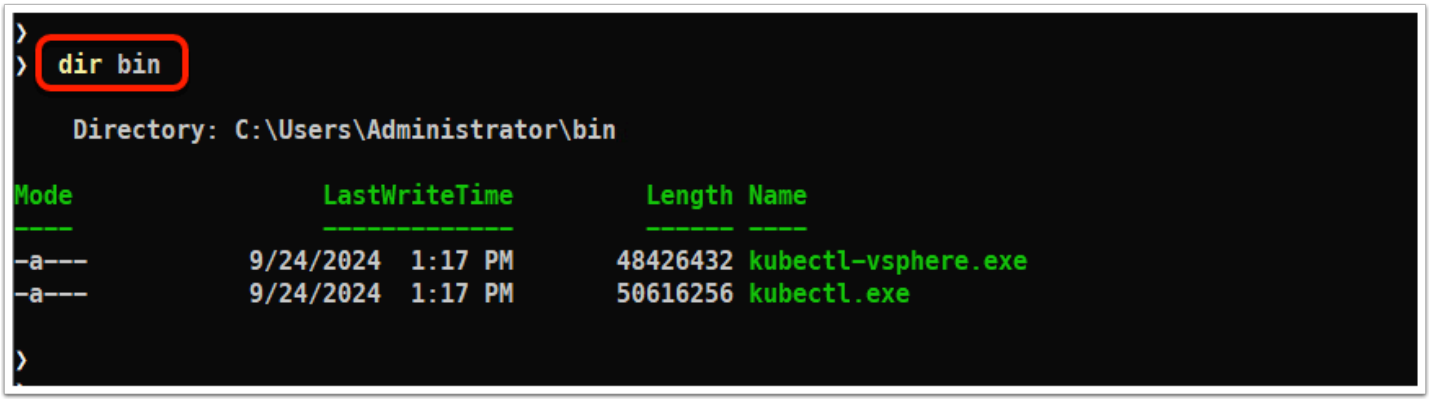
Install the K8s Tools for IaaS Management Service



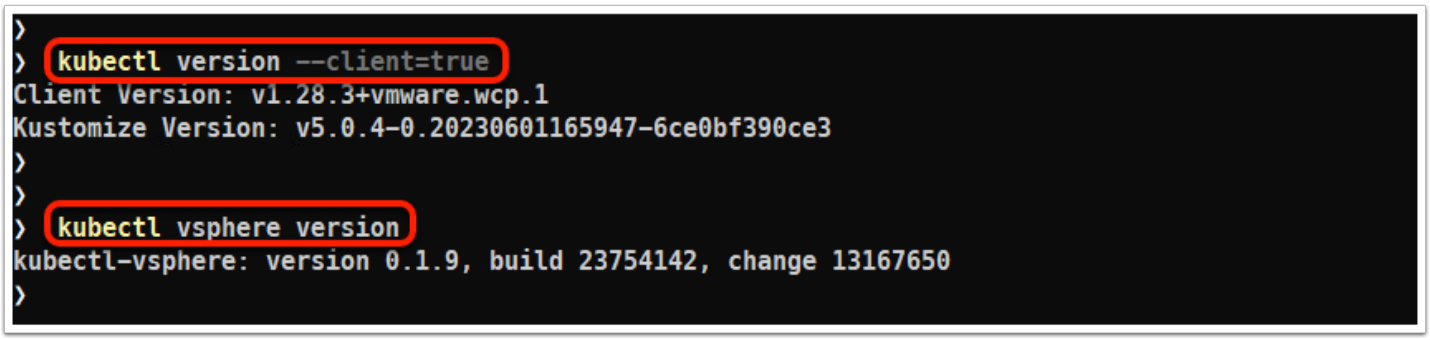
These are your Kubernetes CLI tools for vSphere

Install the K8s Tools for IaaS Management Service

From your desktop taskbar, Open a new **PowerShell Prompt** and check the binaries




```
dir bin
```



Check the Kubernetes CLI version to ensure that the installation was successful.

```
kubectl version --client=true
```

```
kubectl vsphere version
```

Now you are ready to use the CLI and log in to your Supervisor.

Install Tanzu CLI & Plugins

To install the Tanzu CLI for use with Tanzu Kubernetes, you install a compatible version of the **Tanzu Core CLI** and the **Tanzu CLI plugins** . Commands provided by these plugins enable cluster and package operations.

A Tanzu CLI plugin is an executable binary that packages a group of CLI commands. The core CLI has some command groups built in, and CLI plugins extend the CLI with additional command groups.

Specific plugins are relevant to different products and to different contexts that you connect the CLI to, based on the context type. Each product designates its relevant plugins as either standalone or context-scoped, as described in CLI Core, Plugins, Plugin Groups, and Products.

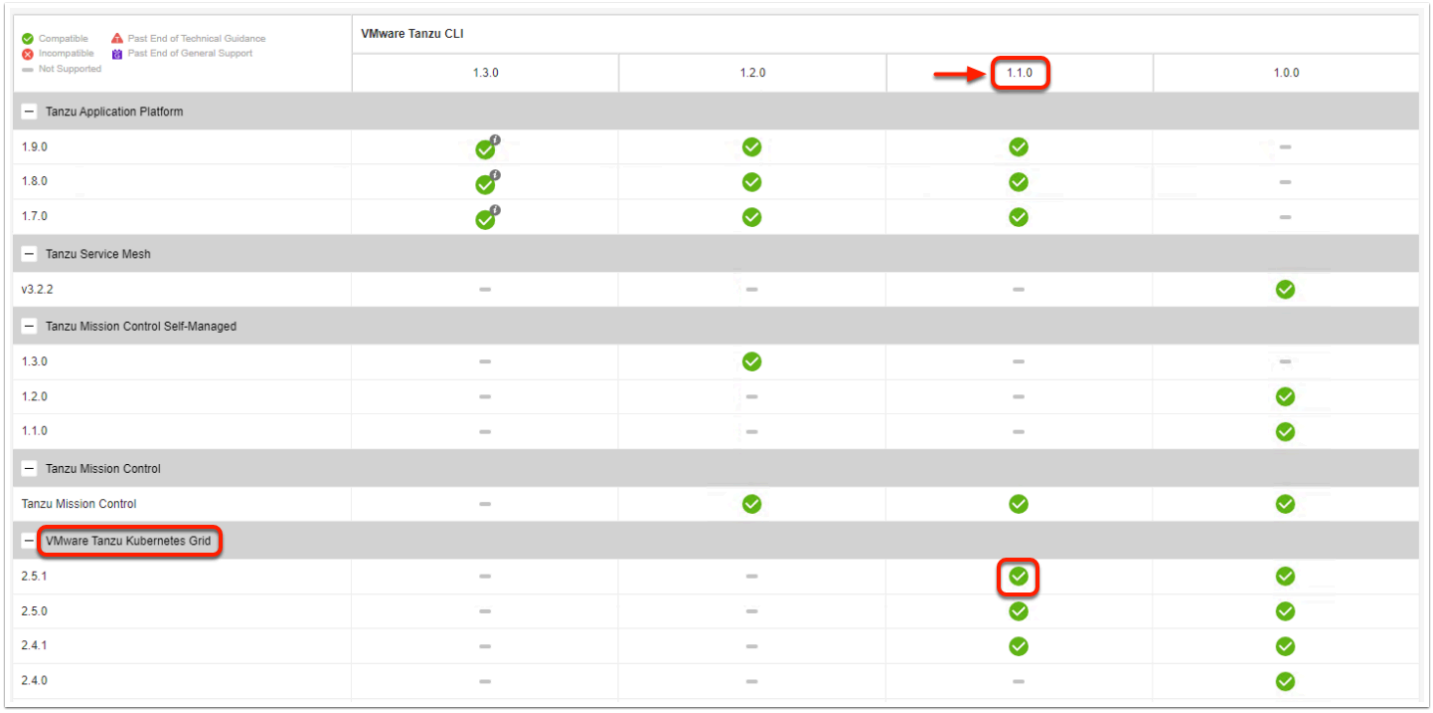
Context-scoped plugins: When you create a Tanzu CLI context to connect to a product, context-scoped plugins install automatically if they are not already installed. As a backup, if plugins do not automatically install, you can install them by running tanzu plugin sync with the CLI set to the desired context.

Standalone plugins: For some but not all products, you need to install standalone plugins by running tanzu plugin install as described below. To find out whether you need to install standalone plugins for your product, see CLI Core, Plugins, Plugin Groups, and Products.

After completing the steps in Install the Tanzu CLI, follow the instructions to install standalone Tanzu CLI plugins. You can use one command to install all plugins in a plugin group or install each plugin individually.

Install Tanzu Core CLI

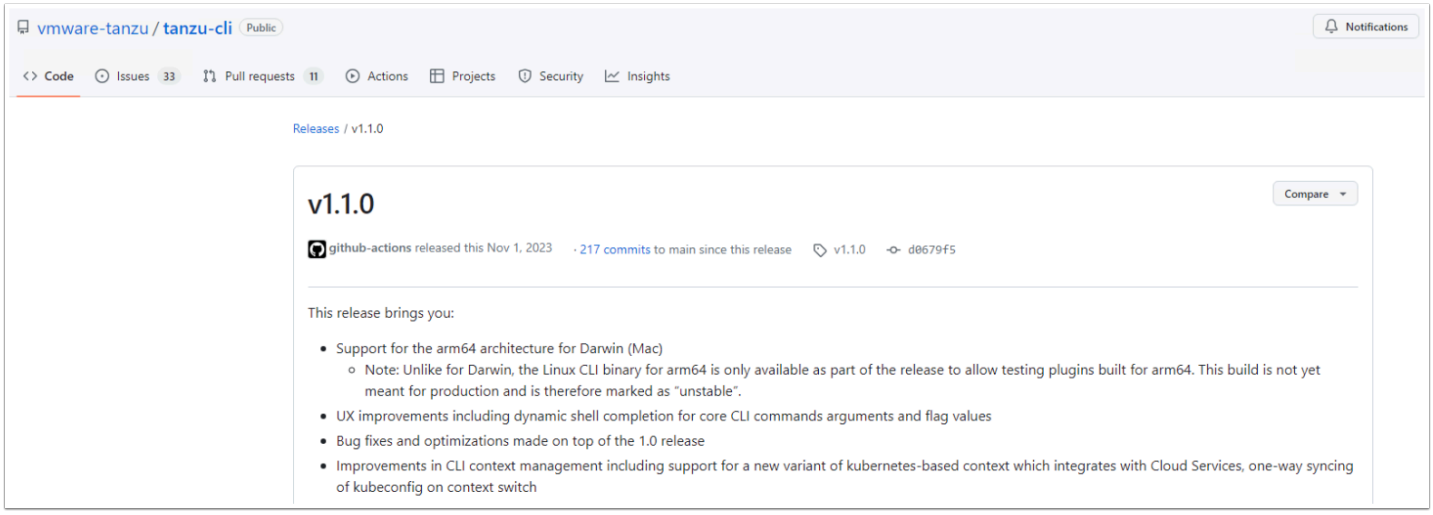
As Taznzu CLI is a multiple solutions oriented CLI, you need to check the [interoperability matrix] and choose the correct version to start with:



Compatible / Incompatible / Not Supported; Past End of Technical Guidance / Past End of General Support	VMware Tanzu CLI			
	1.3.0	1.2.0	1.1.0	1.0.0
Tanzu Application Platform				
1.9.0	✓	✓	✓	–
1.8.0	✓	✓	✓	–
1.7.0	✓	✓	✓	–
Tanzu Service Mesh				
v3.2.2	–	–	–	✓
Tanzu Mission Control Self-Managed				
1.3.0	–	✓	–	–
1.2.0	–	–	–	✓
1.1.0	–	–	–	✓
Tanzu Mission Control				
Tanzu Mission Control	–	✓	✓	✓
VMware Tanzu Kubernetes Grid				
2.5.1	–	–	✓	✓
2.5.0	–	–	✓	✓
2.4.1	–	–	✓	✓
2.4.0	–	–	–	✓

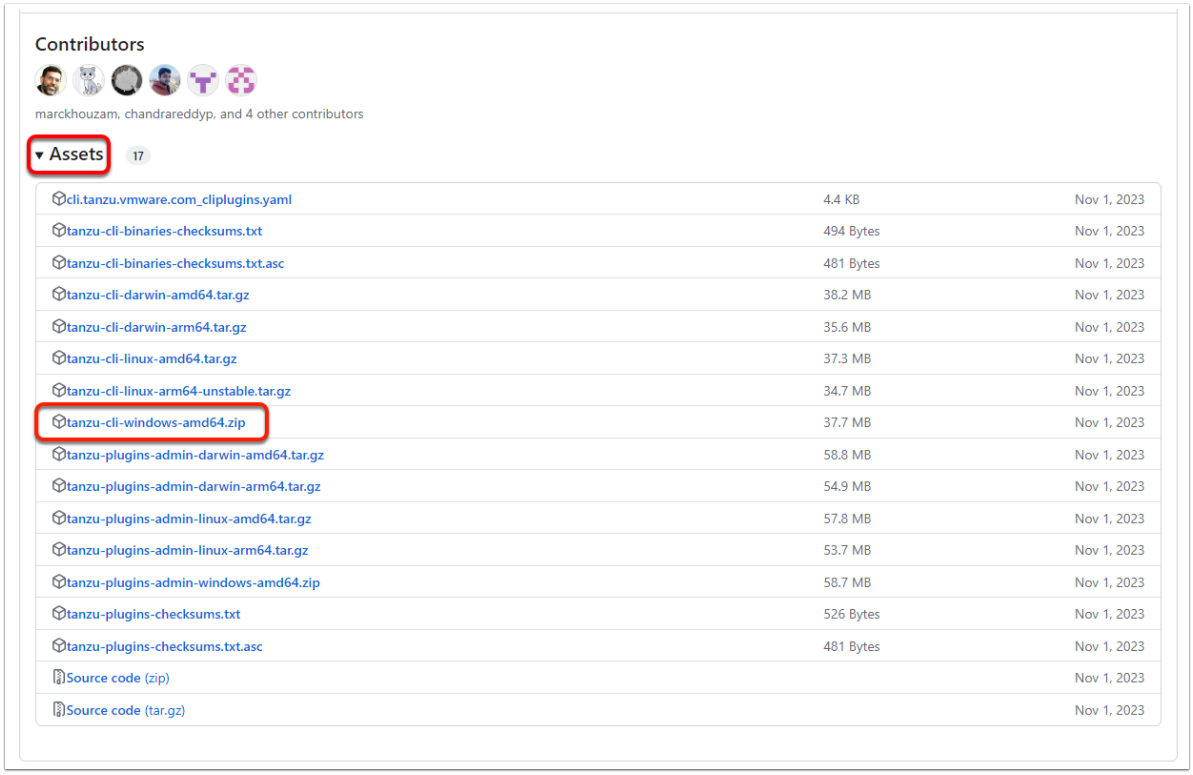
Our goal is to become able to deploy and manage Workload cluters with CLI, hence **the correct version matching our requirement is 1.1.0**

Install Tanzu Core CLI



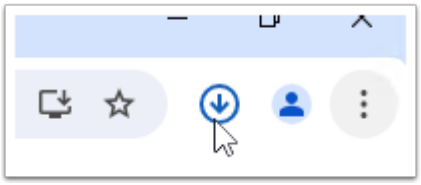
Get the Tanzu CLI core binary file from GitHub in Chrome open the URL https://github.com/vmware-tanzu/tanzu-cli/releases/tag/v1.1.0

Install Tanzu Core CLI



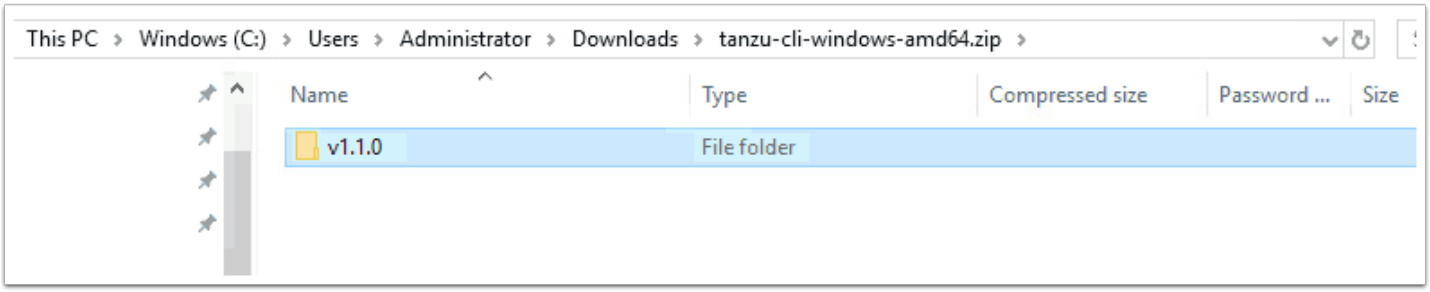
Scroll down to **Assets** and

Click on the the file: **tanzu-cli-windows-amd64.zip** to download the zip.



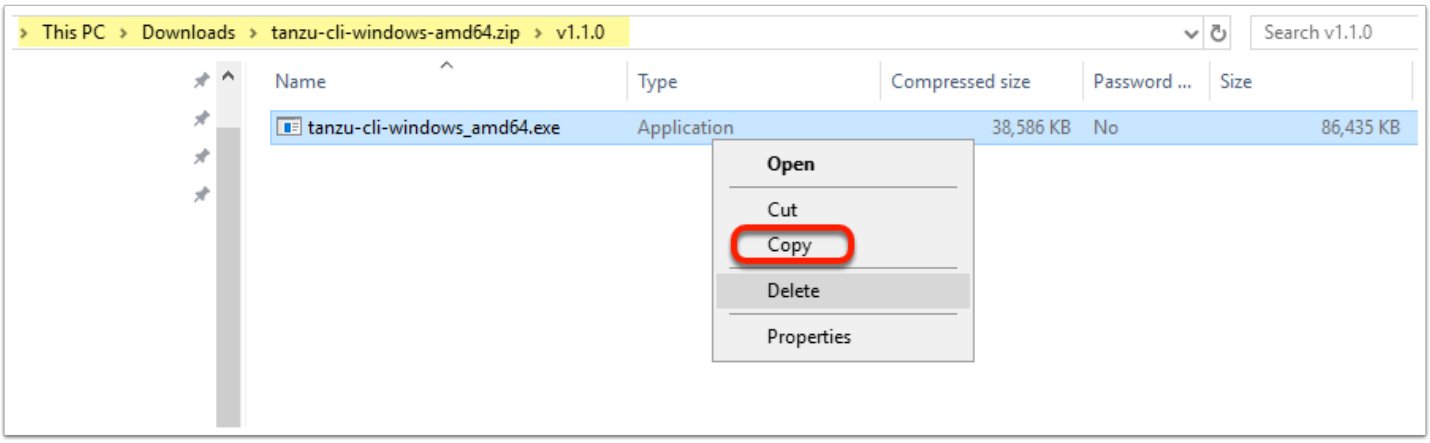
NOTE: A new browser tab may, open. Or you may not notice any change to the browser at all. Look at the right top corner, for the download icon to show the download icon to show the download has started.

Install Tanzu Core CLI



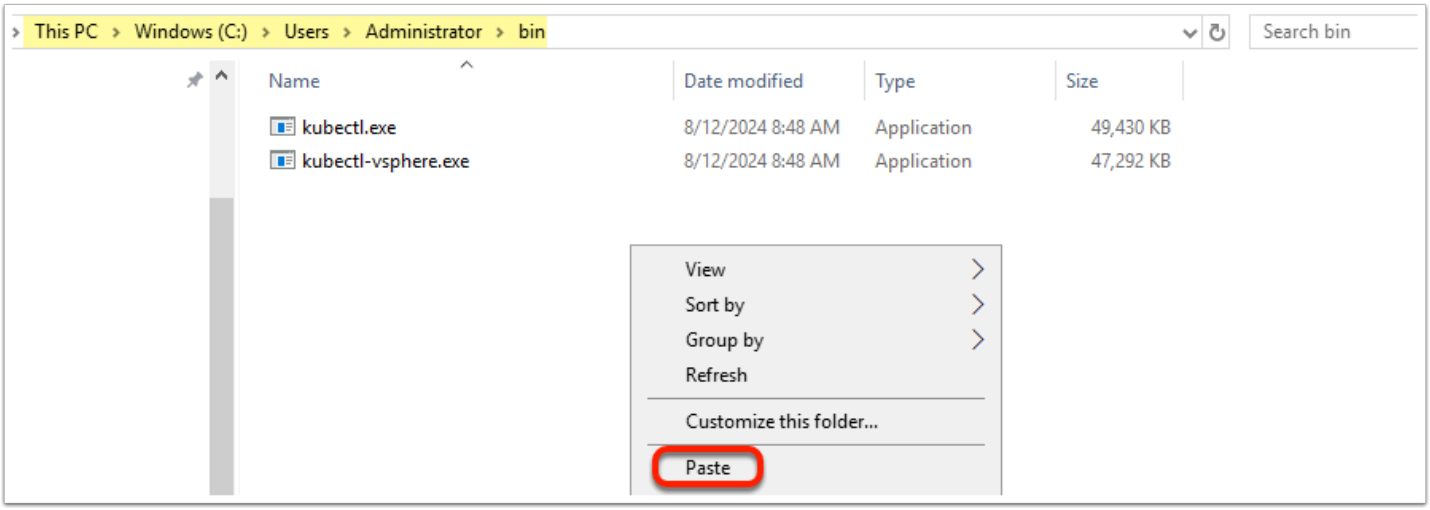
Find the zip archive in your Downloads folder. Double click the the archive, double click into the version subfolder and reach to **tanzu-cli-amd64.exe** binary.

Install Tanzu Core CLI



Copy the tabzu-cli exe file

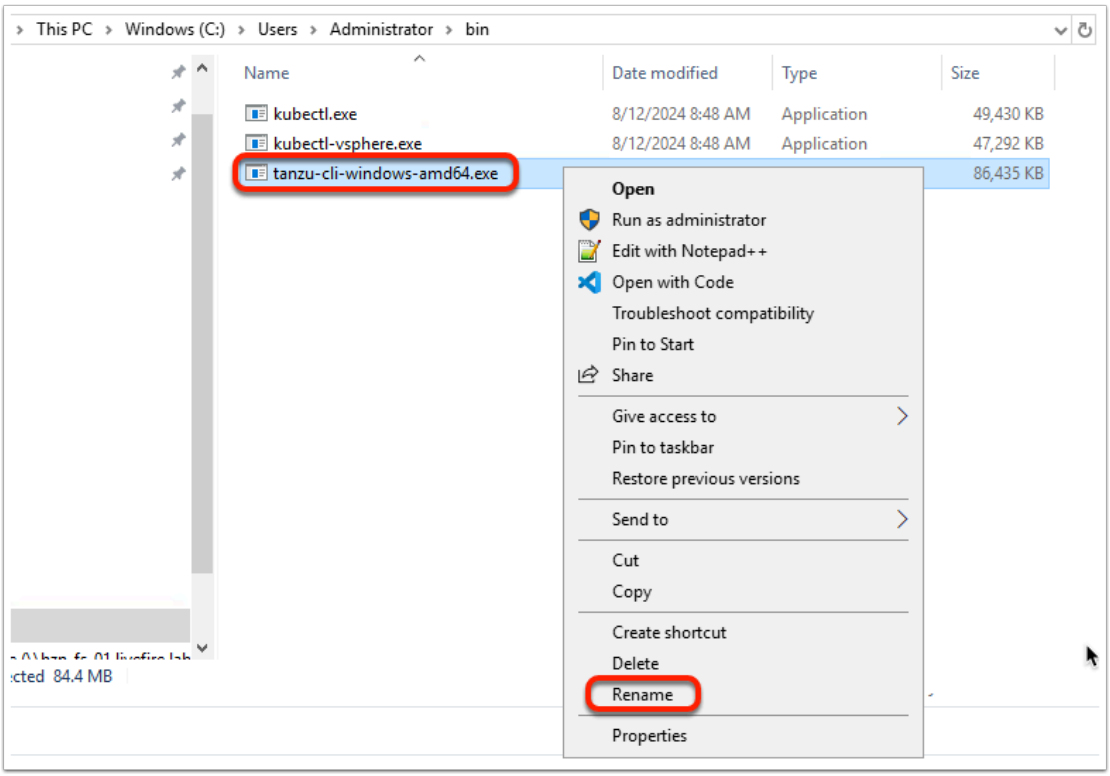
Install Tanzu Core CLI



Paste it into the **C:\Users\Administrator\bin** folder.

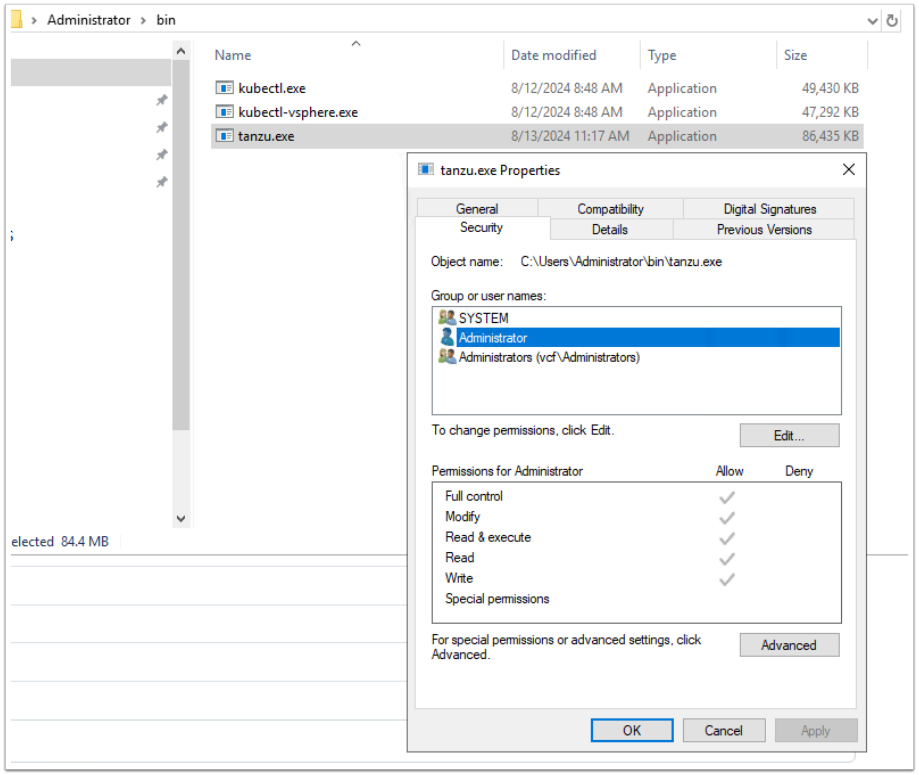
This folder is already in the System Path.

Install Tanzu Core CLI



Rename the extracted binary to **tanzu.exe**

Install Tanzu Core CLI



Right-click the **tanzu.exe** file, select **Properties -> Security**, and make sure that your user account has the **Full Control permission**

Install Tanzu Core CLI

Check that the correct version of the CLI is properly installed

```
tanzu version
```

This is a manual install we are using, choco, homebrew and other package managers work as well.

Install the Tanzu CLI plugins

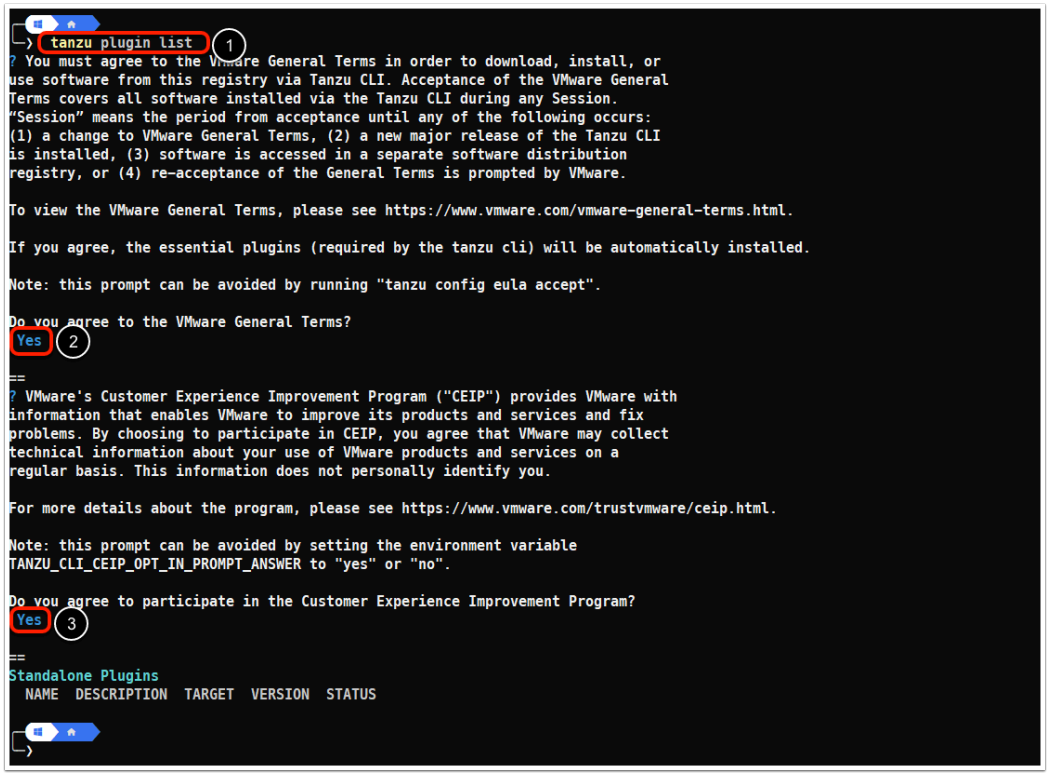
As stated at the beginning of the section, there are two flavors of plugins:

Context-scoped, that are automatically downloaded once you connect the core Tanzu CLI to the supported solution (for example Supervisor in vSphere)

Stand alone, that need to be downloaded additionally. The following section covers these.

The Tanzu CLI Standalone plugins are available through different **plugin groups**.

As a start, you will need to accept a few things:



1. List the available plugins

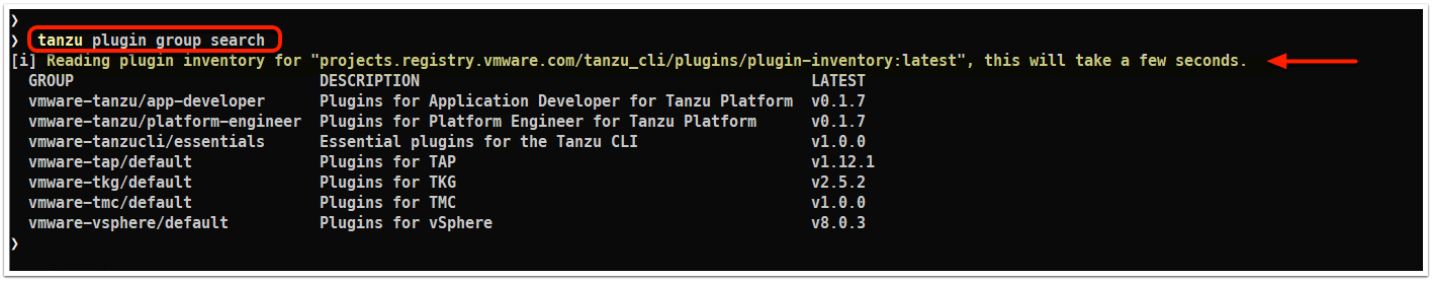
```
tanzu plugin list
```

Agree to:

2. General Terms
3. Participate in CEIP

You can find all the plugin groups:

Install the Tanzu CLI plugins



To find the list of groups available run

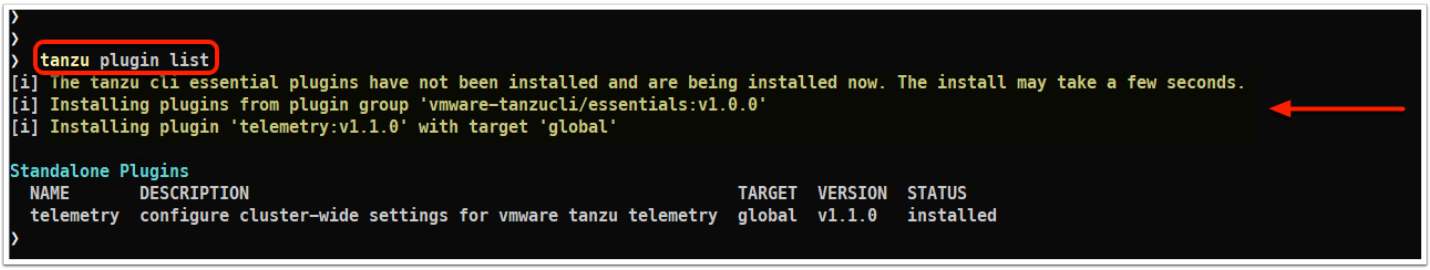
```
tanzu plugin group search
```

These are all the available Plugin groups, containing different **Stand Alone Tanzu CLI plugins**.

By default, on this iteration the **telemetry** plugin from the essentials group is getting installed.

Check it out:

Install the Tanzu CLI plugins



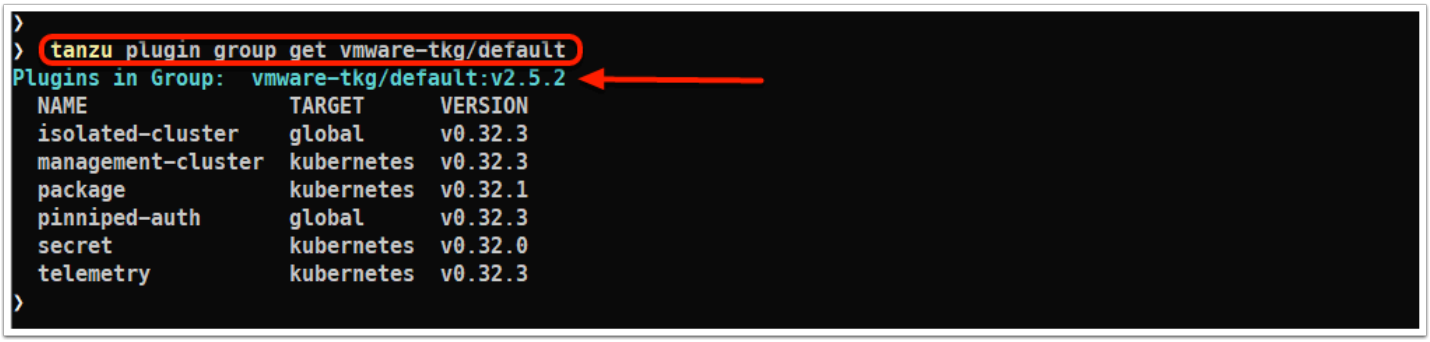
Now to install the essential plugin group run:

```
tanzu plugin list
```

This starts a check process for all the plugins available localy.

After a quick message you will see now that the **essential telemetry** plugin is installed

Install the Tanzu CLI plugins

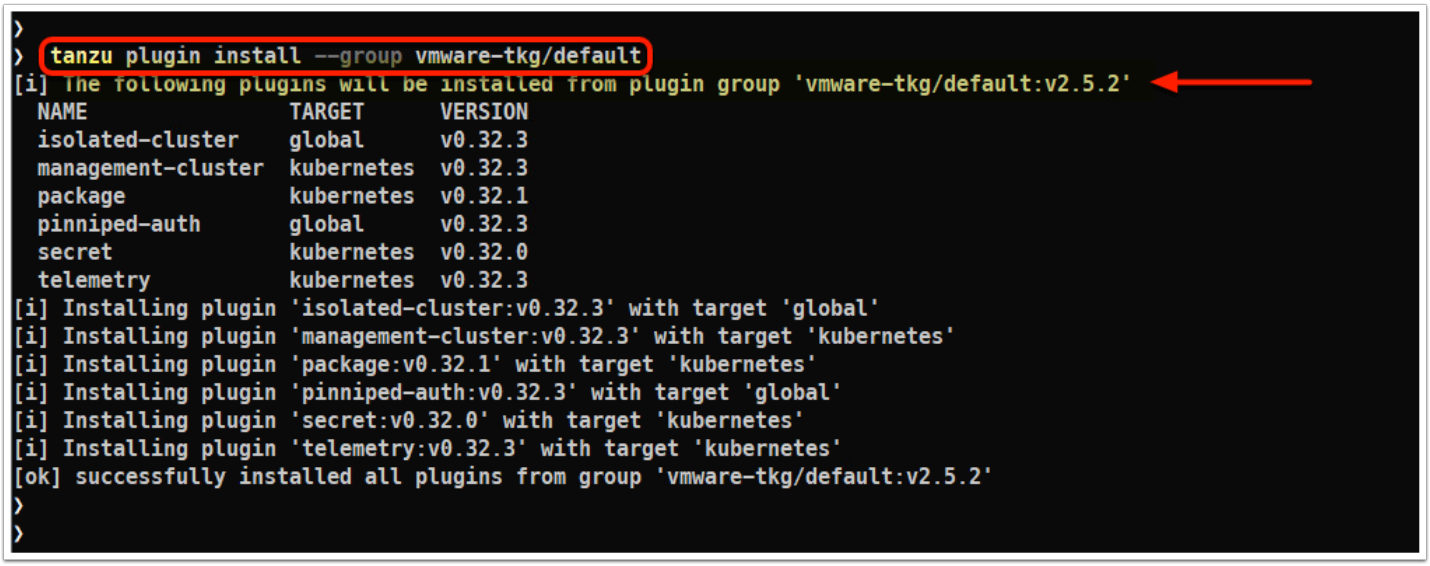


You'll need the TKG Plugin shortly, so go ahead and install them. First get the list confirming versions.

Now you can see what is the current available **vmware-tkg** group version.

```
tanzu plugin group get vmware-tkg/default
```

Install the Tanzu CLI plugins

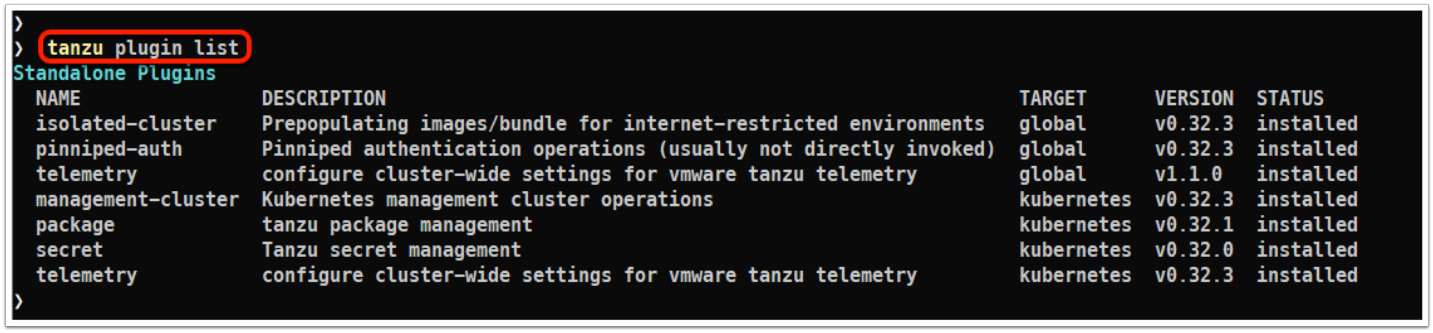


Now run the install

```
tanzu plugin install --group vmware-tkg/default
```

Verify the plugins installed successfully!

Install the Tanzu CLI plugins




```
tanzu plugin list
```

> ⚠ Please, note: These are all **Stand-Alone** type of **Tanzu CLI Plugins**!

After you have installed the Tanzu CLI core and standalone plugins for Tanzu Kubernetes Grid but before you have used it to connect to a management cluster, all context-specific CLI command groups, such as *tanzu cluster* and *tanzu kubernetes-release*, are unavailable and not included in Tanzu CLI --help output.

The Tanzu CLI installs context-scoped plugins automatically when you connect to a management cluster!

Log in to Supervisor via CLI

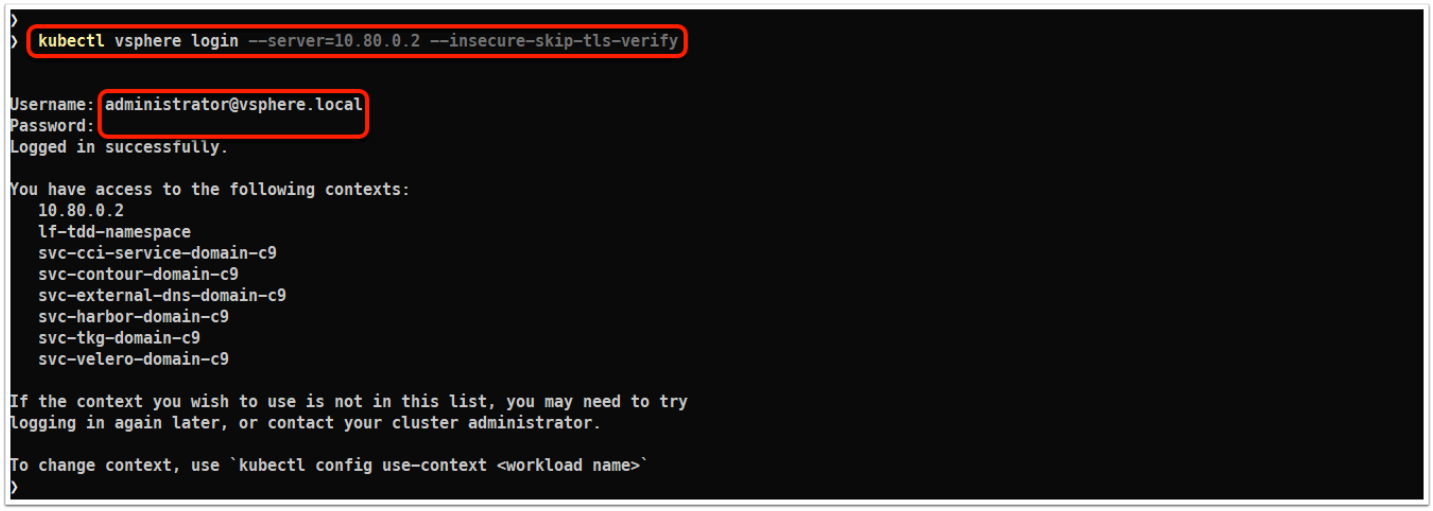
Task description and objectives

As part of your activities, you have to log in to the Supervisor using CLI and the local administrator account.

Log in to Supervisor using CLI kubectl

1. You can continue working in your **PowerShell prompt** and use it as a Terminal window.

2. Create a local **kubeconfig** file by logging in to the Supervisor with the **vSphere plug-in** for kubectl.



Connect to the Tanzu supervisor

```
kubectl vsphere login --server=10.80.0.2 -u administrator@vsphere.local --insecure-skip-tls-verify
```

In this command, and in all commands in later activities in which interaction with the Supervisor is required, you must use the correct Supervisor IP address - `10.80.0.2` - that you recorded in an earlier task.

As the vCenter root certificate is not installed on your windows desktop machine, the **--insecure-skip-tls-verify** flag is required when you log in to the Supervisor

password:

```
VMware123!VMware123!
```

You are creating a local **kubeconfig** file by logging in to the Supervisor with the vSphere plug-in for **kubectl**.

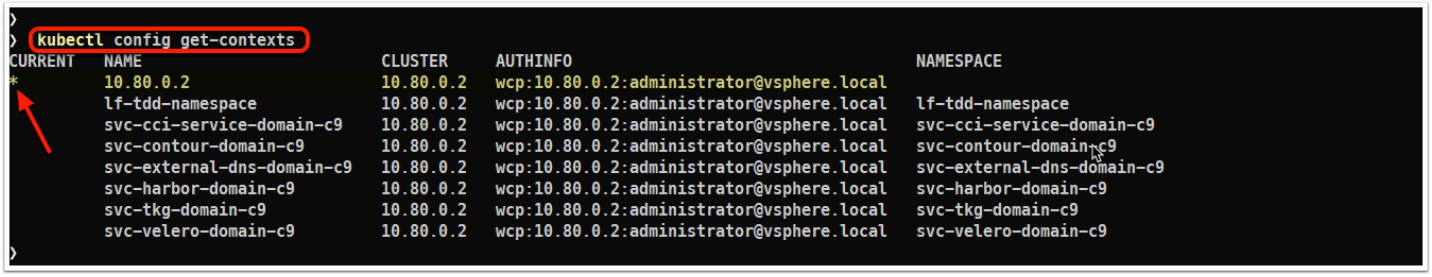
Log in to Supervisor using CLI kubectl


```
dir .kube
```

Work with contexts in your Supervisor

Once you've logged in your Supervisor, you are informed which contexts you have access to.

But you can check this anytime and get a bit more details:

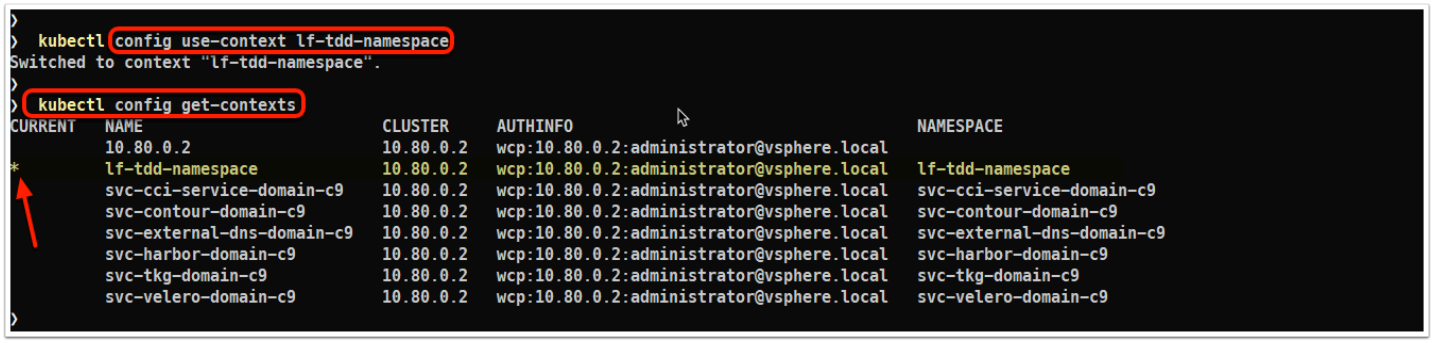



```
kubectl config get-contexts
```

The one that is active is marked as current with (*) in the CLI

You can switch between contexts

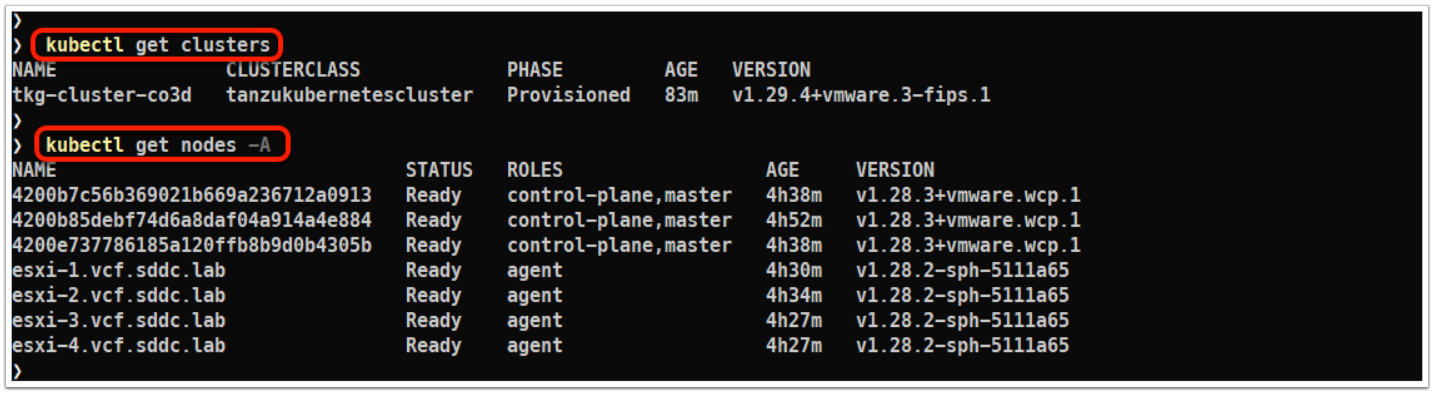
Work with contexts in your Supervisor




```
kubectl config use-context lf-tdd-namespace
```

You can also check the clusters in the current context, if any

Work with contexts in your Supervisor

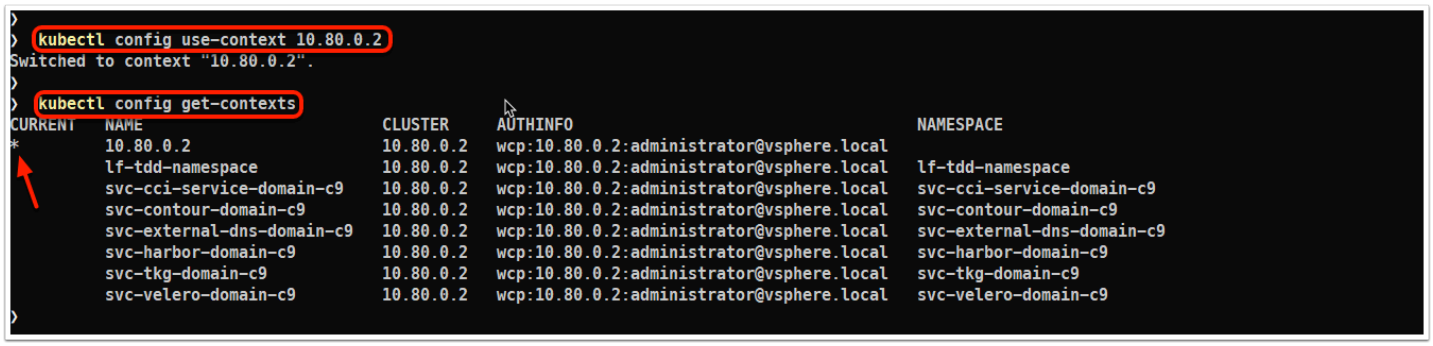



```
kubectl get clusters
```

```
kubectl get nodes -A
```

Connect the Tanzu CLI to the Supervisor

Make sure your current content points back to the Supervisor

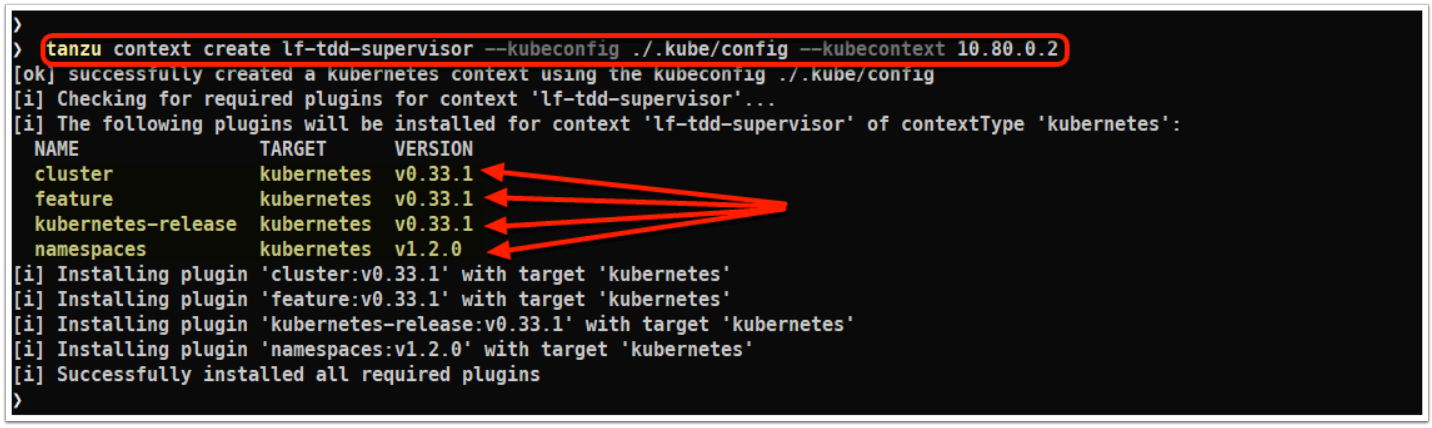



```
kubectl config use-context 10.80.0.2
```

```
kubectl config get-contexts
```

> ⚠ The Tanzu CLI installs **context-scoped plugins** automatically when you connect to a **management cluster**!

Connect the Tanzu CLI to the Supervisor




```
tanzu context create lf-tdd-supervisor --kubeconfig ./.kube/config --kubecontext 10.80.
0.2
```

Let's check what happened - Tanzu CLI should now have a configured context!

Connect the Tanzu CLI to the Supervisor

```
❯
❯ tanzu context list
  NAME               ISACTIVE  TYPE        ENDPOINT  KUBECONFIGPATH  KUBECONTEXT
  lf-tdd-supervisor  true      kubernetes            ./.kube/config  10.80.0.2
❯
```

```
tanzu context list
```

From now on, you can refer to your Supervisor in the way you registered it within the context:

Connect the Tanzu CLI to the Supervisor

```
❯
❯ tanzu context use
? Select a context lf-tdd-supervisor  (./.kube/config:10.80.0.2)
[i] Successfully activated context 'lf-tdd-supervisor'
[i] Checking for required plugins for context 'lf-tdd-supervisor'...
[i] All required plugins are already installed and up-to-date
❯
```

```
tanzu context use
```

Please select your Supervisor **lf-tdd-supervisor** when prompted

Let's inspect the overall plugins in place:

Connect the Tanzu CLI to the Supervisor

```
❯
❯ tanzu plugin list
Standalone Plugins
  NAME                DESCRIPTION                                                    TARGET      VERSION  STATUS
  isolated-cluster    Prepopulating images/bundle for internet-restricted environments  global      v0.32.3  installed
  pinniped-auth       Pinniped authentication operations (usually not directly invoked)  global      v0.32.3  installed
  telemetry           configure cluster-wide settings for vmware tanzu telemetry     global      v1.1.0   installed
  management-cluster  Kubernetes management cluster operations                       kubernetes  v0.32.3  installed
  package             tanzu package management                                       kubernetes  v0.32.1  installed
  secret              Tanzu secret management                                        kubernetes  v0.32.0  installed
  telemetry           configure cluster-wide settings for vmware tanzu telemetry     kubernetes  v0.32.3  installed

Plugins from Context:  lf-tdd-supervisor
  NAME                DESCRIPTION                                          TARGET      VERSION  STATUS
  cluster             Kubernetes cluster operations                        kubernetes  v0.33.1  installed
  feature             Operate on features and featuregates                 kubernetes  v0.33.1  installed
  kubernetes-release  Kubernetes release operations                        kubernetes  v0.33.1  installed
  namespaces          Discover vSphere Supervisor namespaces you have access to  kubernetes  v1.2.0   installed
❯
```

```
tanzu plugin list
```

Now you should see the list of:

- The standalone CLI plugins (installed as standalone in the previous task)
- Supervisor specific plugins (just installed as context-scoped) as well as

Be aware, you can always check if newer versions of your installed plugins are available

Connect the Tanzu CLI to the Supervisor


```
tanzu plugin sync
```

Make a final check if you need to update any of the installed plugins.

You are now ready to create Workload clusters with CLI

Deploy a TKG Workload Cluster via CLI

Task description and objectives

With Tanzu Kubernetes Grid on Supervisor, you can provision different styles of workload clusters, includingTanzu Kubernetesclusters with well-defined defaults and Cluster Class clusters with broad definition options.

With Tanzu Kubernetes Grid on Supervisor, you can provision different styles of workload clusters, including Tanzu Kubernetes clusters with well-defined defaults and Cluster Class clusters with broad definition options.

With this lab you will provision and manage the life cycle of TKG clusters on using the Supervisor in the Sphere IaaS control plane via the Tanzu CLI.

TKG Services come with a default ClusterClass definition named `tanzukubernetescluster`. It provides the template for cluster creation using the `v1beta` API.

The `tanzukubernetescluster` ClusterClass is available in all user namespaces. To create a cluster based on this ClusterClass, you need to reference it in the Cluster specification.

IMPORTANT: This exercise is being done with administrator@vsphere.local permissions set over the lf-tdd-namespace

The default `tanzukubernetescluster` ClusterClass is immutable. It may be updated with each release of the TKG Service.

To view the default t`anzukubernetescluster` ClusterClass that ships with your TKG Service instance, complete the following steps:

Deploy a TKG Workload Cluster

Aligned with the previous section, Make sure you are logged in your **Supervisor**

Navigate to your working Namespace **lf-tdd-namespace** where you will provision an additional **workload cluster**

```
❯
❯ kubectl config get-contexts
CURRENT   NAME                          CLUSTER     AUTHINFO                                       NAMESPACE
*         10.80.0.2                     10.80.0.2   wcp:10.80.0.2:administrator@vsphere.local
          lf-tdd-namespace              10.80.0.2   wcp:10.80.0.2:administrator@vsphere.local      lf-tdd-namespace
          svc-cci-service-domain-c9     10.80.0.2   wcp:10.80.0.2:administrator@vsphere.local      svc-cci-service-domain-c9
          svc-contour-domain-c9         10.80.0.2   wcp:10.80.0.2:administrator@vsphere.local      svc-contour-domain-c9
          svc-external-dns-domain-c9    10.80.0.2   wcp:10.80.0.2:administrator@vsphere.local      svc-external-dns-domain-c9
          svc-harbor-domain-c9          10.80.0.2   wcp:10.80.0.2:administrator@vsphere.local      svc-harbor-domain-c9
          svc-tkg-domain-c9             10.80.0.2   wcp:10.80.0.2:administrator@vsphere.local      svc-tkg-domain-c9
          svc-velero-domain-c9          10.80.0.2   wcp:10.80.0.2:administrator@vsphere.local      svc-velero-domain-c9
❯
❯  kubectl config use-context lf-tdd-namespace
Switched to context "lf-tdd-namespace".
❯
```

```
kubectl config get-contexts
```

```
kubectl config use-context lf-tdd-namespace
```

Get the default **ClusterClass** and output this to a file named **tkc-dcc.yaml**.

You can **Review** the file.

```
❯
❯ kubectl get clusterclass tanzukubernetescluster -o yaml > tkc-dcc.yaml
❯
❯ code .\tkc-dcc.yaml
❯
```

C: > Users > Administrator > tkc-dcc.yaml > apiVersion

io.x-k8s.cluster.v1beta1.ClusterClass (v1beta1@clusterclass.json)

```yaml
apiVersion: cluster.x-k8s.io/v1beta1
kind: ClusterClass
metadata:
  annotations:
    run.tanzu.vmware.com/resolve-tkr: ""
  creationTimestamp: "2024-08-14T11:21:29Z"
  generation: 1
  name: tanzukubernetescluster
  namespace: lf-tdd-namespace
  resourceVersion: "152043"
  uid: 3bb14c04-5230-4add-a34a-04389b9d17ef
```

Note: The screenshot of the **tkc-tdd.yaml** file is truncated, showing only the starting ten rows)

```
kubectl get clusterclass tanzukubernetescluster -o yaml > tkc-dcc.yaml
```

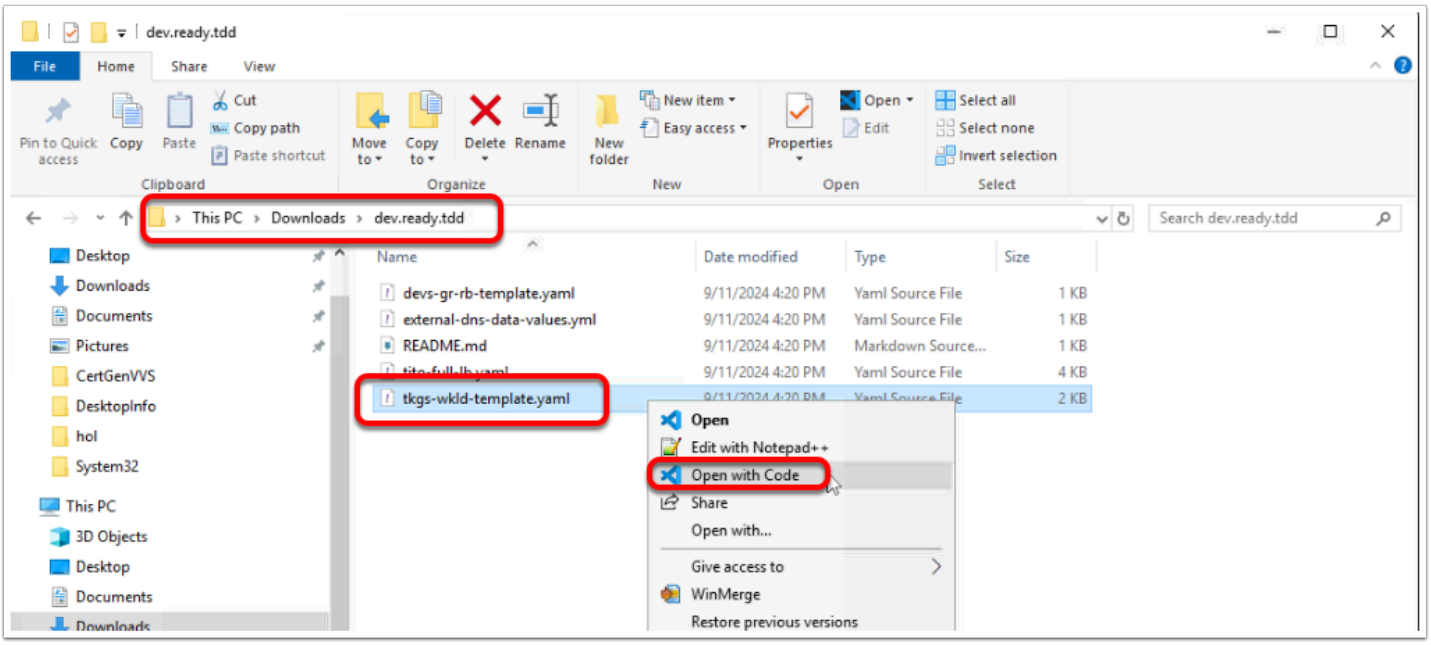
```
code tkc-dcc.yaml
```

TKG Workload Cluster configuration file

You can create a Tanzu Kubernetes Grid **workload cluster configuration file** using the management cluster configuration file as **template**. We have provided a github repository with the files you need. You just need to the clone the repo locally if you did not do it earlier. Assuming you did the git clone ealier then ignore this next step

In a terminal run the command

```
git clone https://github.com/Livefire-Labs/dev.ready.tdd.git C:\Users\Administrator\
Downloads\dev.ready.tdd
```



Open the file *C:\Users\Administrator\Downloads\dev.ready.tdd\tkgs-wkld-template.yaml in* ***VSCode***

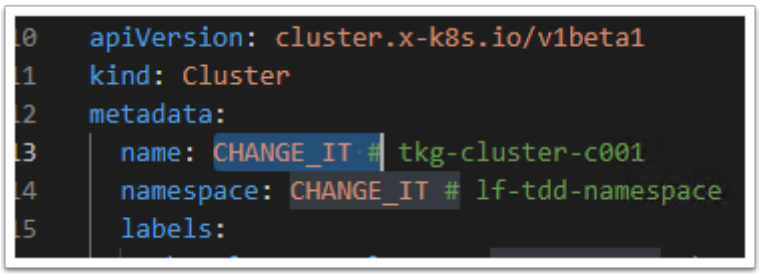
This template references a **Custom Cluster Configuration Based on the Default ClusterClass**

```
C: > Users > Administrator > Downloads > dev.ready.tdd > tkgs-wkld-template.yaml > ...
    io.x-k8s.cluster.v1beta1.Cluster (v1beta1@cluster.json)
#  ------------------------------------------------------------------------------
#                      Livefire TDD DevReady 2024
#
#            You must have enabled your Supervisor in vSphere
#              in order to provision this TKG workload cluster
#
#                      by Konstantin Nikolov, Livefire Team
#  ------------------------------------------------------------------------------

apiVersion: cluster.x-k8s.io/v1beta1
kind: Cluster
metadata:
  name: CHANGE_IT # tkg-cluster-c001
  namespace: CHANGE_IT # lf-tdd-namespace
  labels:
    tkg-cluster-selector: CHANGE_IT # tkg-cluster-c001
spec:
  clusterNetwork:
    pods:
      cidrBlocks:
        - CHANGE_IT # 192.168.156.0/20
    services:
      cidrBlocks:
        - CHANGE_IT # 10.96.0.0/12
    serviceDomain: CHANGE_IT # cluster.local
  topology:
    class: CHANGE_IT # tanzukubernetescluster
    version: CHANGE_IT # v1.29.4---vmware.3-fips.1-tkg.1
    variables:
      - name: storageClasses
        value:
          - CHANGE_IT # vsan-tanzu-storage
      - name: defaultStorageClass
        value: CHANGE_IT # vsan-tanzu-storage
      - name: vmClass
        value: CHANGE_IT # best-effort-small
      - name: storageClass
        value: CHANGE_IT # vsan-tanzu-storage
    controlPlane:
      replicas: CHANGE_IT # 1
      metadata:
        annotations:
          run.tanzu.vmware.com/resolve-os-image: CHANGE_IT # os-name=ubuntu
    workers:
      machineDeployments:
        - class: node-pool
          name: CHANGE_IT # tkg-cluster-nodepool-c001
          replicas: CHANGE_IT # 1
          metadata:
            annotations:
              run.tanzu.vmware.com/resolve-os-image: CHANGE_IT # os-name=ubuntu

```

Use the **SAVE AS** in VSCode and give the template manifest file a custom name: **C:\Downloads\tkgs-wkld-c001.yaml.** Livefire did the work for you, if you look at the tkgs-wkld-template.yaml file, or the copy you saved, we have marked the lines you will need to modify to successfully deploy a TKGs cluster to vSphere. Each line that needs to be changed has the Value "**CHANGE_IT**" followed by a comment with the value to use for the lab.

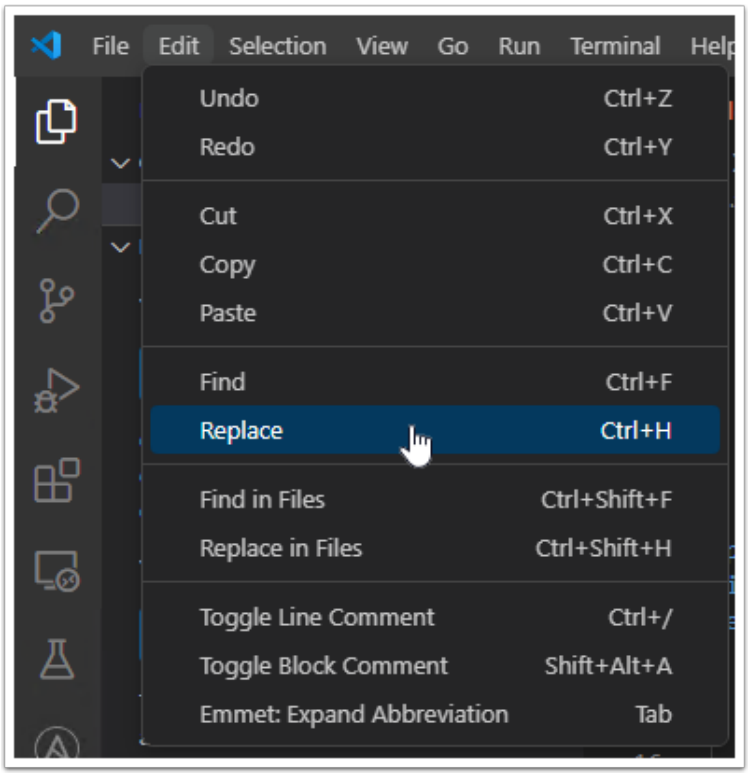
Carefully remove all the **CHANGE_IT #** text on each appropriate line by hand

OR

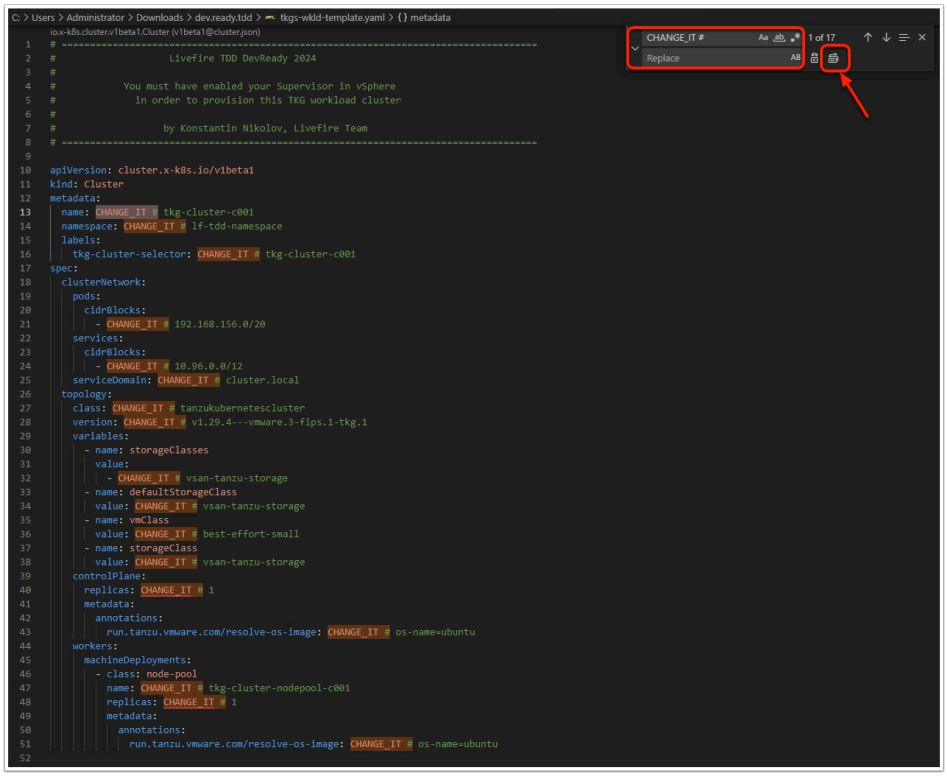


```
10  apiVersion: cluster.x-k8s.io/v1beta1
11  kind: Cluster
12  metadata:
13    name: CHANGE_IT # tkg-cluster-c001
14    namespace: CHANGE_IT # lf-tdd-namespace
15    labels:
```

In VSCode highlight one of the **CHANGE_IT #** entries. You will notice that VSCode recognizes that there are more text strings that match what you have selected.



With the **CHANGE_IT #** string still highlighted in VSCode, select **Edit** --> **Replace**



VSCode should then highlight ALL of the strings to replace. Leave the bottom value empty

Click the **Replace All** button

VSCode should make your configuration file should look exactly as the screenshot above. If not you can always undo and try again.

Remember to Save the file!

You are now ready to start provisioning your new workload cluster with CLI.

Create a TKG Workload Cluster

The Kubernetes Cluster API is a suite of tools which provide for the declarative provisioning, upgrading, and operating of Kubernetes clusters. ClusterClass is an evolution of the Cluster API that lets you define templates for managing the life cycle of sets of clusters. TKG Service supports ClusterClass using the `v1beta1` API.

ClusterClass API is the newest Cluster API that should be used to deploy all new Kubernetes clusters using built in ClusterClass definitions.

TanzuKubernetesCluster API is provided for backward compatibility and a legacy offering, which will be deprecated in a future release.

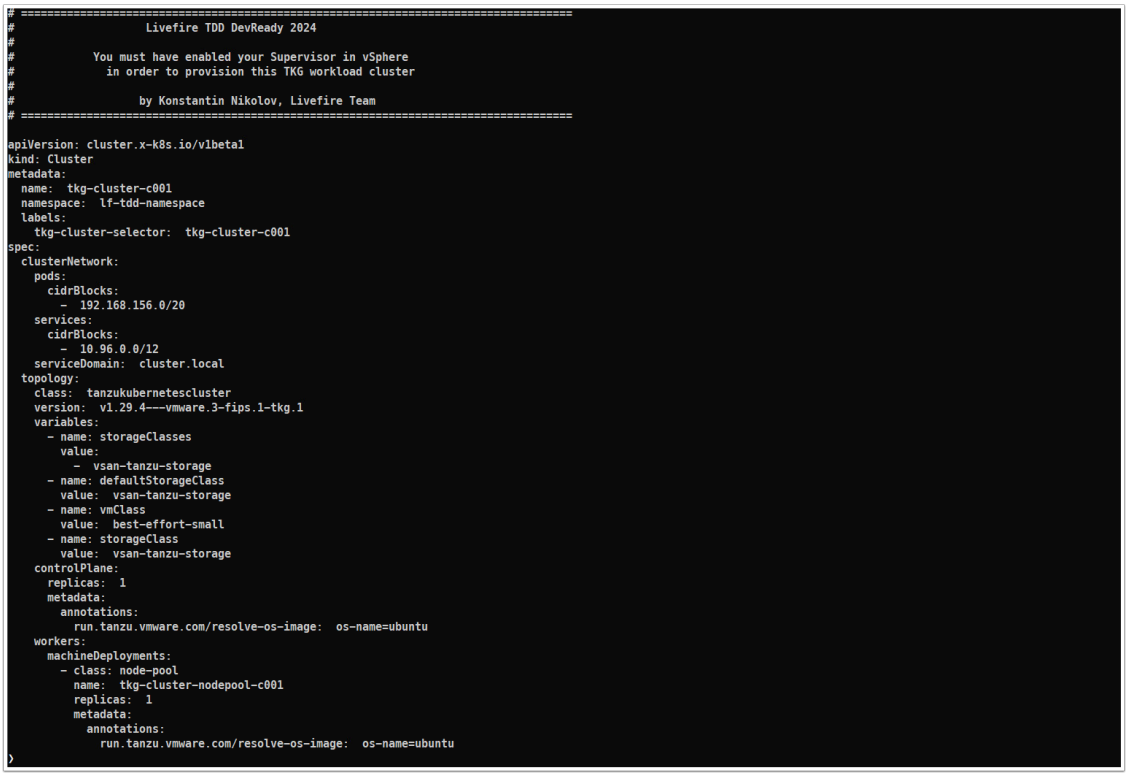
Next step is to provision a workload cluster, using the above described ClusterClass, but also leverage the updated ClusterClass API. This is already configured in your configuration file from the previous step.

Let's check you have the correct file in place:


```
ls .\Downloads\tkgs-wkld-c001.yaml
```

A good idea is to perform an initial dry run before the actual cluster creation, which queries vCenter to ensure that the vSphere resources exist.

Create a TKG Workload Cluster



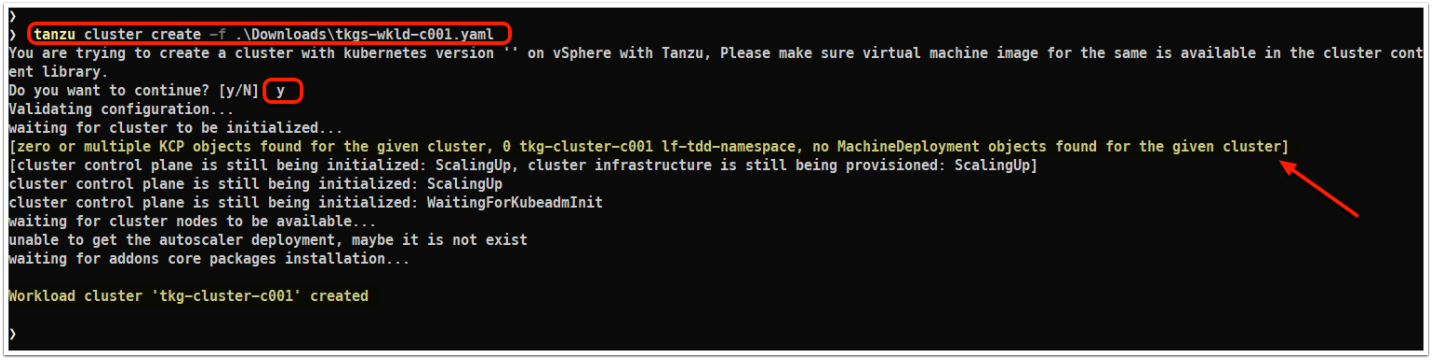
Run the command

```
tanzu cluster create -f .\Downloads\tkgs-wkld-c001.yaml --dry-run
```

If the output shows **exit status 1**, you must review the configuration parameters in the **tkgs-wkld-c001.yaml** configuration file.

If dry run finished okay, go ahead and create the cluster:

Create a TKG Workload Cluster



Run the same command as before, but without the --dry-run

```
tanzu cluster create -f .\Downloads\tkgs-wkld-c001.yaml
```

When prompted type `y` to continue!

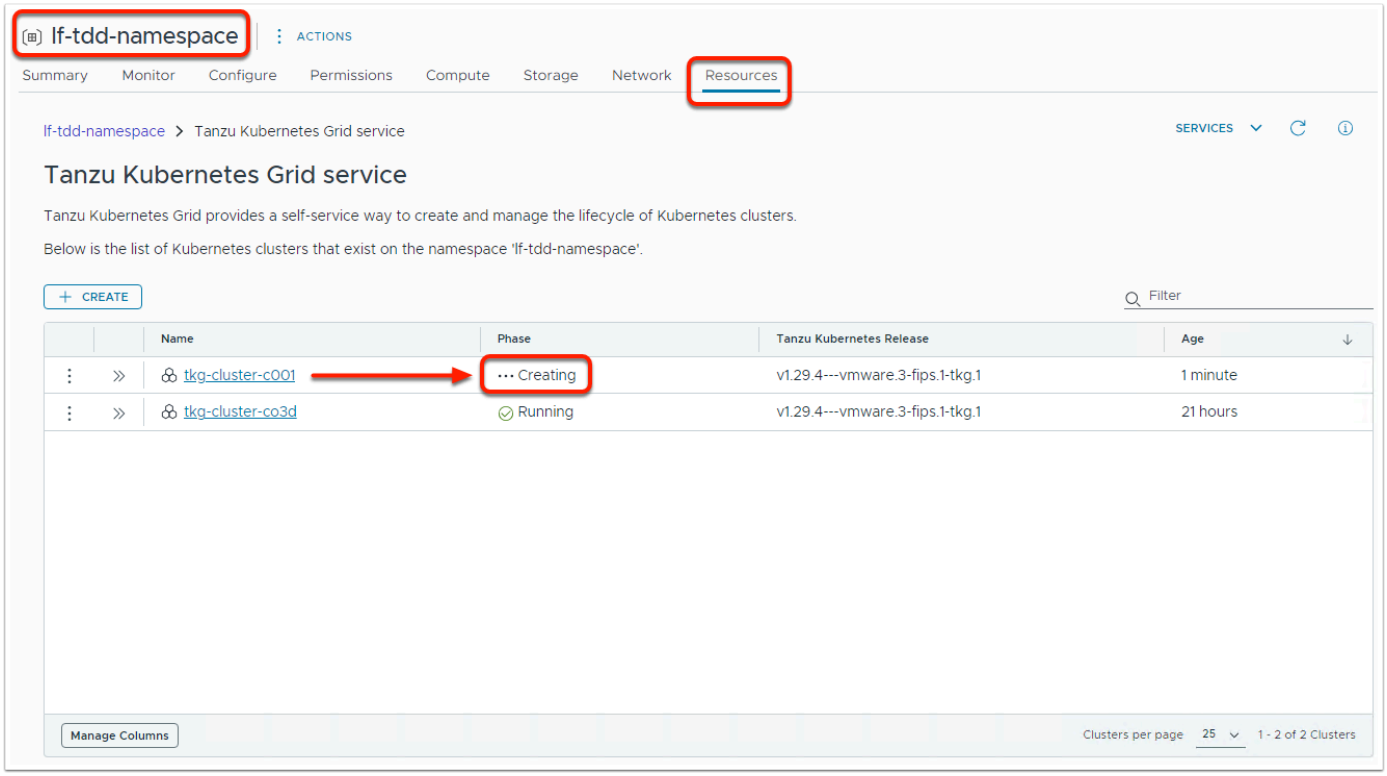
The Tanzu Kubernetes Grid cluster takes **approximately 25 minutes** to deploy.

Wait for the **Workload cluster 'tkg-cluster-c001' created** message to appear.

While your new workload cluster is being provisioned, you can track some visuals in the vSphere client UI

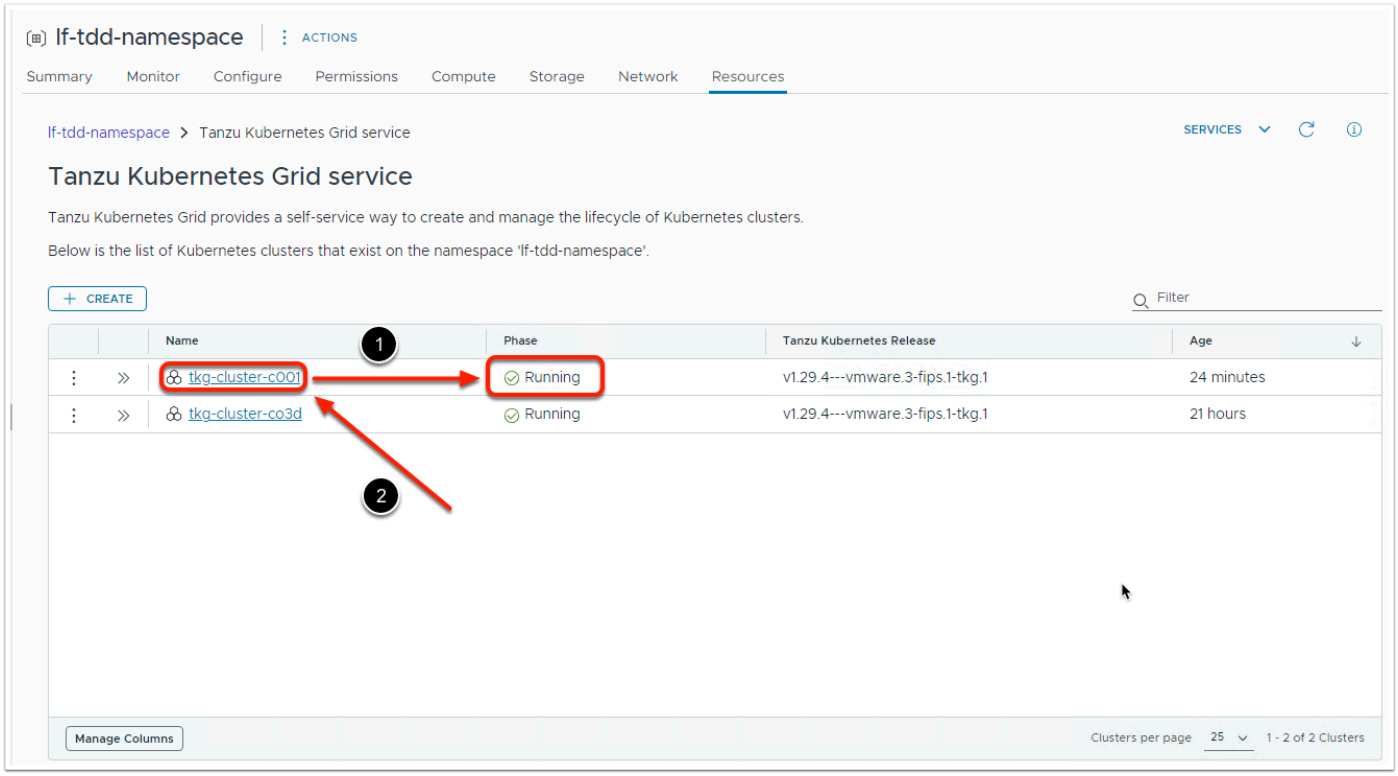
In your Workload Management --> Namespaces --> lf-tdd-namespace pane, you will see the new cluster changing phases:

Create a TKG Workload Cluster



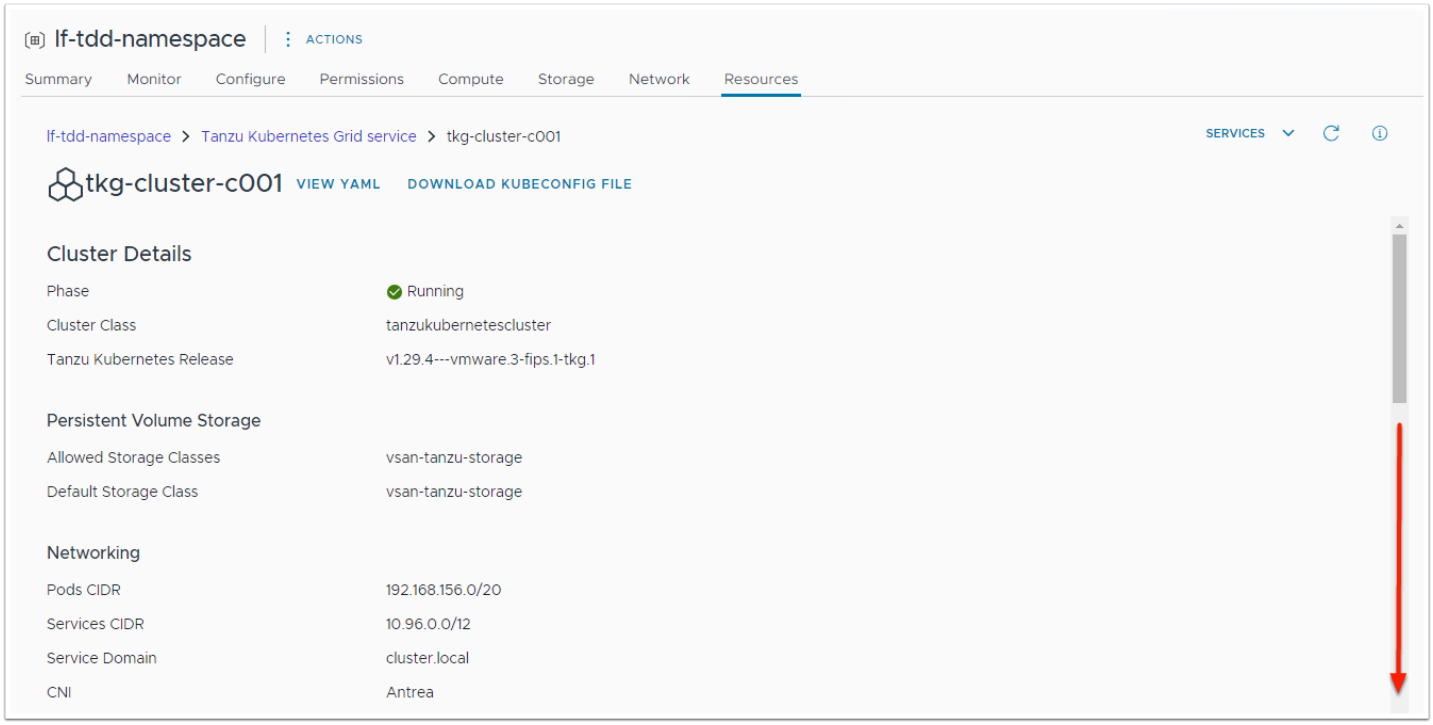
- from ... Creating

Create a TKG Workload Cluster



... to Running within 25 mins or so.

Create a TKG Workload Cluster

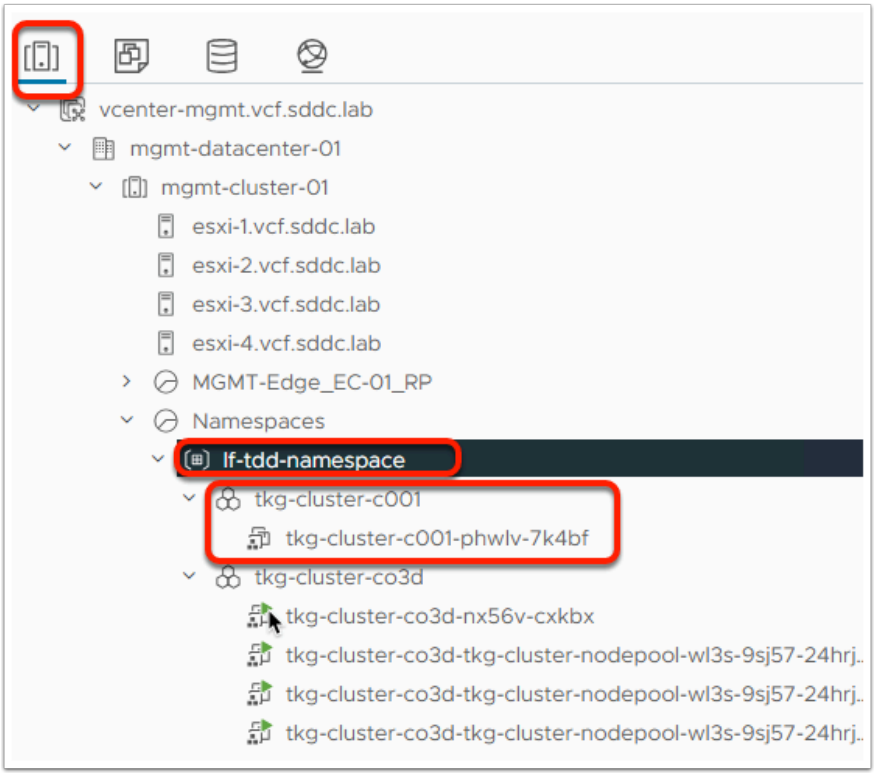


If you click on the **tkg-cluster-c001** name, you will get visual of all the **Cluster Details**

Scroll down the slide bar in the UI

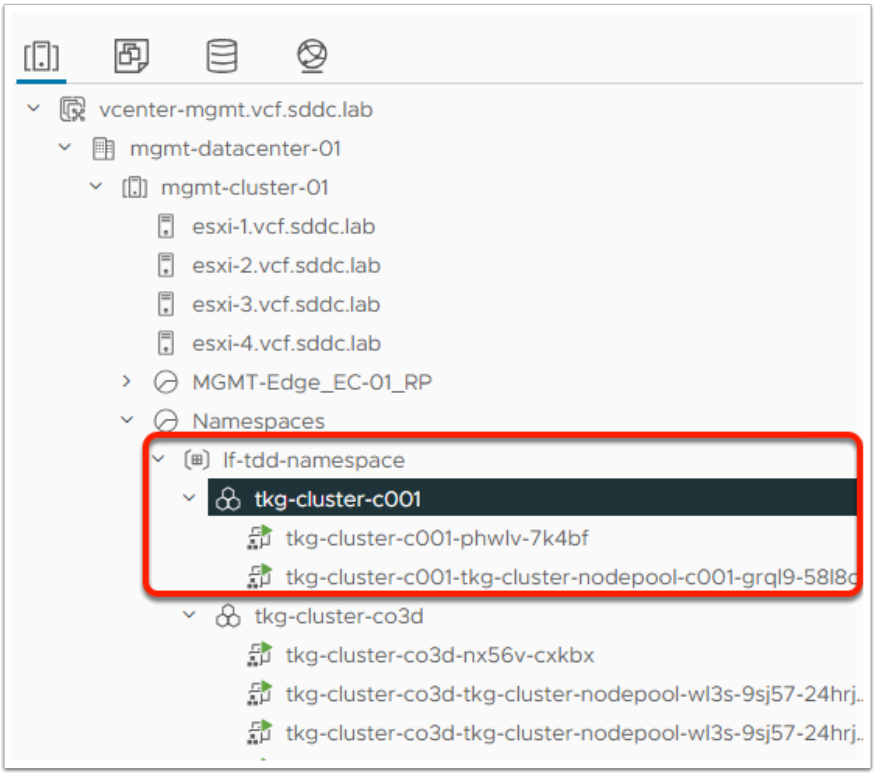
Also, you can track the new cluster appearance from **Inventory** --> **VMs and Clusters** --> under **lf-tdd-namespace**

Create a TKG Workload Cluster



... starting with the Control Plane provisioning

Create a TKG Workload Cluster



... and ending with the Workload nodes added

All seems good, so let's get back in the PS Terminal and check the cluster(s) residing in the lf-tdd-namespace

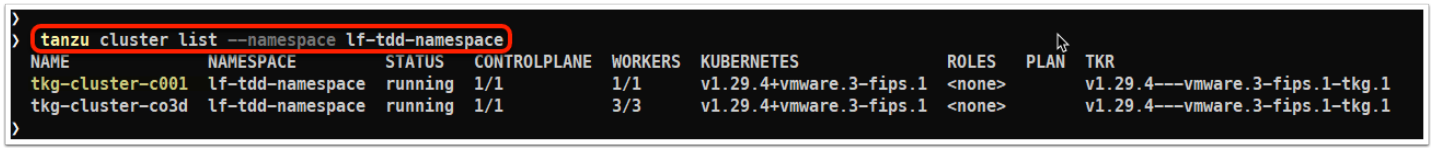
Create a TKG Workload Cluster

```
❯
❯ kubectl get clusters
NAME              CLUSTERCLASS              PHASE         AGE     VERSION
tkg-cluster-c001  tanzukubernetescluster    Provisioned   3h22m   v1.29.4+vmware.3-fips.1
tkg-cluster-co3d  tanzukubernetescluster    Provisioned   24h     v1.29.4+vmware.3-fips.1
❯
```

```
kubectl get clusters
```

You can do the same, but getting a bit more info with the tanzu cli

Create a TKG Workload Cluster




```
tanzu cluster list --namespace lf-tdd-namespace
```

Okay, you are now ready to go to the next section!

Manage a TKG Workload Cluster

Task description and objectives

Your task is to inspect the health of the deployed Tanzu Kubernetes Grid workload cluster you recently created.

The workload cluster currently has one worker node, which is not sufficient for the purpose of the cluster. You must also increase the number of worker nodes so that new applications can be deployed. The number of worker nodes must be increased to two.

As a part of this lab, you must allow privileged workload deployment in the Tanzu Kubernetes Grid workload cluster

IMPORTANT: This exercise is being done with administrator@vsphere.local permissions set over the lf-tdd-namespace

Make sure you are logged in your **Supervisor**

Manage a TKG Workload Cluster

Aligned with the previous section, make sure you are connected to your Supervisor!

```
❯
❯ kubectl config get-contexts
CURRENT   NAME                          CLUSTER     AUTHINFO                                      NAMESPACE
*         10.80.0.2                     10.80.0.2   wcp:10.80.0.2:administrator@vsphere.local
          lf-tdd-namespace              10.80.0.2   wcp:10.80.0.2:administrator@vsphere.local     lf-tdd-namespace
          svc-cci-service-domain-c9     10.80.0.2   wcp:10.80.0.2:administrator@vsphere.local     svc-cci-service-domain-c9
          svc-contour-domain-c9         10.80.0.2   wcp:10.80.0.2:administrator@vsphere.local     svc-contour-domain-c9
          svc-external-dns-domain-c9    10.80.0.2   wcp:10.80.0.2:administrator@vsphere.local     svc-external-dns-domain-c9
          svc-harbor-domain-c9          10.80.0.2   wcp:10.80.0.2:administrator@vsphere.local     svc-harbor-domain-c9
          svc-tkg-domain-c9             10.80.0.2   wcp:10.80.0.2:administrator@vsphere.local     svc-tkg-domain-c9
          svc-velero-domain-c9          10.80.0.2   wcp:10.80.0.2:administrator@vsphere.local     svc-velero-domain-c9
❯
❯ kubectl config use-context lf-tdd-namespace
Switched to context "lf-tdd-namespace".
❯
```

Navigate to your working Namespace **lf-tdd-namespace** where you will provision an additional **workload cluster**

```
kubectl config get-contexts
```

```
kubectl config use-context lf-tdd-namespace
```

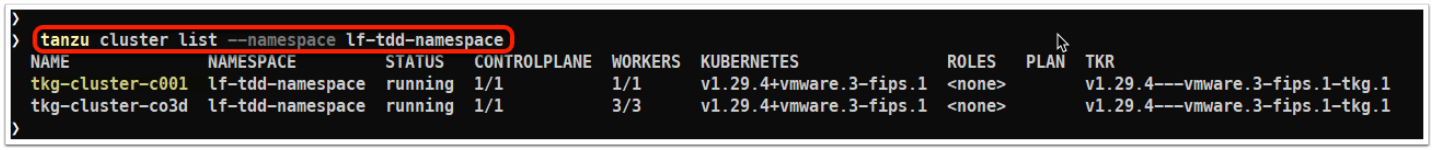
Make sure to log in to your Supervisor context with Tanzu CLI as well

Manage a TKG Workload Cluster


```
tanzu context use lf-tdd-supervisor
```

Manage a TKG Workload Cluster

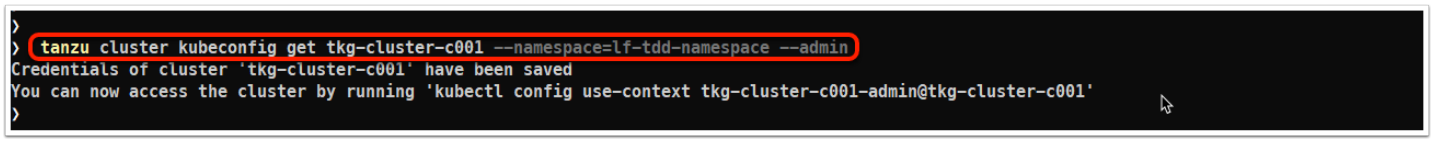
Once again, assure your workload clusters are in place:




```
tanzu cluster list --namespace lf-tdd-namespace
```

Manage a TKG Workload Cluster

Retrieve the administrator **kubeconfig** file for the **tkg-cluster-c001** workload cluster.




```
tanzu cluster kubeconfig get tkg-cluster-c001 --namespace=lf-tdd-namespace --admin
```

Now set the **kubectl** context to point to your new workload cluster. You can do this following Option A or Option B below:

Option A: Use the **kubectl** tool

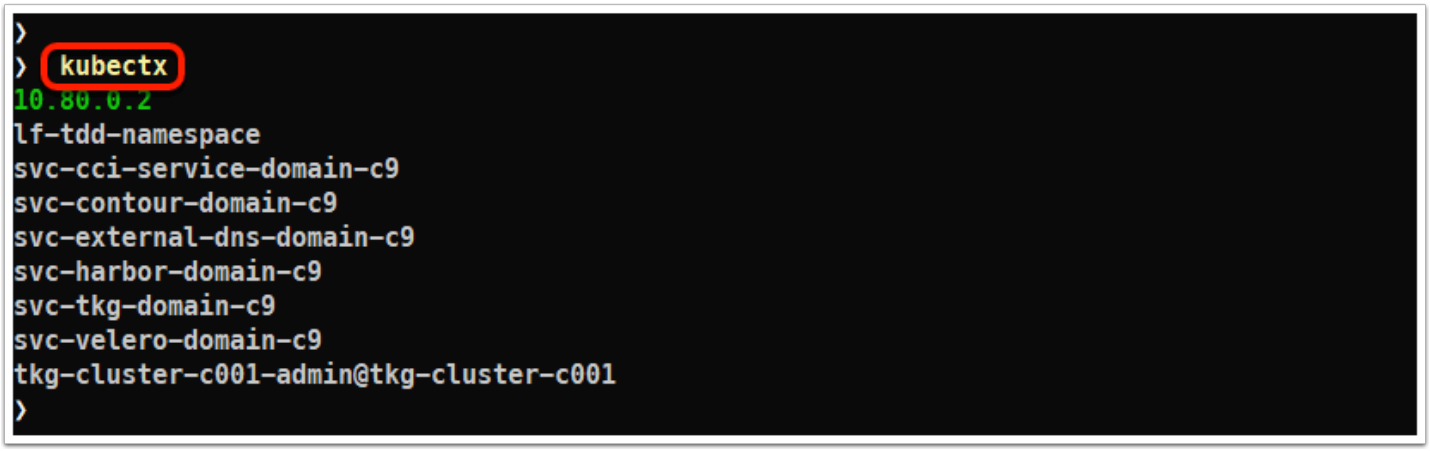

```
kubectl config use-context tkg-cluster-c001-admin@tkg-cluster-c001
```

Option B: Use the **kubectx** tool

Pro tip: Alternatively, for changing the defaut kubectl context, you can use the **kubectx** tool, available on your PS Terminal.

It can list the available contexts for you

Manage a TKG Workload Cluster

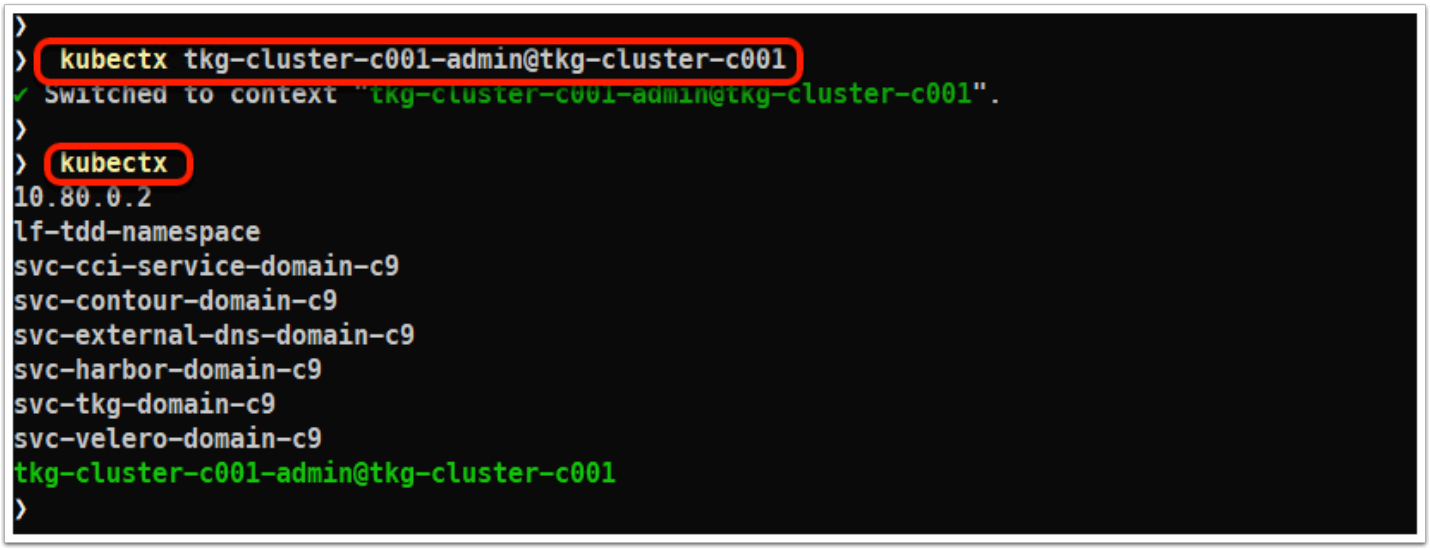



```
kubectx
```

The active/default context is shown in green.

And you can easily switch to what's needed for you

Manage a TKG Workload Cluster




```
kubectx tkg-cluster-c001-admin@tkg-cluster-c001
```

In your workload cluster context:

Get info on the nodes

Manage a TKG Workload Cluster

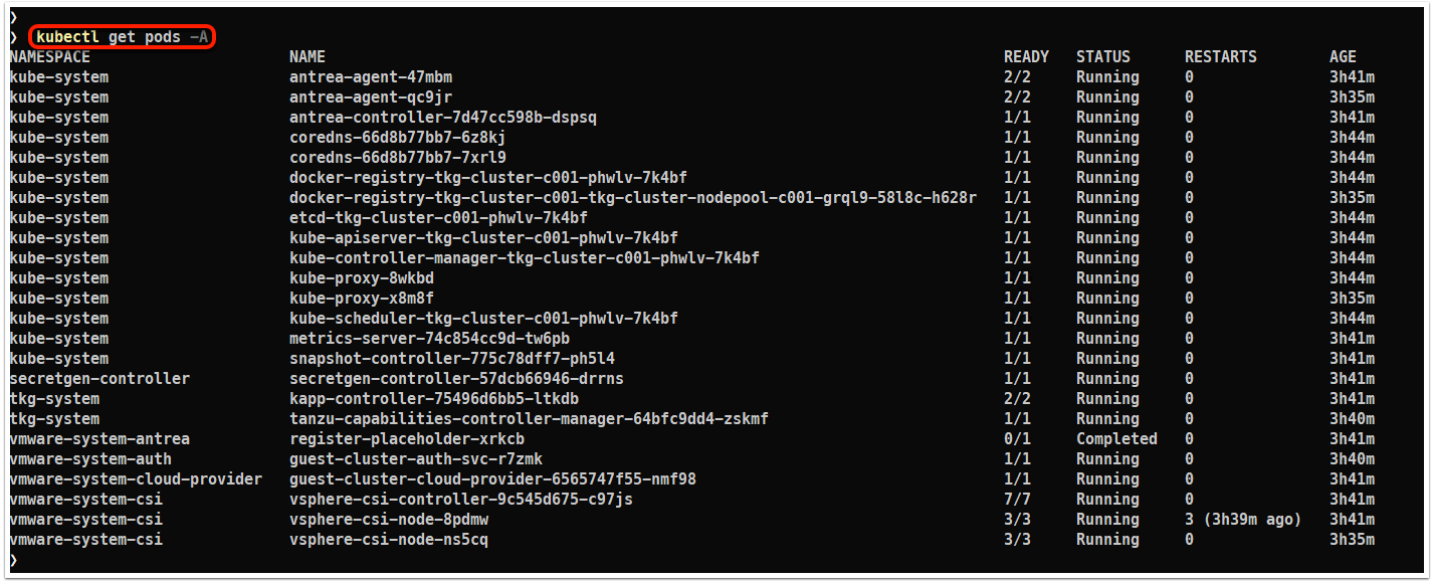

```
kubectl get nodes -o wide
```

Please, note the VM name of your workload TKG cluster node!

In the example above its name ends with `-h628r` Yours should be different. Just note down whatever your end-of-name is.

Get info on the pods

Manage a TKG Workload Cluster

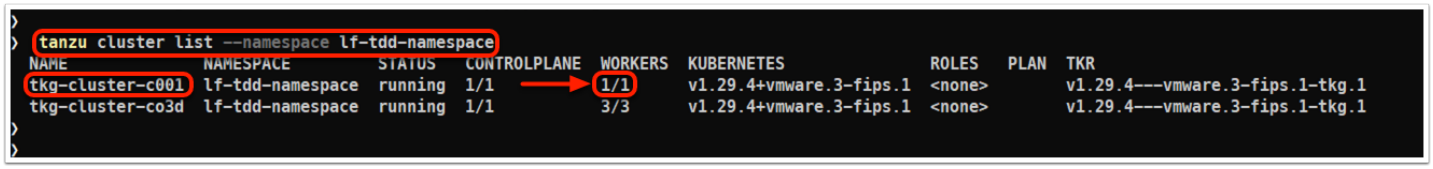


```
❯ kubectl get pods -A
NAMESPACE                      NAME                                                                READY   STATUS      RESTARTS        AGE
kube-system                    antrea-agent-47mbm                                                  2/2     Running     0               3h41m
kube-system                    antrea-agent-qc9jr                                                  2/2     Running     0               3h35m
kube-system                    antrea-controller-7d47cc598b-dspsq                                  1/1     Running     0               3h41m
kube-system                    coredns-66d8b77bb7-6z8kj                                            1/1     Running     0               3h44m
kube-system                    coredns-66d8b77bb7-7xrl9                                            1/1     Running     0               3h44m
kube-system                    docker-registry-tkg-cluster-c001-phwlv-7k4bf                        1/1     Running     0               3h44m
kube-system                    docker-registry-tkg-cluster-c001-tkg-cluster-nodepool-c001-grql9-58l8c-h628r   1/1     Running     0               3h35m
kube-system                    etcd-tkg-cluster-c001-phwlv-7k4bf                                   1/1     Running     0               3h44m
kube-system                    kube-apiserver-tkg-cluster-c001-phwlv-7k4bf                         1/1     Running     0               3h44m
kube-system                    kube-controller-manager-tkg-cluster-c001-phwlv-7k4bf                1/1     Running     0               3h44m
kube-system                    kube-proxy-8wkbd                                                    1/1     Running     0               3h44m
kube-system                    kube-proxy-x8m8f                                                    1/1     Running     0               3h35m
kube-system                    kube-scheduler-tkg-cluster-c001-phwlv-7k4bf                         1/1     Running     0               3h44m
kube-system                    metrics-server-74c854cc9d-tw6pb                                     1/1     Running     0               3h41m
kube-system                    snapshot-controller-775c78dff7-ph5l4                                1/1     Running     0               3h41m
secretgen-controller           secretgen-controller-57dcb66946-drrns                               1/1     Running     0               3h41m
tkg-system                     kapp-controller-75496d6bb5-ltkdb                                    2/2     Running     0               3h41m
tkg-system                     tanzu-capabilities-controller-manager-64bfc9dd4-zskmf               1/1     Running     0               3h40m
vmware-system-antrea           register-placeholder-xrkcb                                          0/1     Completed   0               3h41m
vmware-system-auth             guest-cluster-auth-svc-r7zmk                                        1/1     Running     0               3h40m
vmware-system-cloud-provider   guest-cluster-cloud-provider-6565747f55-nmf98                       1/1     Running     0               3h41m
vmware-system-csi              vsphere-csi-controller-9c545d675-c97js                              7/7     Running     0               3h41m
vmware-system-csi              vsphere-csi-node-8pdmw                                              3/3     Running     3 (3h39m ago)   3h41m
vmware-system-csi              vsphere-csi-node-ns5cq                                              3/3     Running     0               3h35m
❯
```

```
kubectl get pods -A
```

Scale a TKG Workload Cluster

You add capacity to the Tanzu Kubernetes Grid workload cluster by scaling the number of worker nodes.



```
❯ tanzu cluster list --namespace lf-tdd-namespace
  NAME              NAMESPACE         STATUS   CONTROLPLANE  WORKERS  KUBERNETES               ROLES   PLAN  TKR
  tkg-cluster-c001  lf-tdd-namespace  running  1/1           1/1      v1.29.4+vmware.3-fips.1  <none>        v1.29.4---vmware.3-fips.1-tkg.1
  tkg-cluster-co3d  lf-tdd-namespace  running  1/1           3/3      v1.29.4+vmware.3-fips.1  <none>        v1.29.4---vmware.3-fips.1-tkg.1
❯
❯
```

```
tanzu cluster list --namespace lf-tdd-namespace
```

You have just one worker in the workload cluster.

Let's add another one via Terminal:

Scale a TKG Workload Cluster

```
❯ tanzu cluster scale tkg-cluster-c001 --namespace lf-tdd-namespace --worker-machine-count 2
Workload cluster 'tkg-cluster-c001' is being scaled
❯
```

```
tanzu cluster scale tkg-cluster-c001 --namespace lf-tdd-namespace --worker-machine-
count 2
```

The scale operation takes place in the background.

Check the status of the workload cluster, being expanded with a workload node:

Scale a TKG Workload Cluster


```
tanzu cluster list --namespace lf-tdd-namespace
```

It takes a couple of minutes to provision the second workload node and add it to the cluster.

Actually, you can get a bit more wider picture about it:

Scale a TKG Workload Cluster

1. Change your kubectl context to the Supervisor and
2. Monitor the cluster API resources.

```
kubectx 10.80.0.2
```

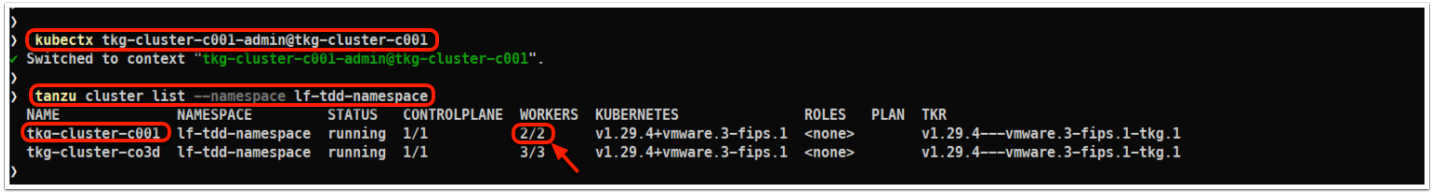
```
kubectl get machines -n lf-tdd-namespace --watch
```

3. At the begging of the section, you were asked to note the name of your VM representing the initial workload node. It ended with `-h628r`.

4. Now you can distinguish the other node, being added to the cluster - a different VM, which name ends on `-kvbx5`. Just as an example.
5. Please wait for the new VM phase to change from **Provisioning** to **Running** before continuing.
6. Press **Ctrl+C to exit** the **watch** command.

Once you get the new workload node ready, please change your kubeconfig context back to your workload cluster and inspect it:

Scale a TKG Workload Cluster




```
kubectx tkg-cluster-c001-admin@tkg-cluster-c001
```

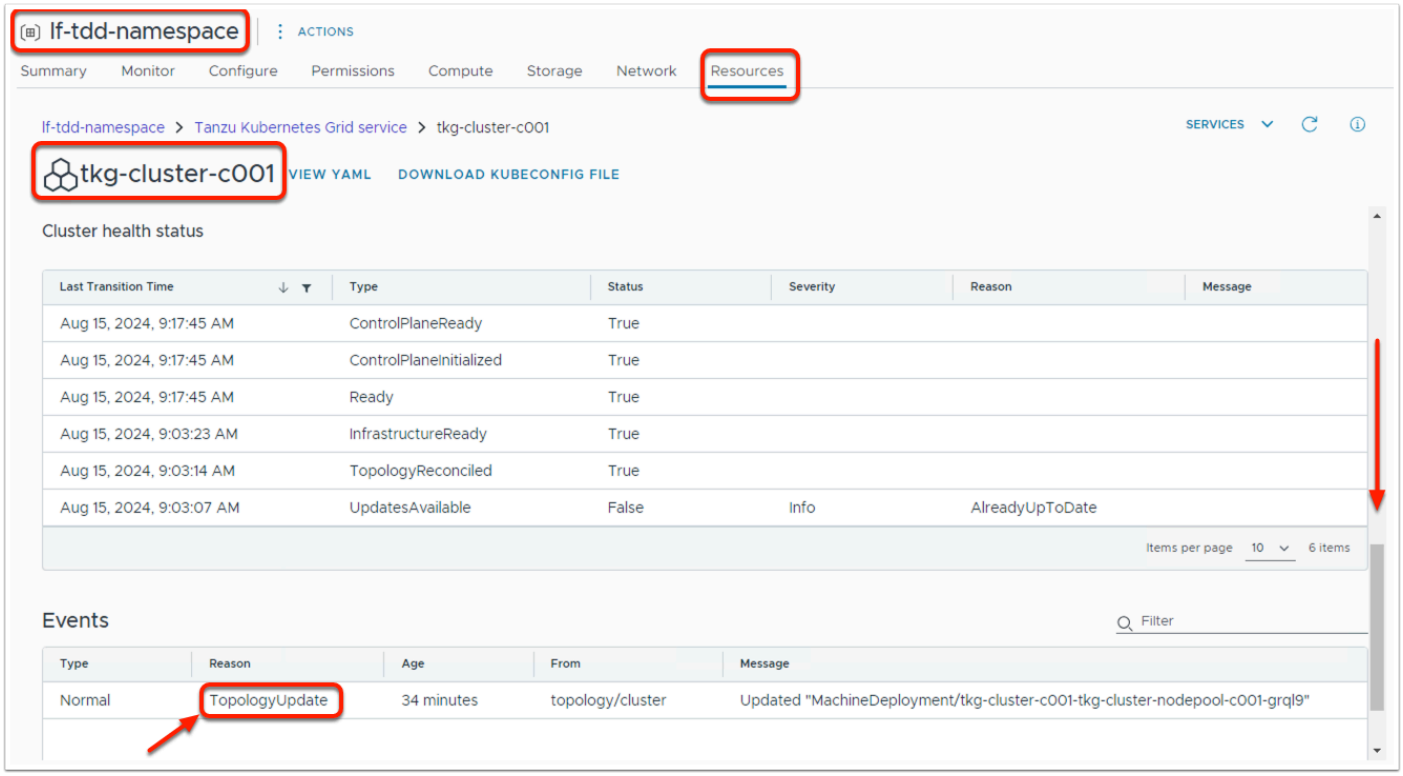
```
tanzu cluster list --namespace lf-tdd-namespace
```

The output shows STATUS running and WORKERS 2/2.

You can also double-check this in the vSphere UI:

In your **Workload Management** --> **Namespaces** --> **lf-tdd-namespace** pane, you will see the new cluster Toplogy update. Make sure you scroll the slide bar to the bottom to reveal the info:

Scale a TKG Workload Cluster

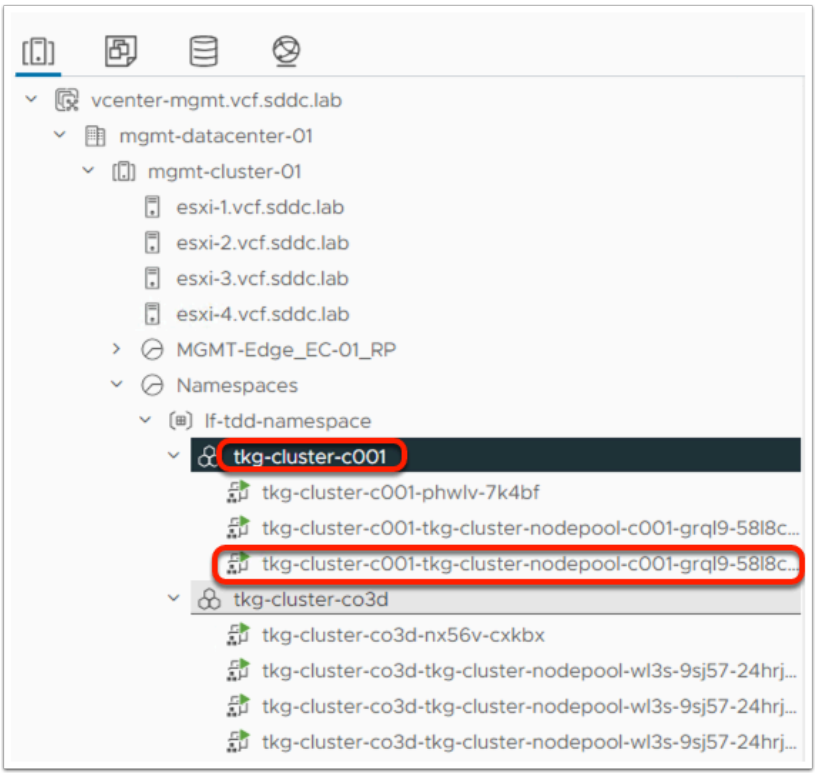


Check the:

- Cluster health status

- Events

Additionally, while in the vSphere client:



From **Inventory** --> **VMs and Clusters** --> under **lf-tdd-namespace**

You will see the second workload node added to your recent cluster.

Packaging in a TKG Workload Cluster

Task description and objectives

Your task is to examine the functionality of Tanzu CLI ans its Package plug-in to manage repositories and packages in a TKG workload cluster. You will get the experience of working with Tanzu Package CLI Plug-In and repositories. To make sure hosting modern Applications on top of your workload TKG cluster will go smooth, a bunch of packages must be installed in addition.

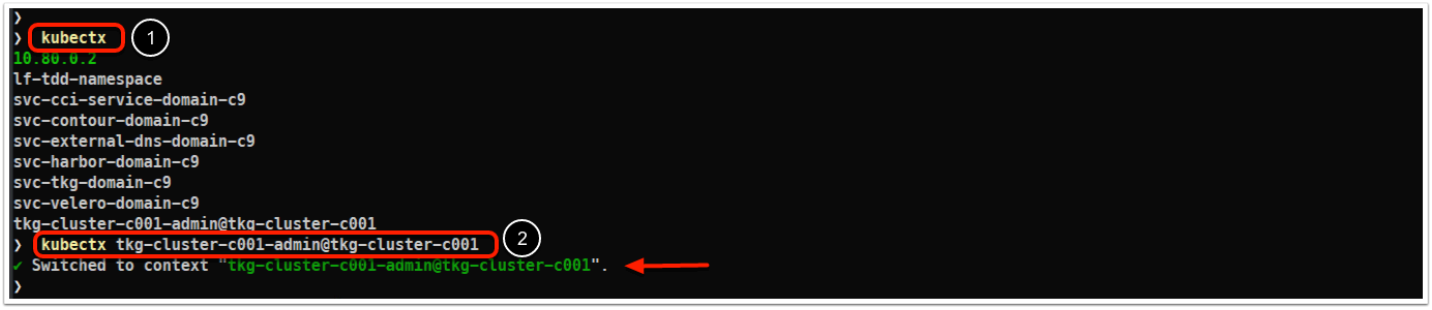
IMPORTANT:

This exercise is being done with administrator@vsphere.local permissions set over the **lf-tdd-namespace.** Make sure you are logged in your **Supervisor**

Your **kubeconfig** context must be set to the **tkg-cluster-c001** cluster!

Working context

Make sure you are working on the **tkg-cluster-c001** workload cluster you created and managed in previous steps




```
kubectx
```

```
kubectx tkg-cluster-c001-admin@tkg-cluster-c001
```

Packaging

In your workload cluster context: Get info on the package repositories, if any available at this point

```
❯
❯ tanzu package repository list -A
  NAMESPACE  NAME  SOURCE  STATUS
❯
```

```
tanzu package repository list -A
```

No repos are configured yet. Let's add some:

Packaging

```
❯
❯ tanzu package repository add standard-repo --url projects.registry.vmware.com/tkg/packages/standard/repo:v2024.5.16 -n tkg-system
Waiting for package repository to be added

7:36:53AM: Waiting for package repository reconciliation for 'standard-repo'
7:36:54AM: Fetch started (30s ago)
7:37:24AM: Fetching
           | apiVersion: vendir.k14s.io/v1alpha1
           | directories:
           | - contents:
           |   - imgpkgBundle:
           |       image: projects.registry.vmware.com/tkg/packages/standard/repo@sha256:b667d6de3c813b9cefd73075b46404a727a42bfda44dbc0a8cd7cc4a0d219925
           |       tag: v2024.5.16
           |     path: .
           |   path: "0"
           | kind: LockConfig
           |
7:37:24AM: Fetch succeeded
7:37:37AM: Template succeeded
7:37:37AM: Deploy started (12s ago)
7:37:49AM: Deploying
```

1

The screenshot is truncated!

```
           | 7:37:48AM: ---- applying complete [171/171 done] ----
           | 7:37:48AM: ---- waiting complete [171/171 done] ----
           | Succeeded
7:37:49AM | Deploy succeeded
```

```
tanzu package repository add tanzu-standard --url projects.registry.vmware.com/tkg/
packages/standard/repo:v2024.5.16 -n tkg-system
```

Deploy succeeded:

After adding this package repo a reconciliation process kicks-in, and you can see it as **tanzu-standard** under namespace **tkg-system** in your workload cluster, in about of couple of minutes.

Packaging

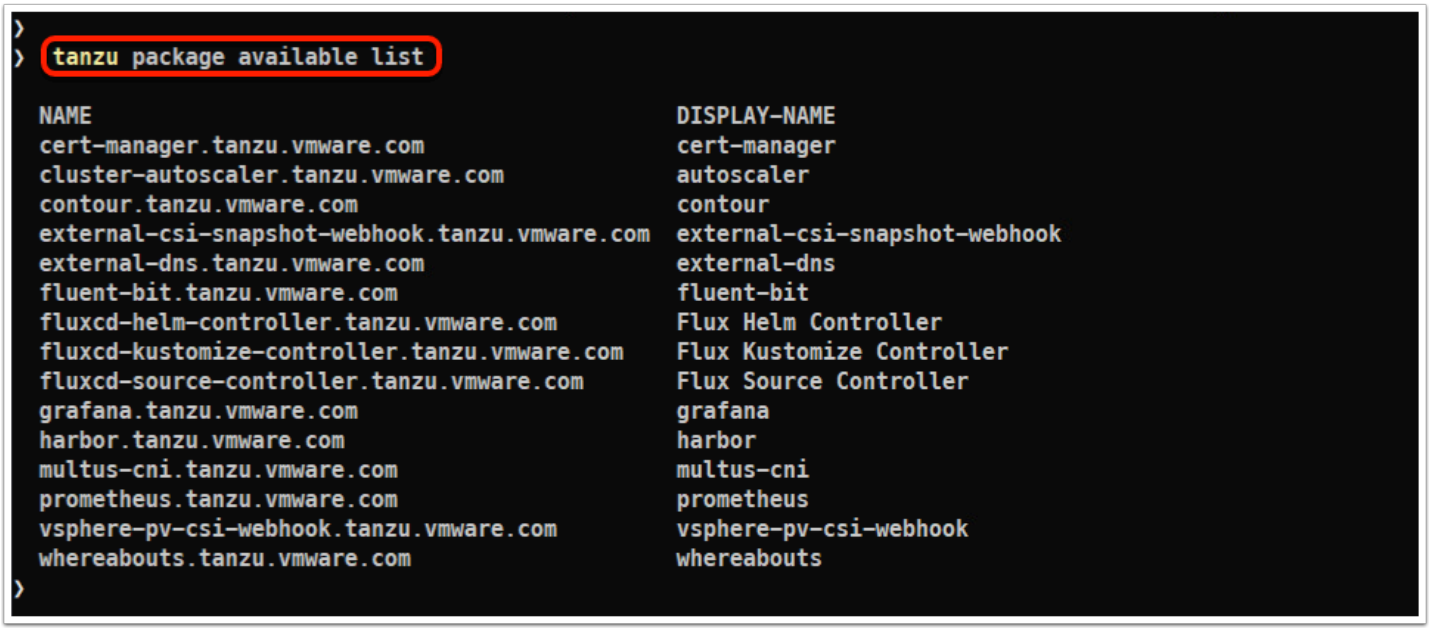
The **tanzu-standard** package repository appears in the **tkg-system** namespace with STATUS `Reconcile succeeded`


```
tanzu package repository list -A
```

The **tanzu-standard** package repository appears in the **tkg-system** namespace with **version v2024.5.16** and **STATUS Reconcile succeeded**

Packaging

Okay, after adding a **tanzu-standard** repo, let's inspect what's available from it as packages

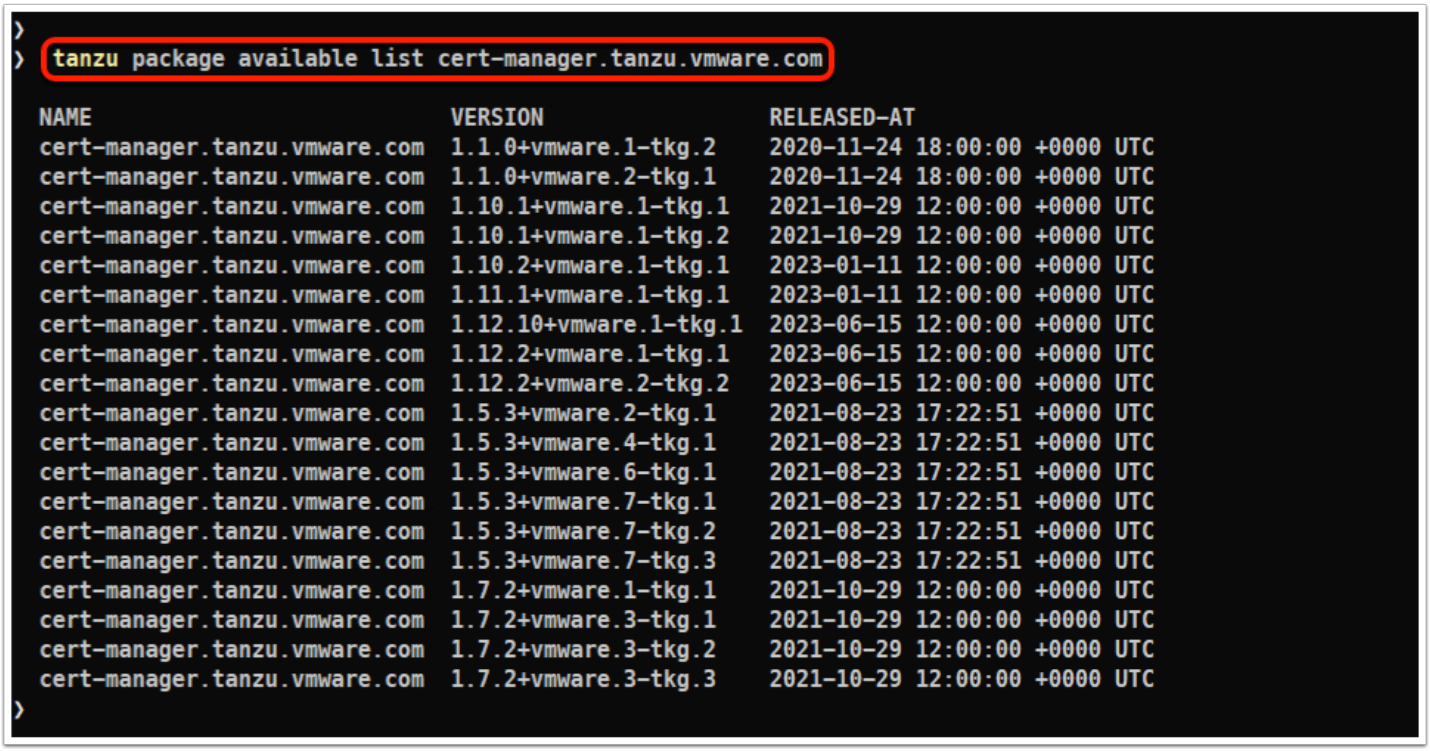



```
tanzu package available list
```

A list of all available packages and their latest versions is seen.

If you focus on any of those, you can retrieve a bunch of additional information, for example **cert-manager**:

Packaging

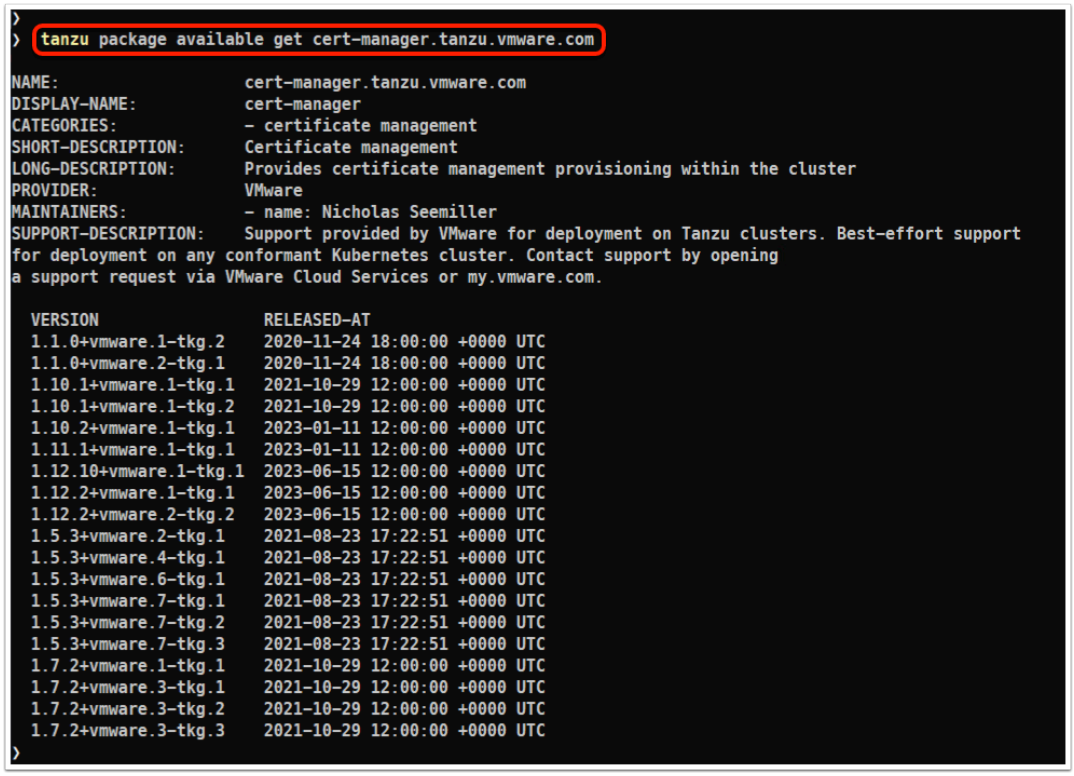


A list of all available versions for the **cert-manager** package is seen

```
tanzu package available list cert-manager.tanzu.vmware.com
```

But you can retrieve some more and get details about the **cert-manager** package:

Packaging




```
tanzu package available get cert-manager.tanzu.vmware.com
```

Information about the cert-manager package appears, including:

- package name
- categories
- description
- support details
- versions

You can List all the installed packages in the cluster that were installed automatically during the cluster creation appear.

Packaging

```
❯ tanzu package installed list -A
  NAMESPACE          NAME                                    PACKAGE-NAME                                    PACKAGE-VERSION              STATUS
  vmware-system-tkg  tkg-cluster-c001-antrea                 antrea.tanzu.vmware.com                         1.13.3+vmware.3-tkg.3-vmware Reconcile succeeded
  vmware-system-tkg  tkg-cluster-c001-capabilities           capabilities.tanzu.vmware.com                   0.33.1+vmware.1              Reconcile succeeded
  vmware-system-tkg  tkg-cluster-c001-gateway-api            gateway-api.tanzu.vmware.com                    1.0.0+vmware.1-tkg.2-vmware  Reconcile succeeded
  vmware-system-tkg  tkg-cluster-c001-guest-cluster-auth-service guest-cluster-auth-service.tanzu.vmware.com 1.3.3+tkg.2-vmware          Reconcile succeeded
  vmware-system-tkg  tkg-cluster-c001-metrics-server         metrics-server.tanzu.vmware.com                 0.6.2+vmware.3-tkg.6-vmware  Reconcile succeeded
  vmware-system-tkg  tkg-cluster-c001-pinniped               pinniped.tanzu.vmware.com                       0.25.0+vmware.2-tkg.2-vmware Reconcile succeeded
  vmware-system-tkg  tkg-cluster-c001-secretgen-controller   secretgen-controller.tanzu.vmware.com           0.16.1+vmware.1-tkg.1-vmware Reconcile succeeded
  vmware-system-tkg  tkg-cluster-c001-vsphere-cpi            vsphere-cpi.tanzu.vmware.com                    1.29.0+vmware.1-tkg.1-vmware Reconcile succeeded
  vmware-system-tkg  tkg-cluster-c001-vsphere-pv-csi         vsphere-pv-csi.tanzu.vmware.com                 3.2.0+vmware.1-tkg.2-vmware  Reconcile succeeded
❯
```

```
tanzu package installed list -A
```

Packaging

So obviously, **cert-manager** is not installed in your workload cluster. Although **cert-manager** is installed automatically in management clusters, you must install it using the `tanzu package` CLI plug-in in workload clusters.

The Installation process is quite neat:

- you create a namespace to fetch the package files into your cluster
- then install the package itself from there, through a reconciliation process

Let's start by creating a cert-manager namespace, where the package needs to be installed on top of your cluster:

Packaging

```
❯ kubectl create ns cert-manager
namespace/cert-manager created
❯
```

```
kubectl create ns cert-manager
```

Packaging

Typically, we install the most recent version.

As a refresher, let's get the versions available:

Packaging

```
❯
❯ tanzu package available list cert-manager.tanzu.vmware.com (1)

  NAME                           VERSION                RELEASED-AT
  cert-manager.tanzu.vmware.com  1.1.0+vmware.1-tkg.2   2020-11-24 18:00:00 +0000 UTC
  cert-manager.tanzu.vmware.com  1.1.0+vmware.2-tkg.1   2020-11-24 18:00:00 +0000 UTC
  cert-manager.tanzu.vmware.com  1.10.1+vmware.1-tkg.1  2021-10-29 12:00:00 +0000 UTC
  cert-manager.tanzu.vmware.com  1.10.1+vmware.1-tkg.2  2021-10-29 12:00:00 +0000 UTC
  cert-manager.tanzu.vmware.com  1.10.2+vmware.1-tkg.1  2023-01-11 12:00:00 +0000 UTC
  cert-manager.tanzu.vmware.com  1.11.1+vmware.1-tkg.1  2023-01-11 12:00:00 +0000 UTC
  cert-manager.tanzu.vmware.com  1.12.10+vmware.1-tkg.1 2023-06-15 12:00:00 +0000 UTC
  cert-manager.tanzu.vmware.com  1.12.2+vmware.1-tkg.1  2023-06-15 12:00:00 +0000 UTC
  cert-manager.tanzu.vmware.com  1.12.2+vmware.2-tkg.2  2023-06-15 12:00:00 +0000 UTC
  cert-manager.tanzu.vmware.com  1.5.3+vmware.2-tkg.1   2021-08-23 17:22:51 +0000 UTC
  cert-manager.tanzu.vmware.com  1.5.3+vmware.4-tkg.1   2021-08-23 17:22:51 +0000 UTC
  cert-manager.tanzu.vmware.com  1.5.3+vmware.6-tkg.1   2021-08-23 17:22:51 +0000 UTC
  cert-manager.tanzu.vmware.com  1.5.3+vmware.7-tkg.1   2021-08-23 17:22:51 +0000 UTC
  cert-manager.tanzu.vmware.com  1.5.3+vmware.7-tkg.2   2021-08-23 17:22:51 +0000 UTC
  cert-manager.tanzu.vmware.com  1.5.3+vmware.7-tkg.3   2021-08-23 17:22:51 +0000 UTC
  cert-manager.tanzu.vmware.com  1.7.2+vmware.1-tkg.1   2021-10-29 12:00:00 +0000 UTC
  cert-manager.tanzu.vmware.com  1.7.2+vmware.3-tkg.1   2021-10-29 12:00:00 +0000 UTC
  cert-manager.tanzu.vmware.com  1.7.2+vmware.3-tkg.2   2021-10-29 12:00:00 +0000 UTC
  cert-manager.tanzu.vmware.com  1.7.2+vmware.3-tkg.3   2021-10-29 12:00:00 +0000 UTC
❯
```

```
tanzu package available list cert-manager.tanzu.vmware.com
```

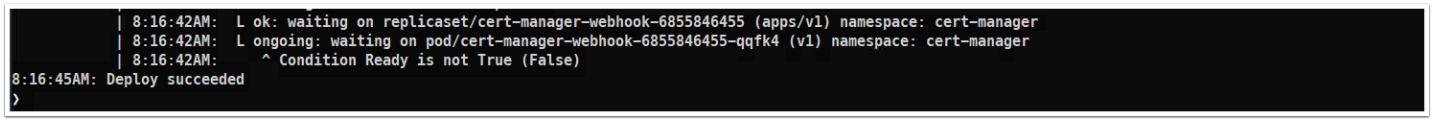
So from this list, you can pick-up a version for install.

Let's get the most recent one: `1.12.2+vmware.2-tkg.2`

Packaging

```
❯
❯ tanzu package install cert-manager -p cert-manager.tanzu.vmware.com -n cert-manager -v 1.12.2+vmware.2-tkg.2

8:15:46AM: Creating service account 'cert-manager-cert-manager-sa'
8:15:46AM: Creating cluster admin role 'cert-manager-cert-manager-cluster-role'
8:15:46AM: Creating cluster role binding 'cert-manager-cert-manager-cluster-rolebinding'
8:15:46AM: Creating overlay secrets
8:15:46AM: Creating package install resource
8:15:46AM: Waiting for PackageInstall reconciliation for 'cert-manager'
8:15:48AM: Fetch started (17s ago)
8:16:06AM: Fetching
           | apiVersion: vendir.k14s.io/v1alpha1
           | directories:
           | - contents:
           |   - imgpkgBundle:
           |       image: projects.registry.vmware.com/tkg/packages/standard/cert-manager@sha256:15563d5759851b6cf59a2e0891ddc2d6d0b204e68130914adf8e62bfb4762a24
           |     path: .
           |   path: "0"
           | kind: LockConfig
           |
8:16:06AM: Fetch succeeded
8:16:07AM: Template succeeded
8:16:07AM: Deploy started (1s ago)
8:16:09AM: Deploying
```

The output on the screenshot is truncated!




```
tanzu package install cert-manager -p cert-manager.tanzu.vmware.com -n cert-manager -v
1.12.2+vmware.2-tkg.2
```

Installation takes up to 5min.

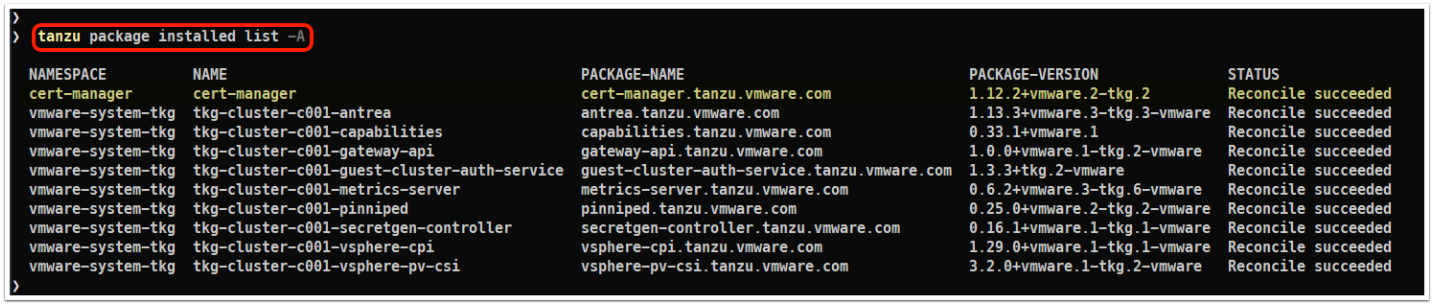
The process involves kapp controller at the backend, implementing the following:

- Creating a dedicated service account
- Creating a cluster admin role
- Creating cluster role binding
- Creating overlay secrets
- Creating package install resource
- Reconciling changes untill package gets installed

You can check the changes and get info update, after the cert-manager gets successfully installed

From list of installed packages perspective:

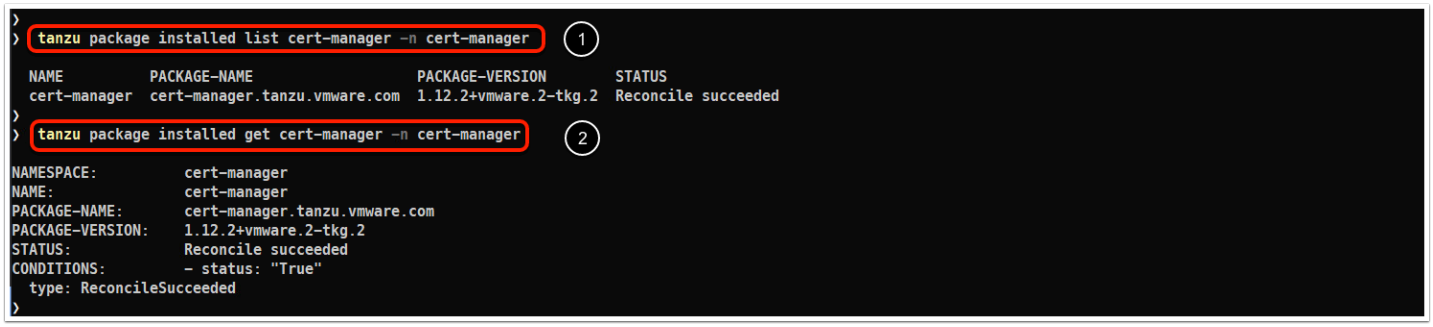
Packaging




```
tanzu package installed list -A
```

From installation namespace perspective:

Packaging



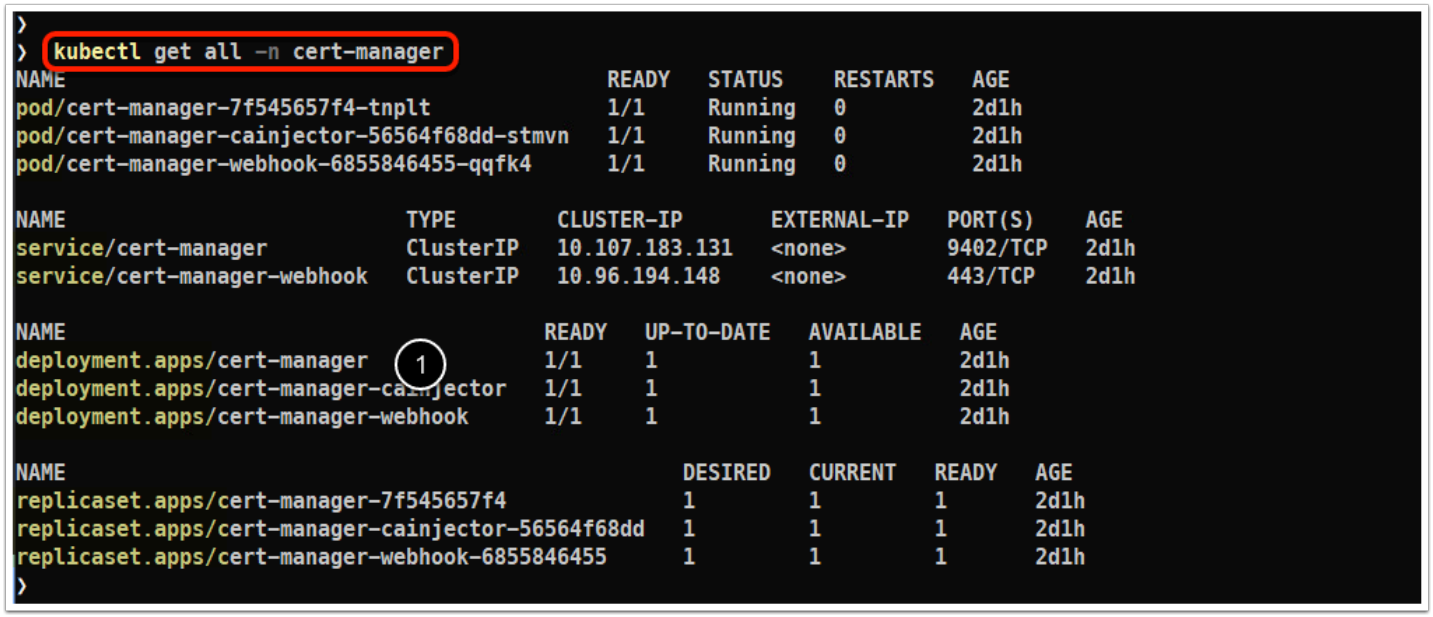

```
tanzu package installed list -n cert-manager
```

```
tanzu package installed get cert-manager -n cert-manager
```

Upon successful installation of **cert-manager** package, you can check:

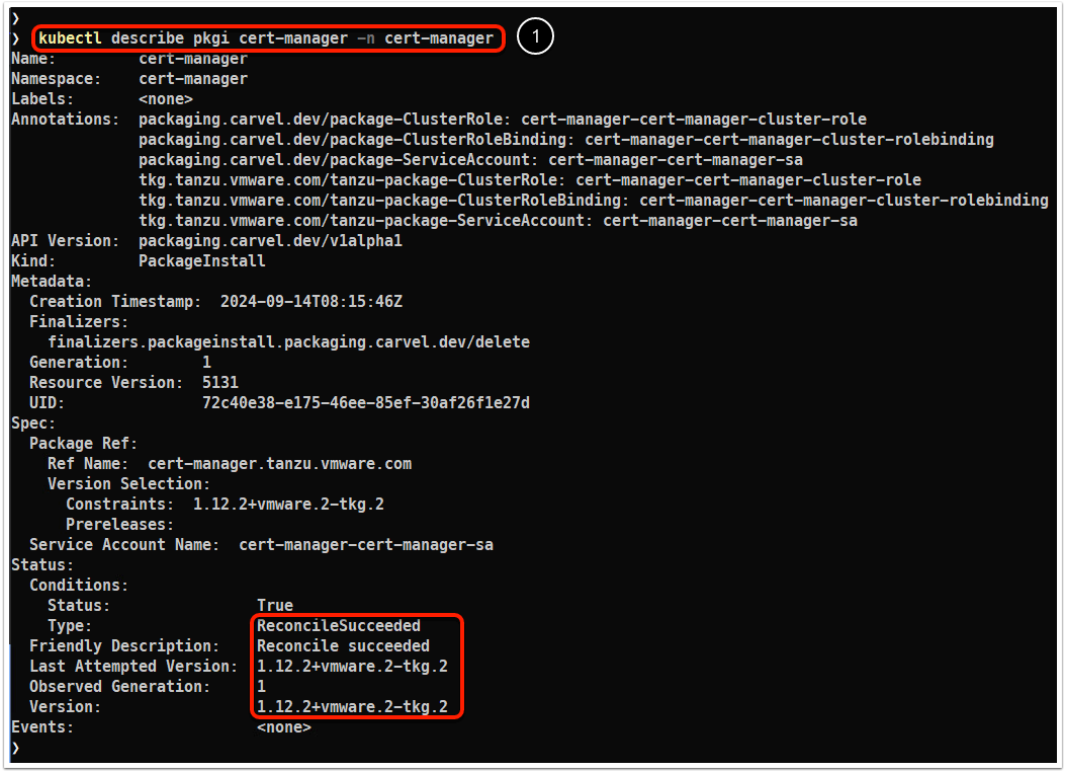
- pods
- services
- deployments
- replicasets

Packaging




```
kubectl get all -n cert-manager
```

In case it is needed, the best approach for troubleshooting is from description perspective




```
kubectl describe pkgi cert-manager -n cert-manager
```

You are now ready to continue with the next chapter!

Set the Image Registry

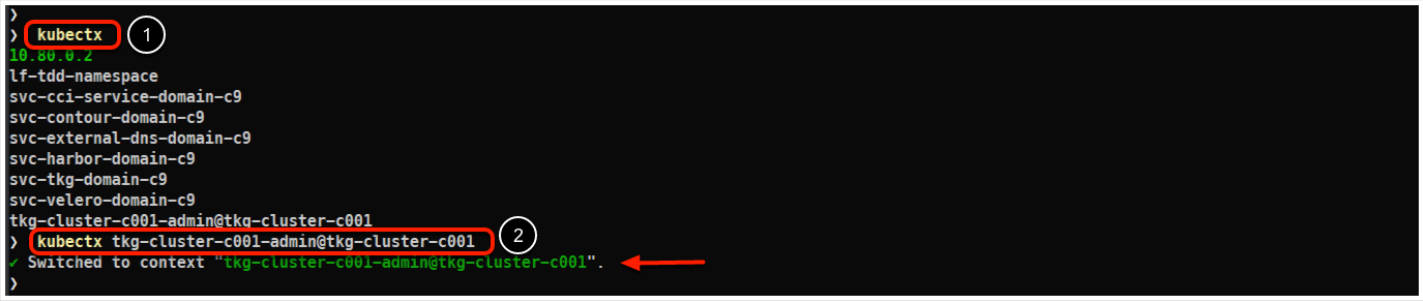
1. Task description and objectives

Your task is to log in to Harbor, create a project, and obtain the certificate for login to the registry.

> **IMPORTANT:**
>
> This exercise is being done with administrator@vsphere.local permissions set over the **lf-tdd-namespace.** Make sure you are logged in your **Supervisor**
>
> Your **kubeconfig** context must be set to the **tkg-cluster-c001** cluster!

Working context

Make sure you are working on the **tkg-cluster-c001** workload cluster you created and managed in previous steps




```
kubectx
```

```
kubectx tkg-cluster-c001-admin@tkg-cluster-c001
```

2. Work with Harbor

2.1. Access and Prepare Harbor

Access Harbor using its FQDN

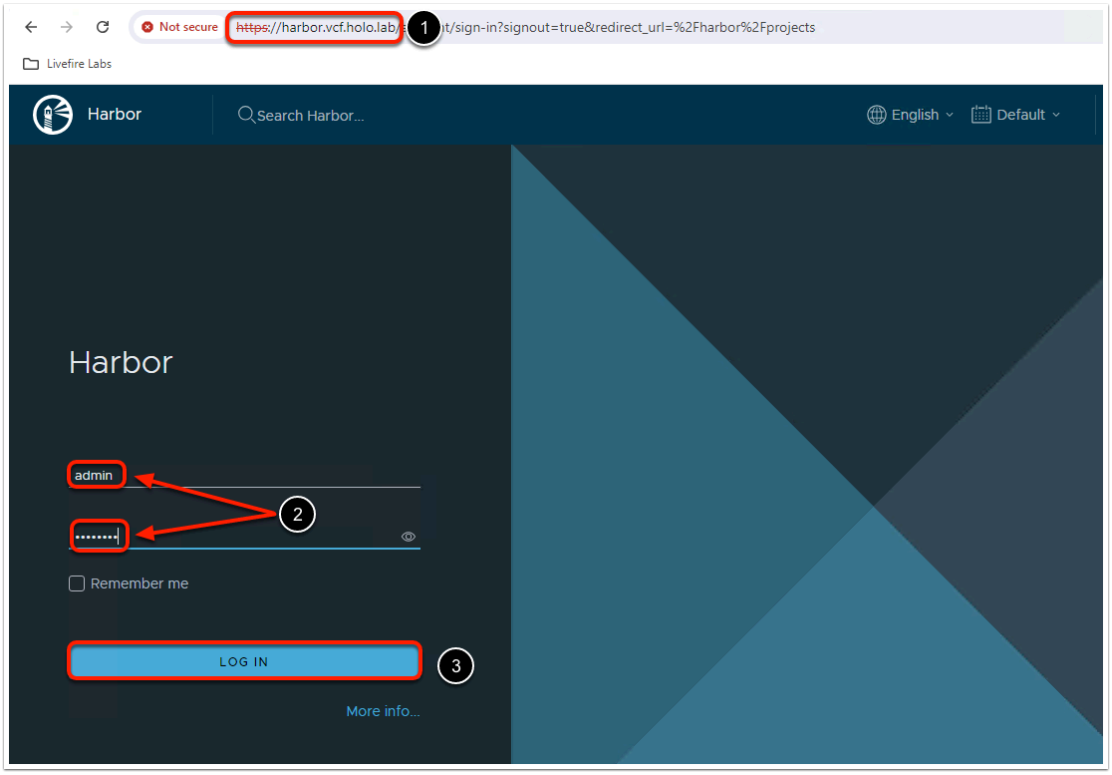
The Harbor FQDN is provided by ExternalDNS, and it is set in the **harbor-data-values.yml** file.

```
❯
❯  cat .\Downloads\harbor-data-values.yml | grep hostname
hostname: harbor.vcf.holo.lab
❯
```

You can review this value from **harbor-data-values.yml** file by running a command in Terminal.

```
cat .\Downloads\harbor-data-values.yml | grep hostname
```

Now let's prepare your environment for Harbor



1. In Chrome, Go to your harbor URL: https://harbor.vcf.holo.lab
2. Provide the credentials
3. Confirm with LOG IN

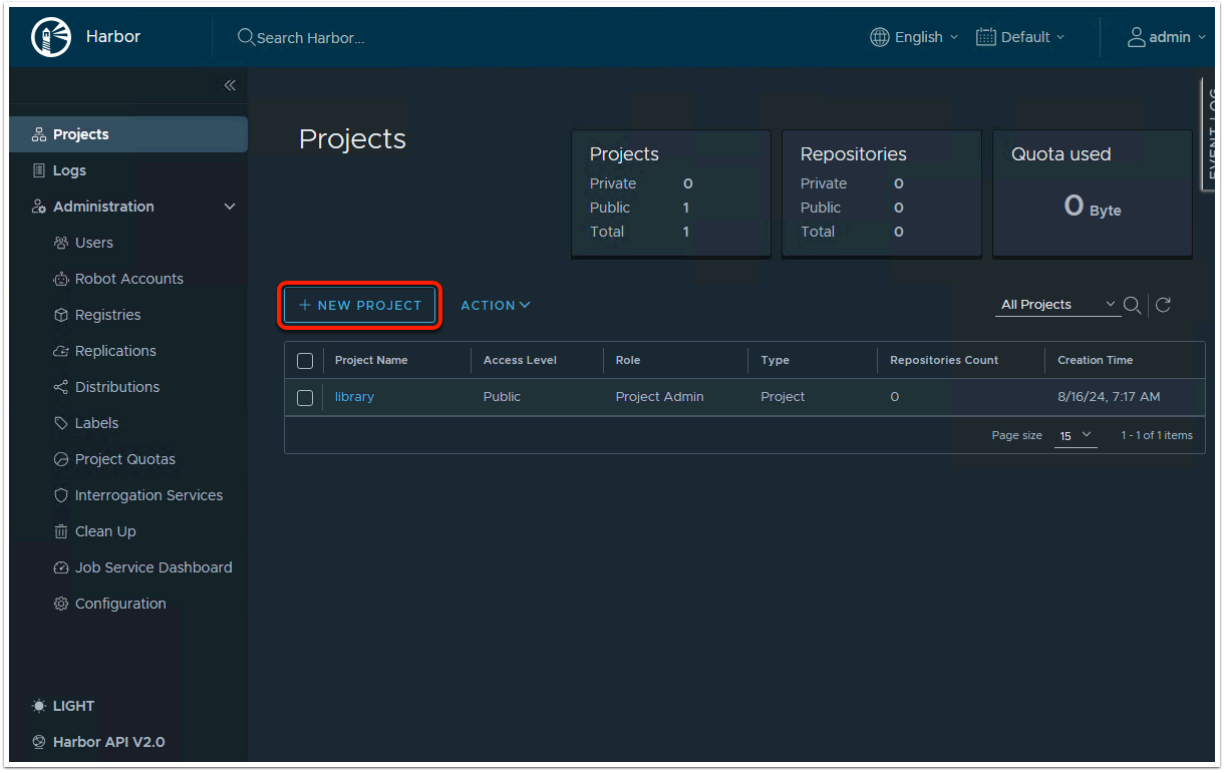
Username:

```
admin
```

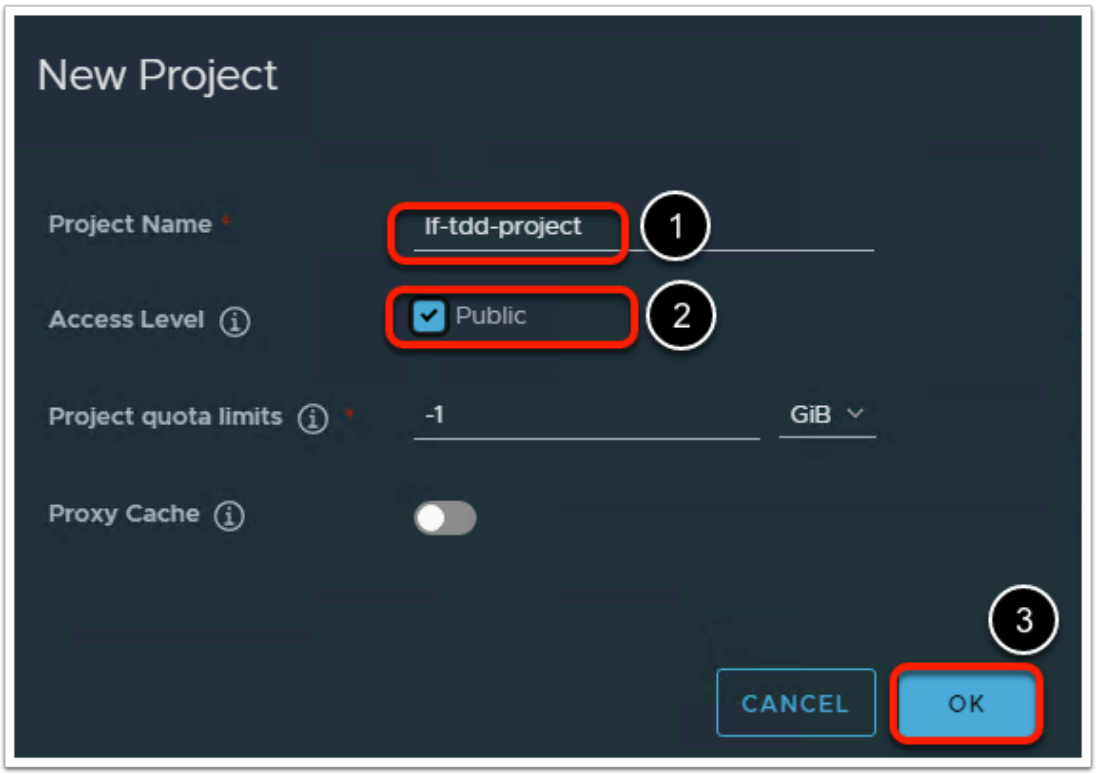
Password:

```
VMware1!
```

Let's create a Project



Click on NEW PROJECT

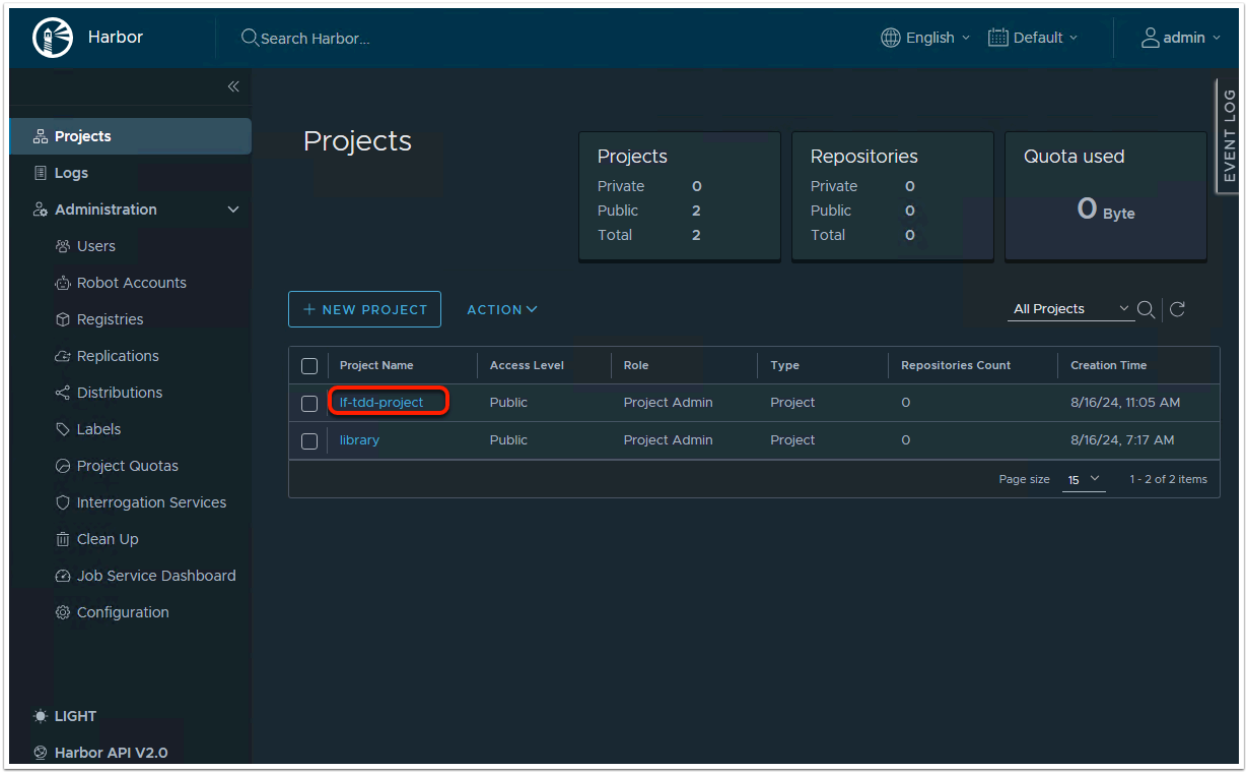


Project Name:

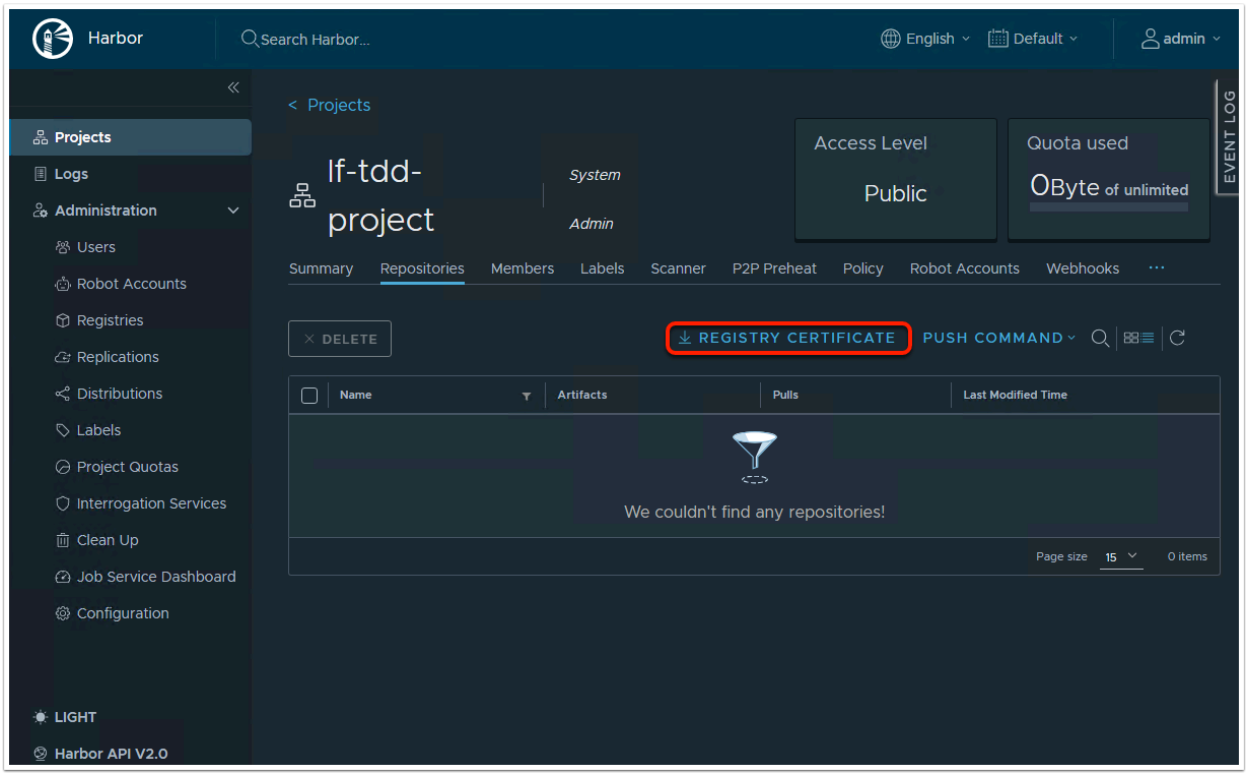
```
lf-tdd-project
```

Access Level: mark Public Leave the other options unchanged and

click **OK**

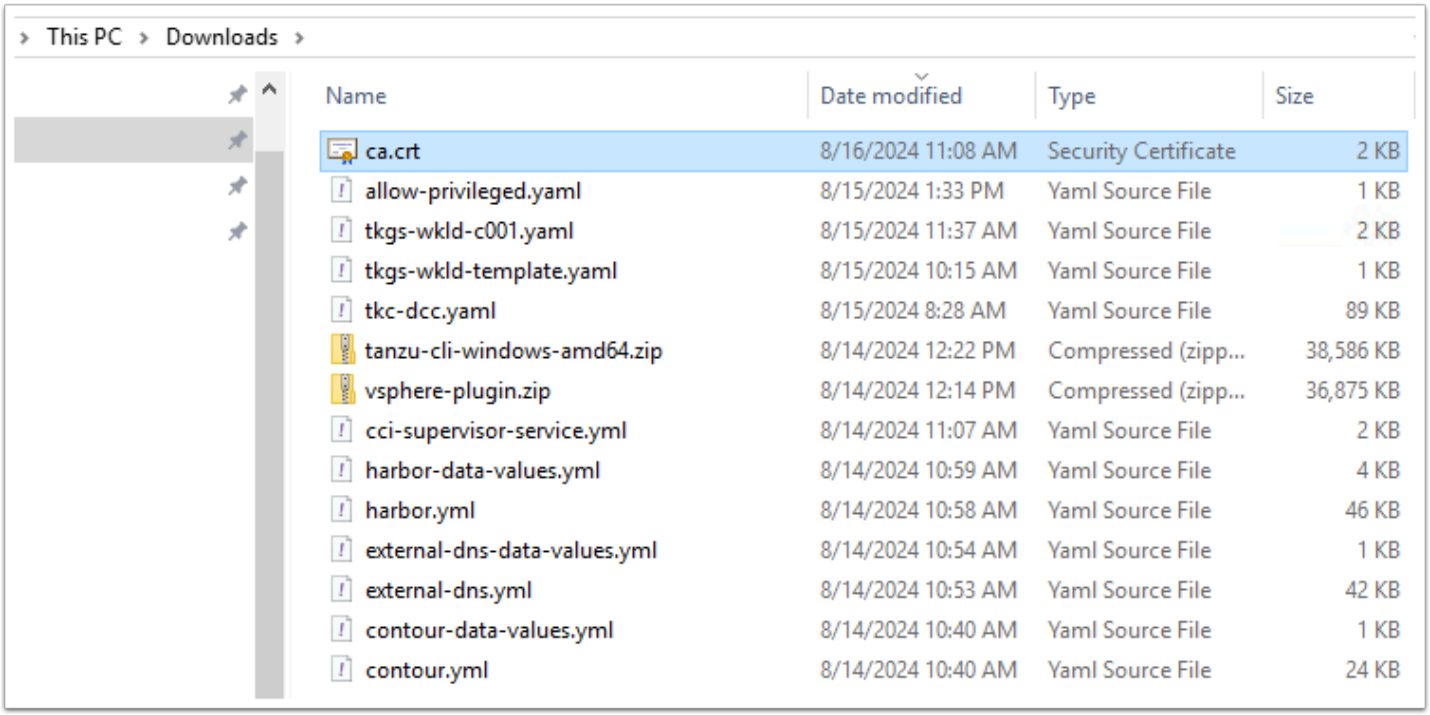


On the Projects page go to your lf-tdd-project

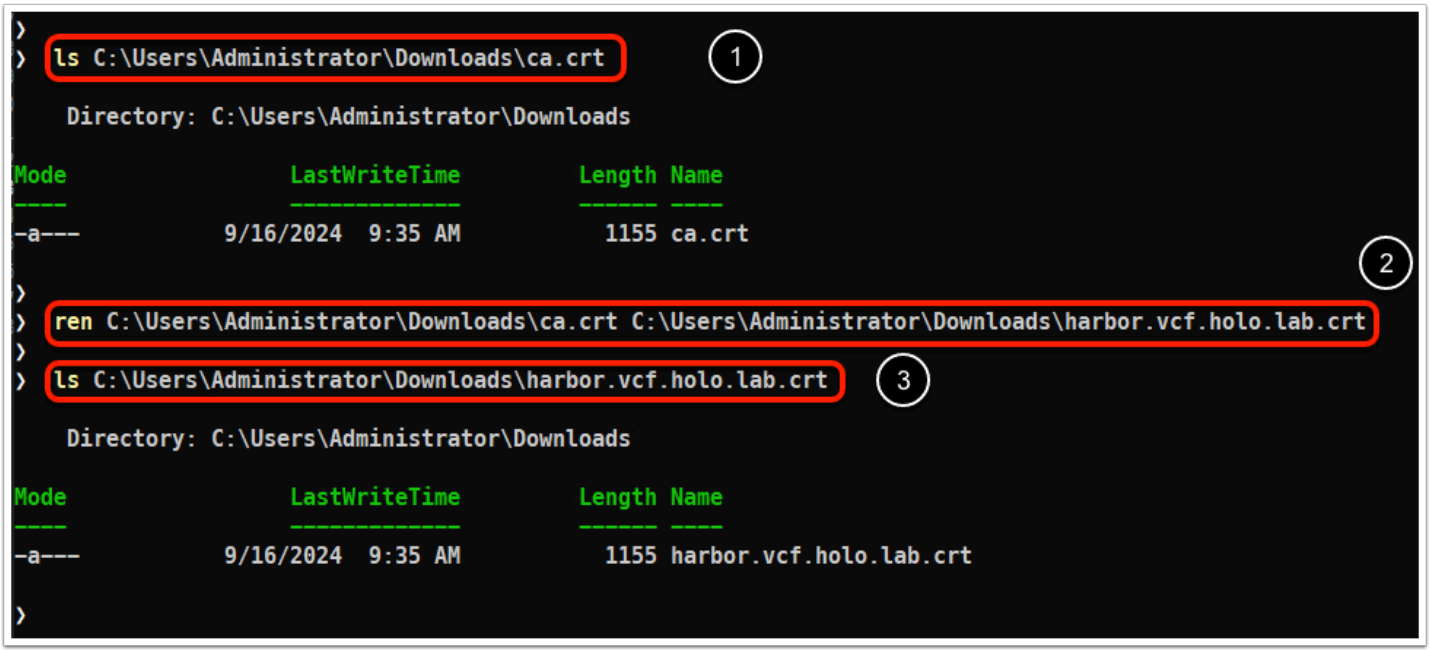


Click on REGISTRY CERTIFICATE

The **ca.cer** file is downloaded in your Downloads folder



Rename this file to more distinguished name:




```
ls C:\Users\Administrator\Downloads\ca.crt
```

```
ren C:\Users\Administrator\Downloads\ca.crt C:\Users\Administrator\Downloads\harbor.vcf.holo.lab.crt
```

```
ls C:\Users\Administrator\Downloads\harbor.vcf.holo.lab.crt
```

2.2. Use the Harbor Repository

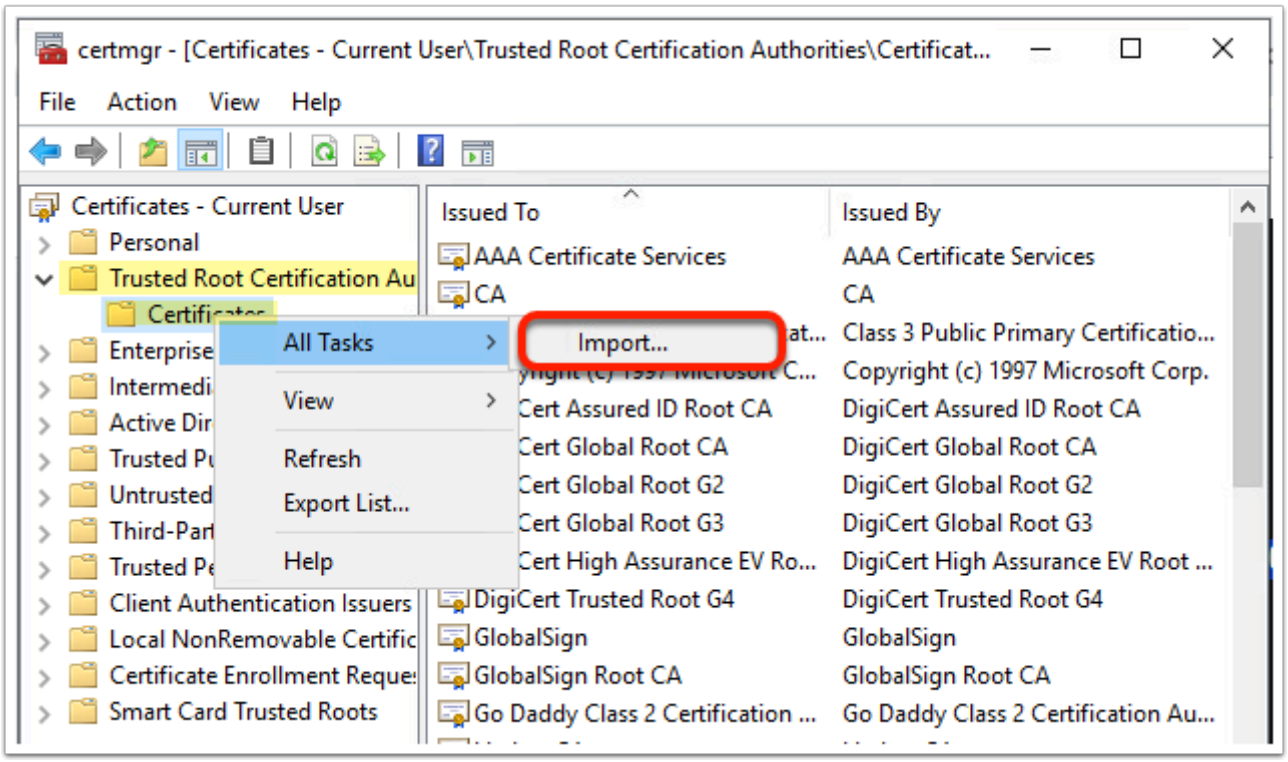
We need to import the **harbor.vcf.holo.lab** certificate file you got in the previous step.

You can do this via the Certificate snap-in for Microsoft Management Console on your Holo-Console machine:

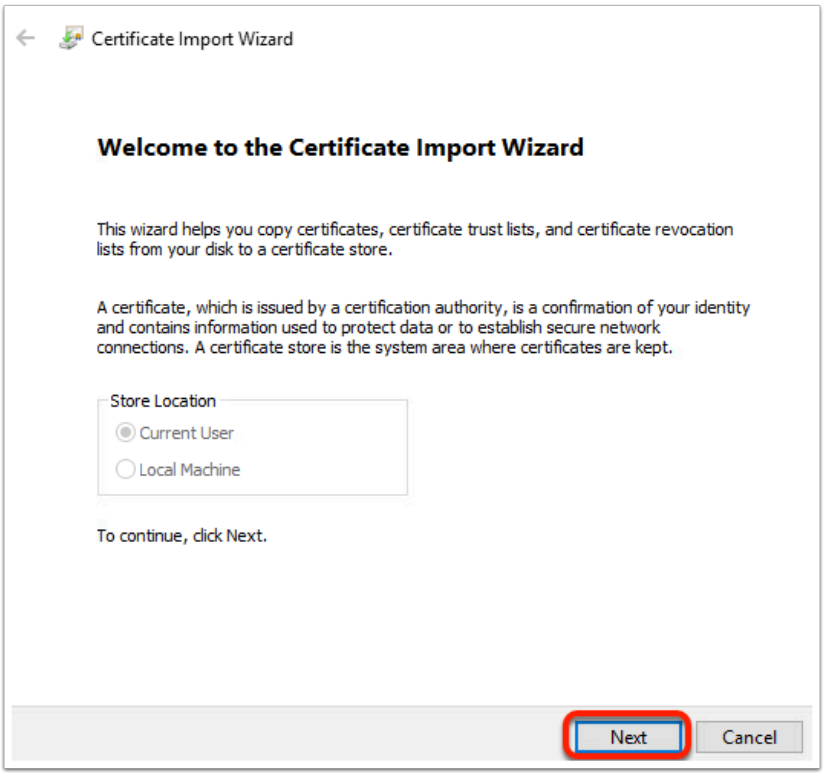
```
certmgr.msc
```

```
certmgr.msc
```

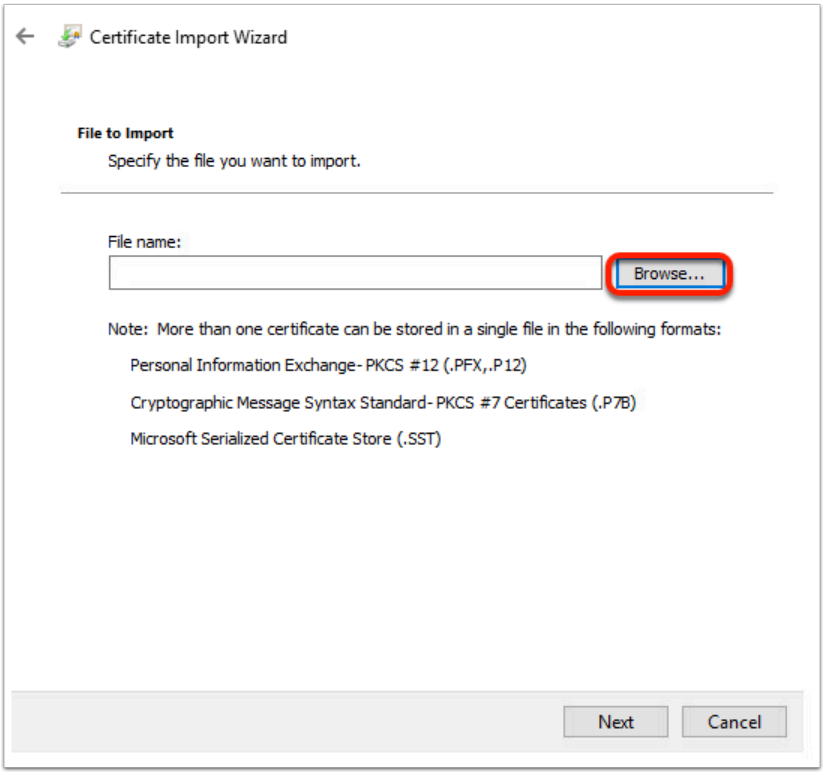
This invokes Microsoft Management Console, where you need to add the needed certificate



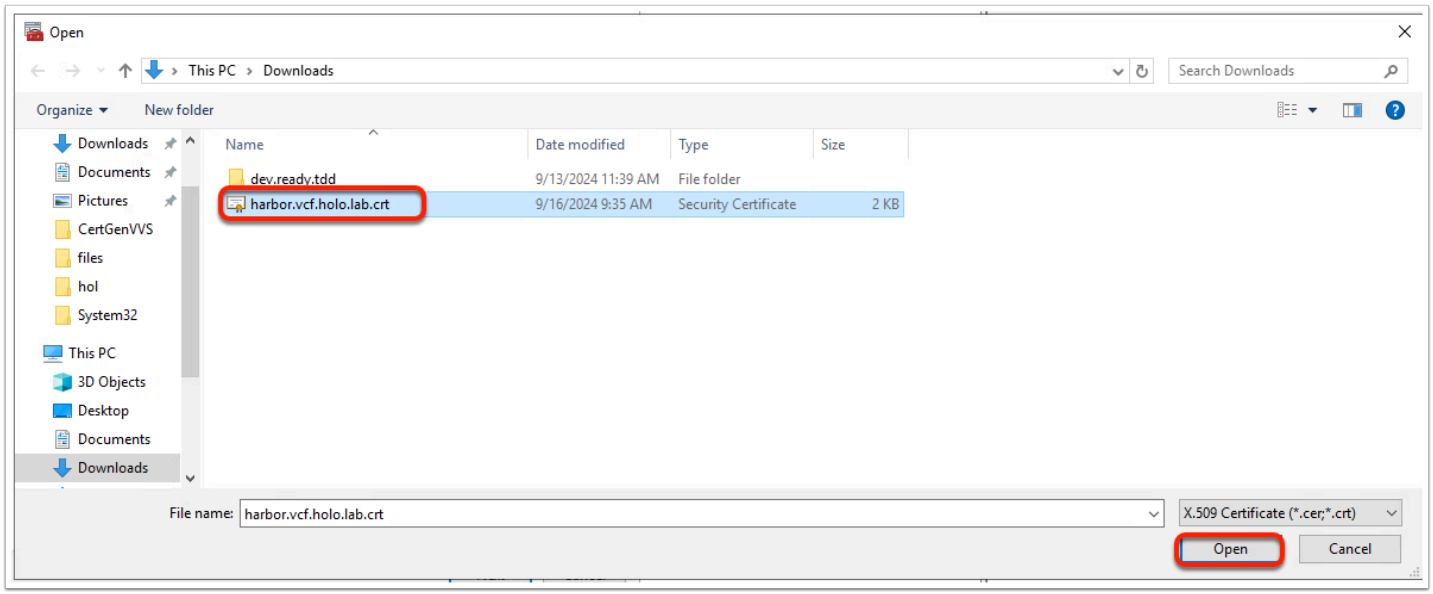
From **Certificates - Current User** -> reveal **Trusted Root Certification Authorities** -> right click on **Certificates** -> select **All Taks** -> then **Import**



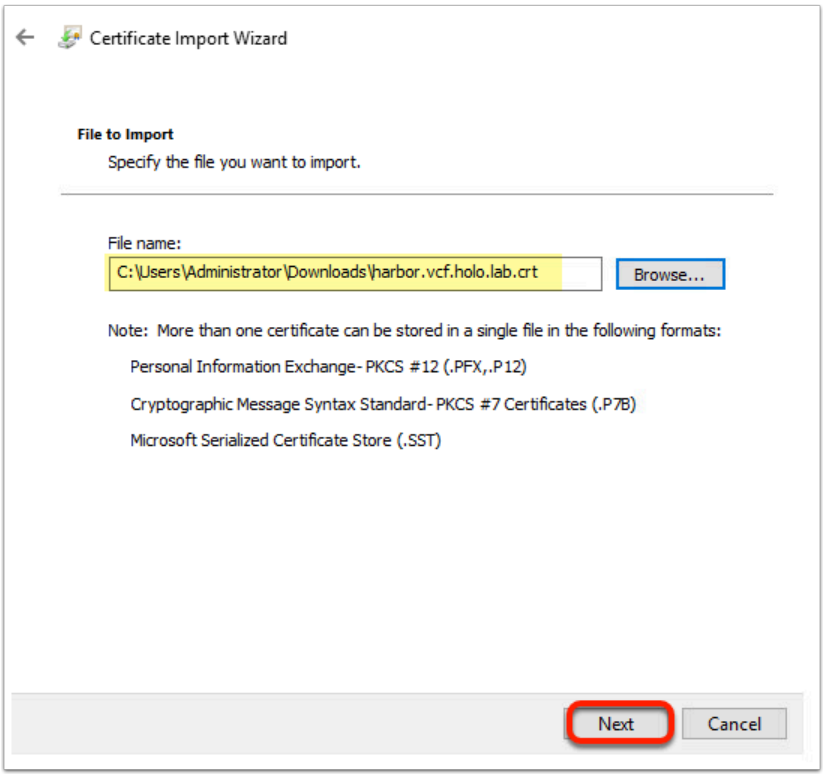
Select Next



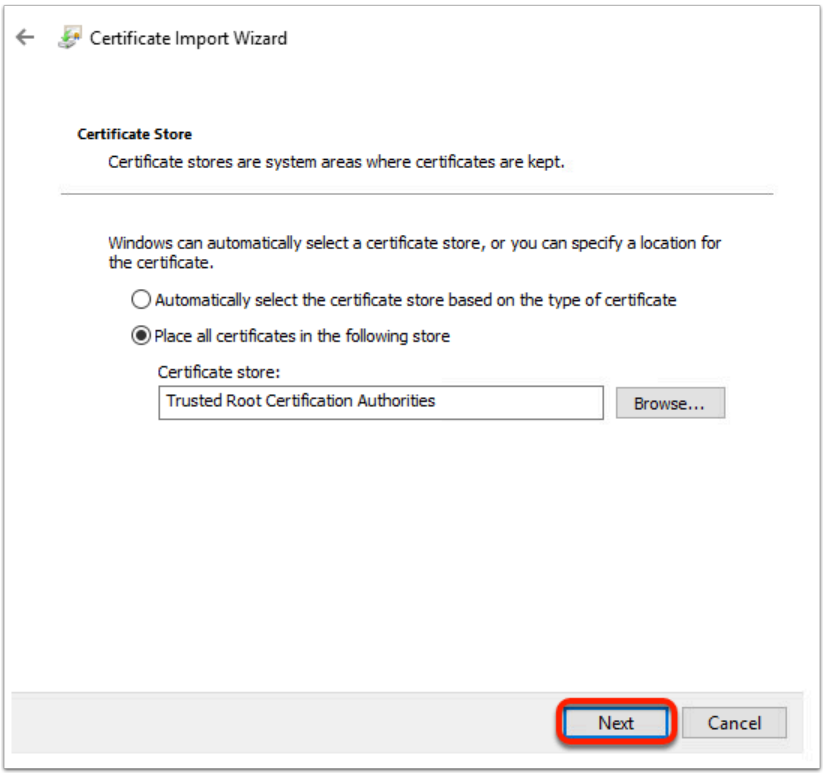
Select **Browse**



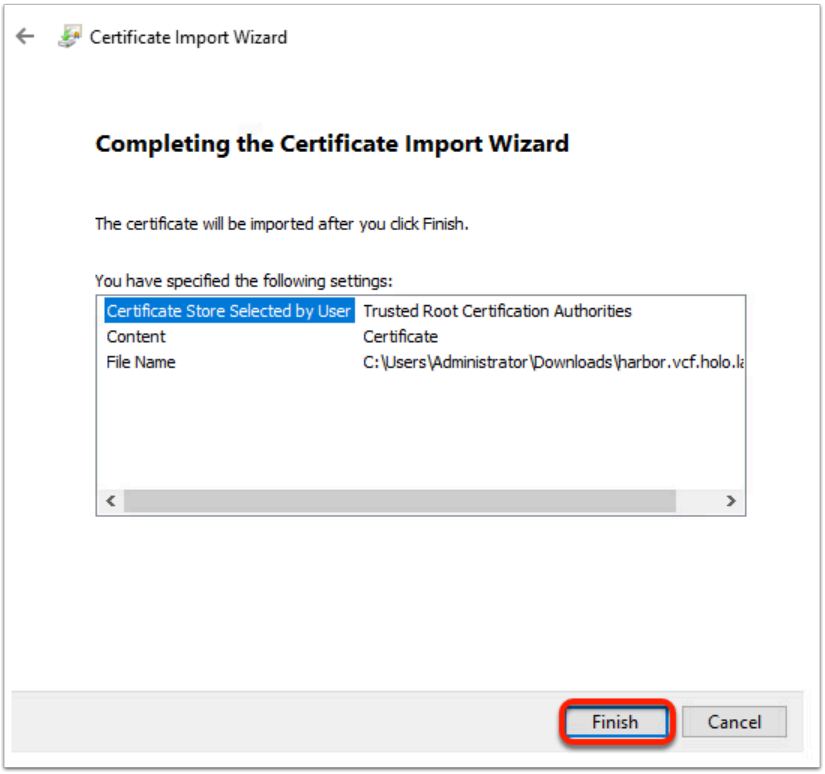
Point your **harbor.vcf.holo.lab** certificate and confirm with **Open**



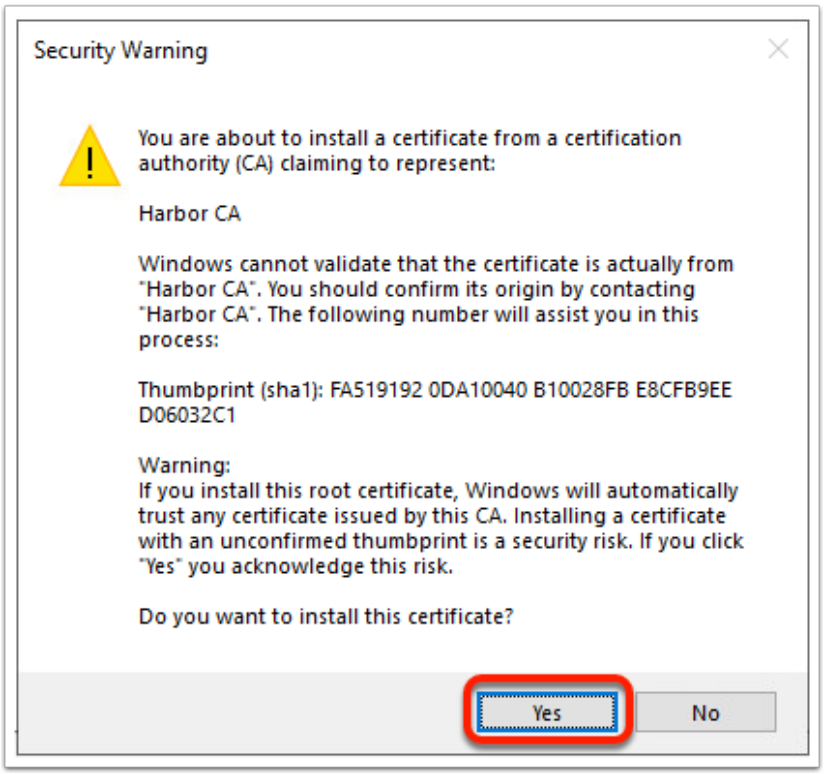
Select **Next**



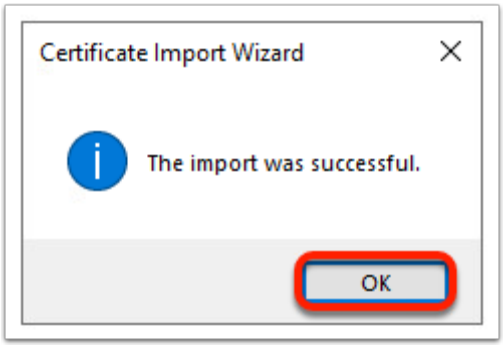
Select **Next**



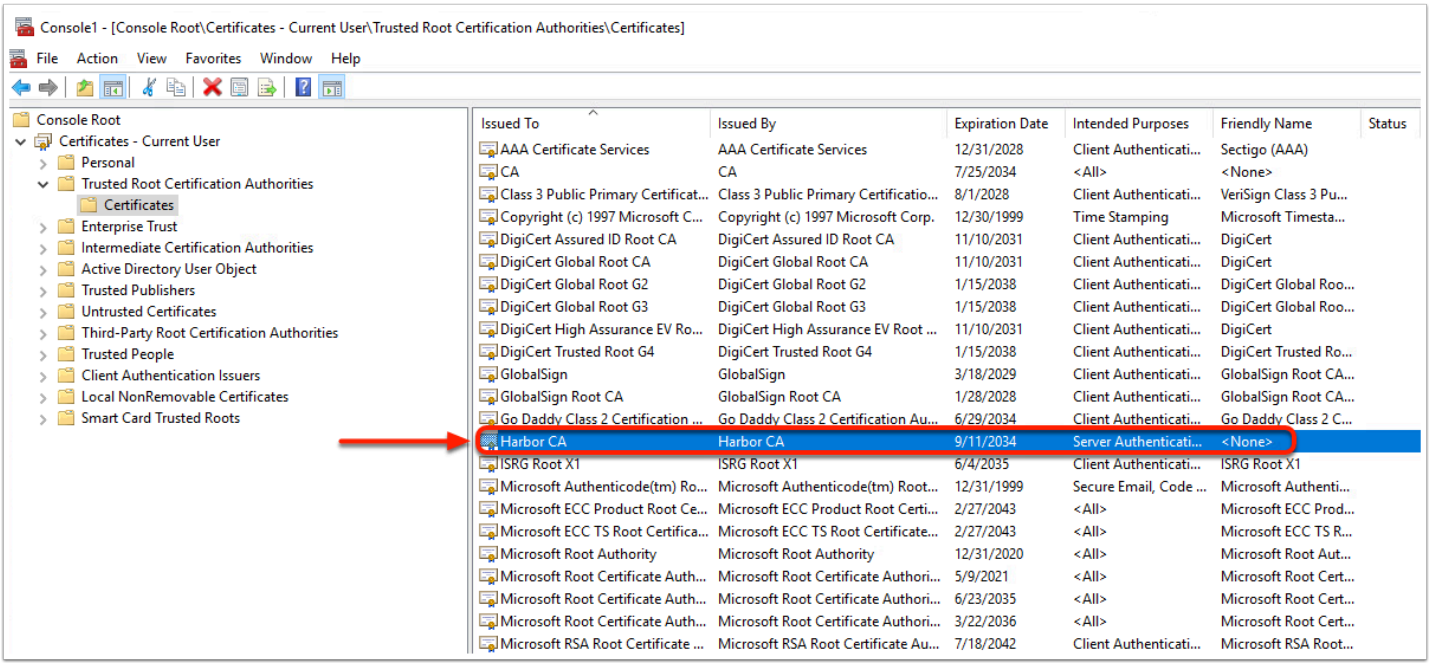
Click on **Finish**



In the **Security Wanring** dialog, select **Yes**

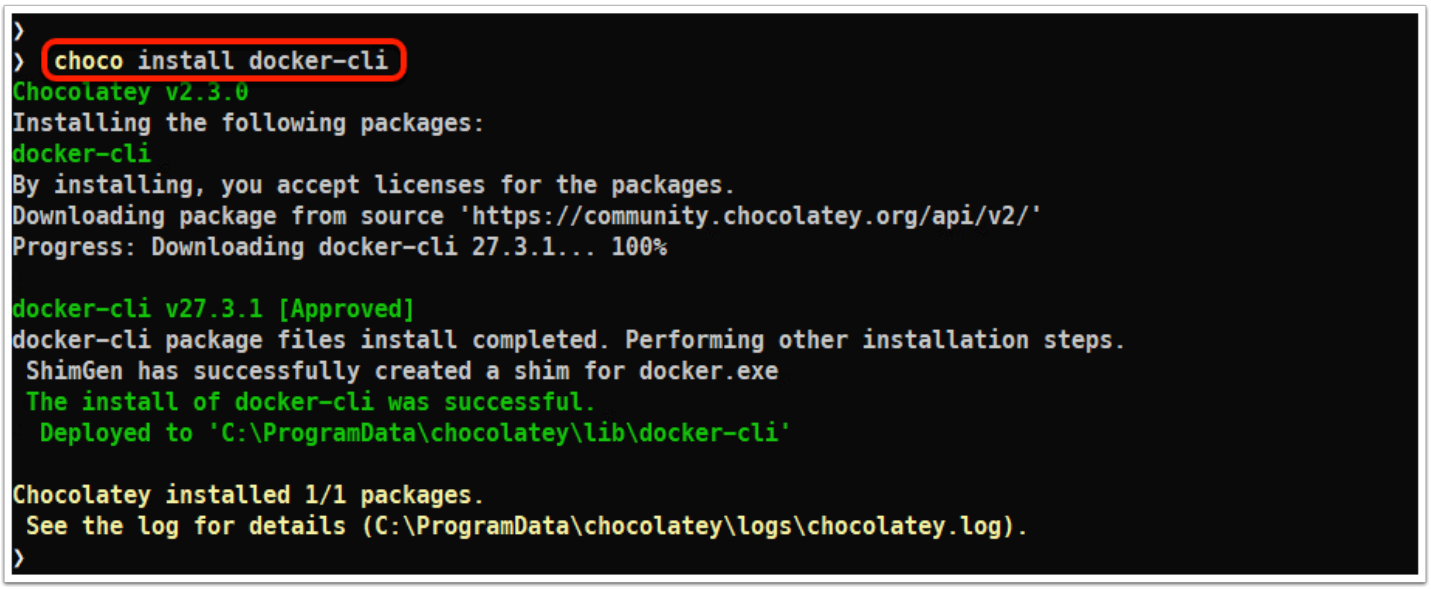


Click **OK**



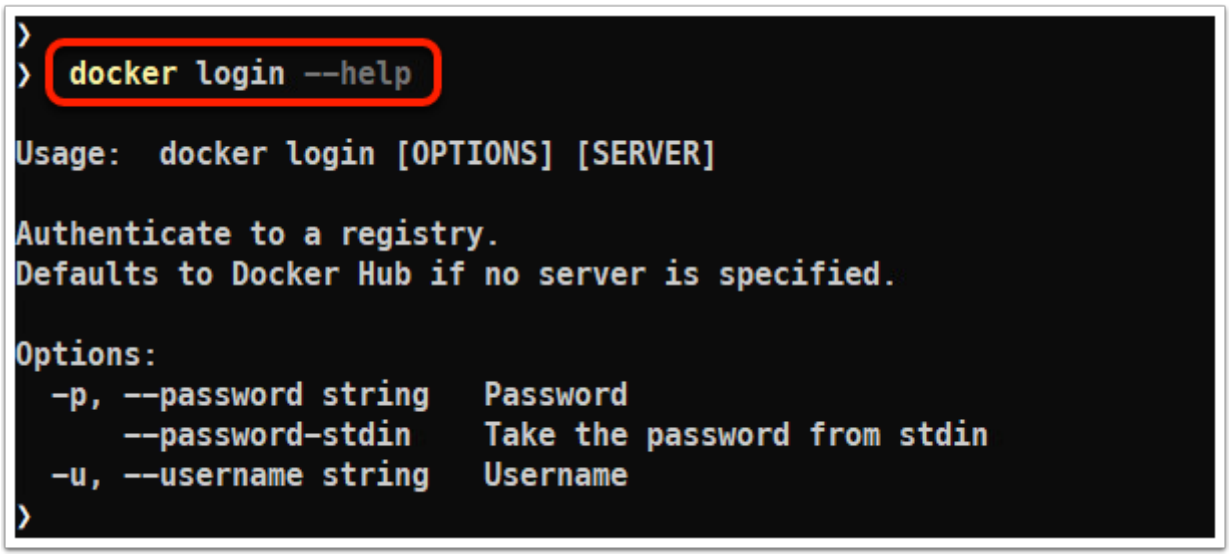
Now you have the harbor cert in place!

As a next step, we need to have Docker CLI in windows desktop / PS Terminal



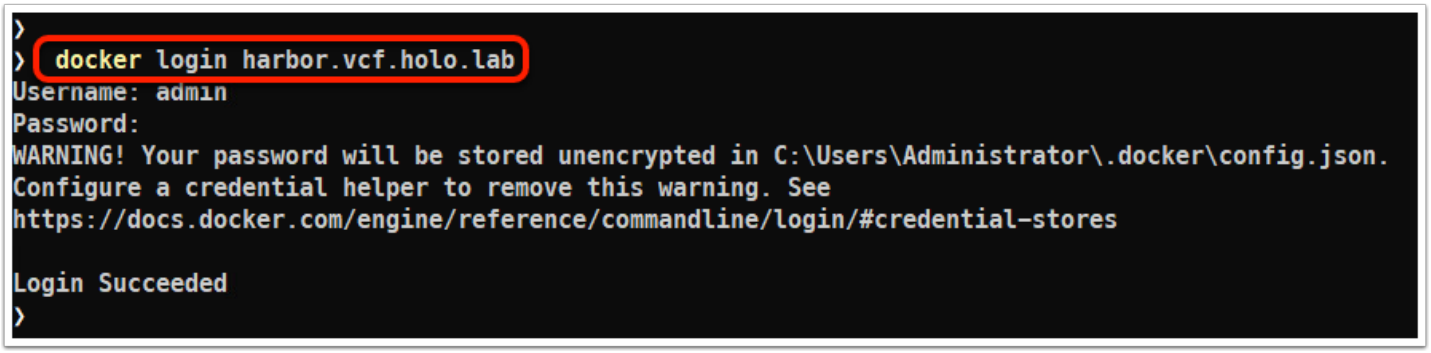

```
choco install docker-cli
```

Check it is working:




```
docker login --help
```

Okay, time to log in to our Harbor:




```
docker login harbor.vcf.holo.lab
```

username: `admin`

password: `VMware1!`

You are successfully connected to your Harbor.

You are now ready to move to the next section!

Configure Access to Developers and Deploy a Workload

Task description and objectives

As a cluster administrator you can grant cluster access to other users, such as developers.

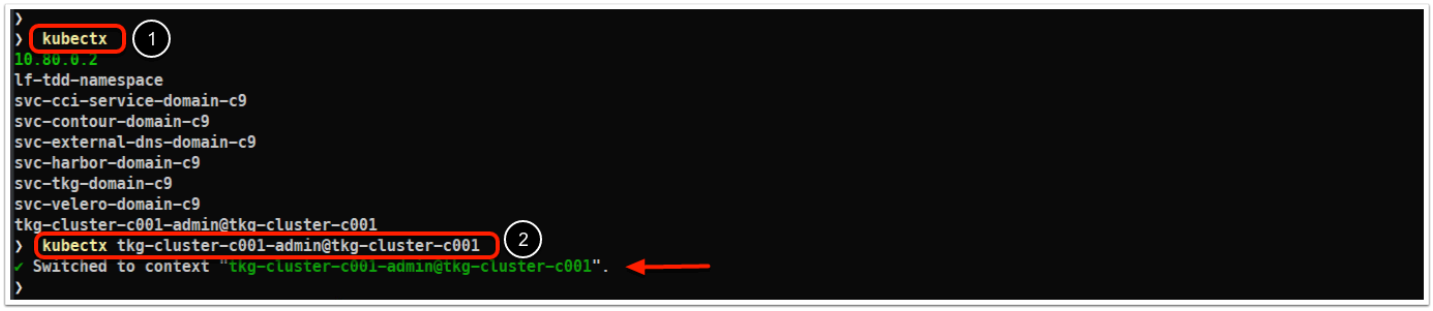
Developers can deploy pods to clusters directly using their user accounts, or indirectly using service accounts.

Your task is to establish access to Developers and Deploy a Workload.

> **IMPORTANT:**
>
> This exercise is being done with administrator@vsphere.local permissions set over the **lf-tdd-namespace.** Make sure you are logged in your **Supervisor**
>
> Your **kubeconfig** context must be set to the **tkg-cluster-c001** cluster!

Working context

Make sure you are working on the **tkg-cluster-c001** workload cluster you created and managed in previous steps




```
kubectx
```

```
kubectx tkg-cluster-c001-admin@tkg-cluster-c001
```

Configure Access to Developers

For user account authentication, Tanzu Kubernetes clusters support vCenter Single Sign-On users and groups. The user or group can be local to the vCenter Server, or synchronized from a supported directory server.

For service account authentication, you can use service tokens

To grant cluster access to developers:

You have to:

- Define a Role or ClusterRole for the user or group and apply it to the cluster.
- Create a RoleBinding or ClusterRoleBinding for the user or group and apply it to the cluster
- Prepare a Namespace in the cluster where you configure the Developer's access.

Create a Namespace for the Developer

Another aspect you need to consider, is **Pod Security Admission**

IMPORTANT: You need to consider Configuration of **PSA** for TKR 1.25 and Later

Tanzu Kubernetes releases v1.25 and later enable the **Pod Security Admission (PSA)** controller. With **PSA** you can uniformly enforce pod security using namespace labels.

The P**od Security Admission (PSA)** controller is a Kubernetes controller that lets you apply security standards to pods running on TKG clusters. By default, Tanzu Kubernetes releases v1.25 and later enable the Pod Security Admission (PSA) controller. The PSA controller replaces the Pod Security Policy (PSP) controller, which is deprecated and removed.

PSA Modes

Mode	Description
enforce	Security violations cause the pod to be rejected.
audit	Security violations trigger the addition of an audit annotation to the event recorded in the audit log, but are otherwise allowed
Warn	Security violations trigger a user-facing

Mode	Description
	warning, but are otherwise allowed

Tanzu Kubernetes release v1.25 is a transitional release with PSA configured to warn. Starting with Tanzu Kubernetes release v1.26, **PSA is enforced**. You should plan to migrate pod workloads from PSP to PSA in anticipation of upgrading TKG clusters or modify your PSA in accordance with your workloads.

PSA Standards

Level	Description
privileged	Unrestricted control, providing the widest possible level of permissions. This security standard allows for known privilege escalations.
baseline	Minimally restrictive control which prevents known privilege escalations. This security standard allows the default (minimally specified) pod configuration
restricted	Heavily restricted control, following pod hardening best practices

IMPORTANT!

By default, TKG clusters provisioned with Tanzu Kubernetes releases v1.26 and later have the **PSA** mode **enforce** set to **restricted** for **non-system namespaces**.

If a pod violates security, it is rejected. You must configure PSA on the namespace to run pods with less restrictive control.

The TKR of the **tkg-cluster-c001** you deployed is **v1.29**

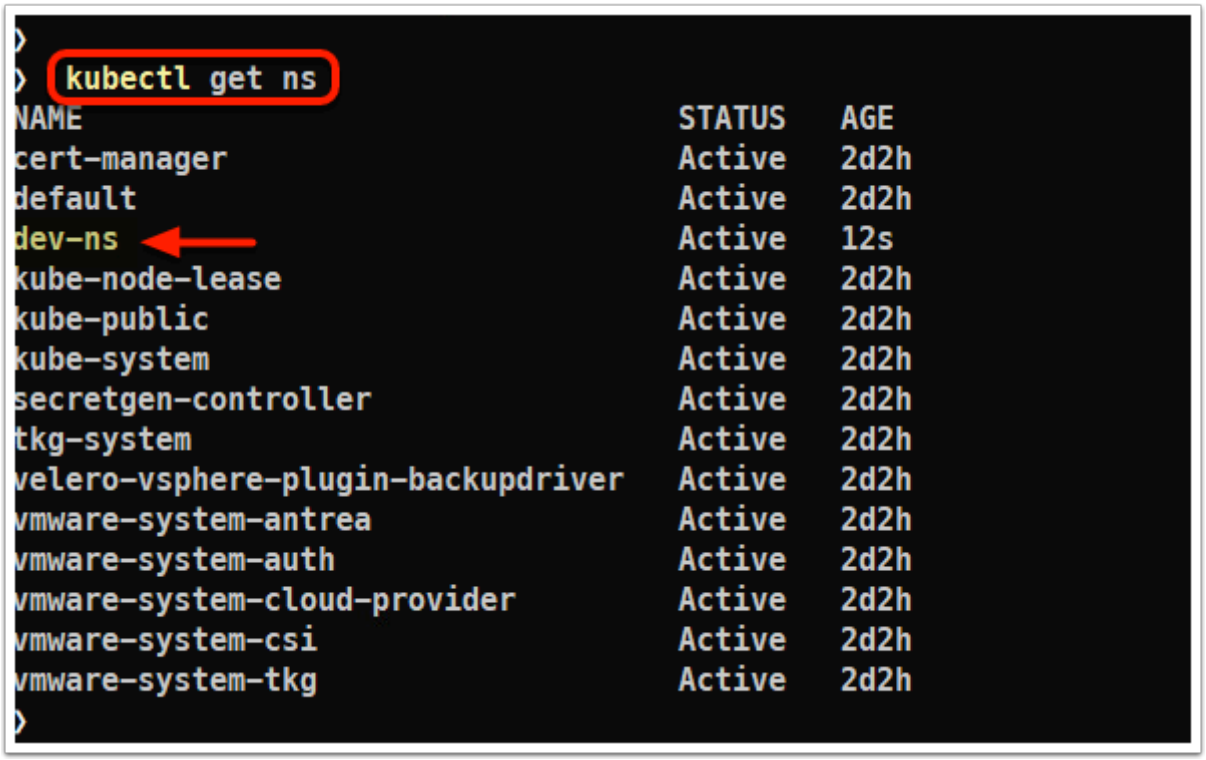
To create objects such as namespaces, you must be logged in as a user that is assigned the cluster admin role.

Create a namespace called **dev-ns** in **tkg-cluster-c001**

```
kubectl create ns dev-ns
```

Verify that the namespace was created successfully.

Create a Namespace for the Developer




```
kubectl get ns
```

However, you need to regulate the **PSA** accordingly so that the placeholder in your cluster, has the needed relaxed permissions:


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

Once the developer's namespace is ready, you can continue with establishment of role and permissions.

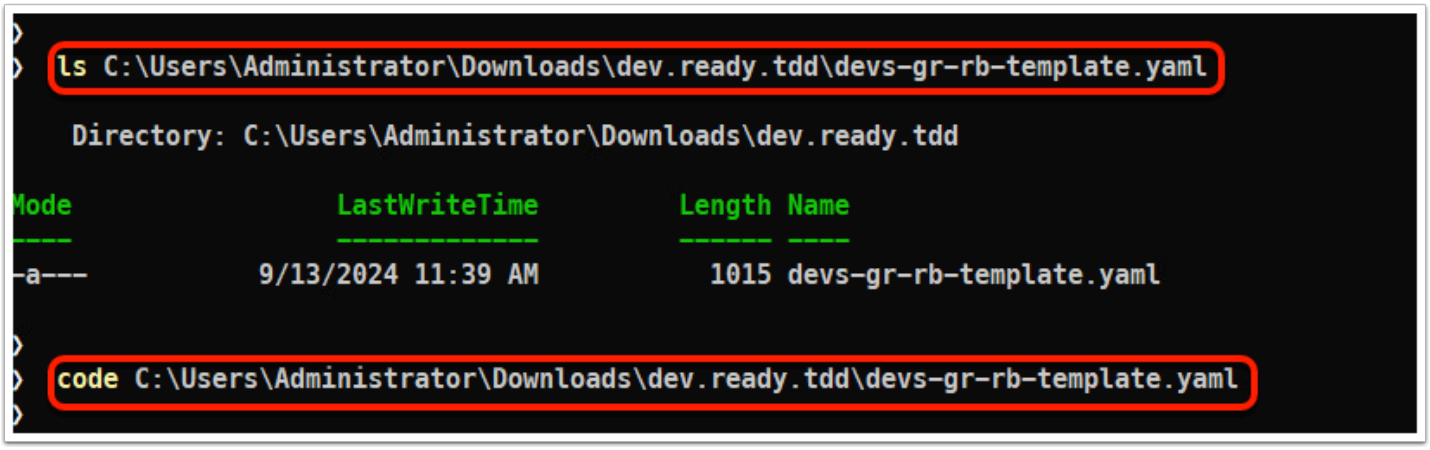
Create a RBAC for the Developer's Account

To grant access to a vCenter Single Sign-On user or group, the subject in the RoleBinding must contain either of the following values for the `name` parameter

Supported User and Group Fields

Field	Description
sso:USER-NAME@DOMAIN	For example, a local user name, such as sso:administrator@vsphere.local
sso:GROUP-NAME@DOMAIN	For example, a group name from a directory server integrated with the vCenter Server, such as sso:gg-kube-devs@vcf.holo.lab

Open the file *C:\Users\Administrator\Downloads\dev.ready.tdd**devs-gr-rb-template.yaml** in **VSCode***

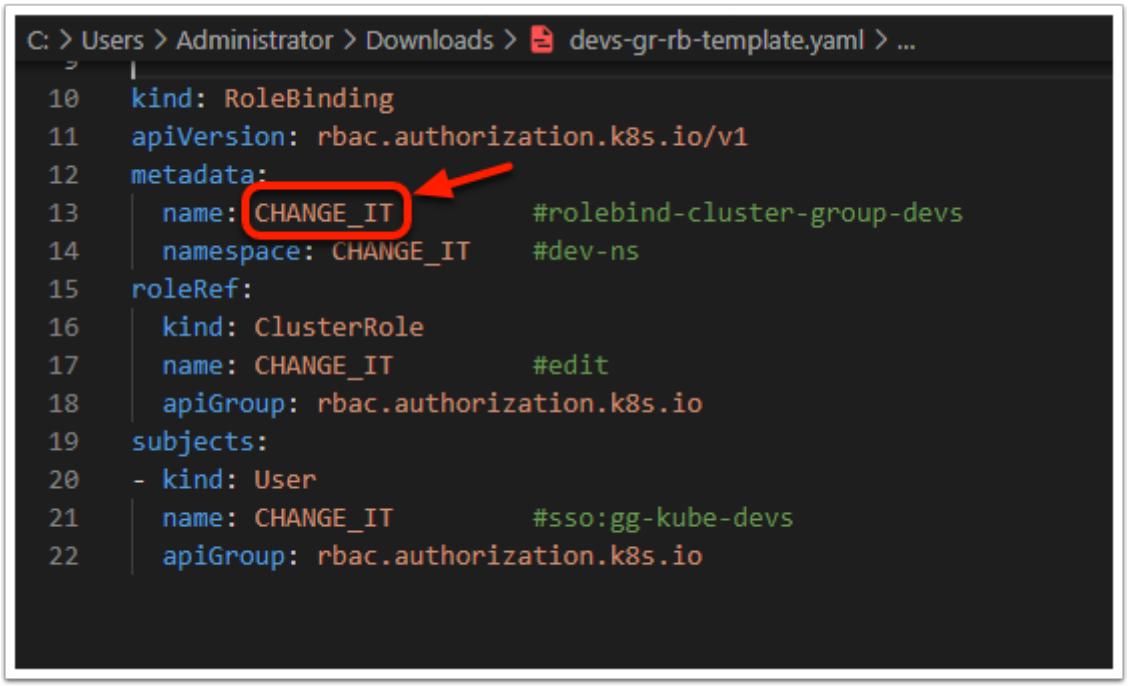



```
ls C:\Users\Administrator\Downloads\dev.ready.tdd\devs-gr-rb-template.yaml

code C:\Users\Administrator\Downloads\dev.ready.tdd\devs-gr-rb-template.yaml
```

The following template **RoleBinding** binds the vCenter Single Sign-On SSO integrated AD group named **gg-kube-devs@vcf.holo.lab** to the default ClusterRole named `edit`.

This role permits read/write access to most objects in a namespace, in this case the `dev-ns` namespace..

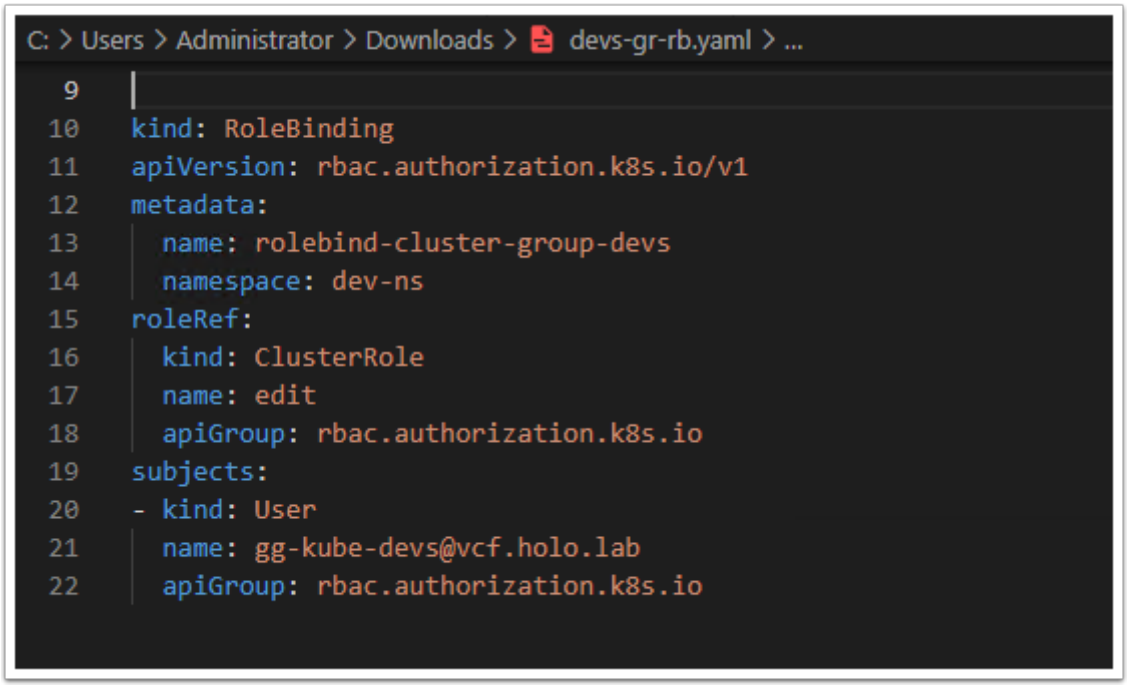
Create a RBAC for the Developer's Account



Use the **SAVE AS** in VSC and give the template manifest file a custom name: C:\Users\Administrator\Downloads**devs-gr-rb.yaml**

Please note the "**CHANGE_IT**" strings, you will need to substitute these and provide actual values

Create a RBAC for the Developer's Account

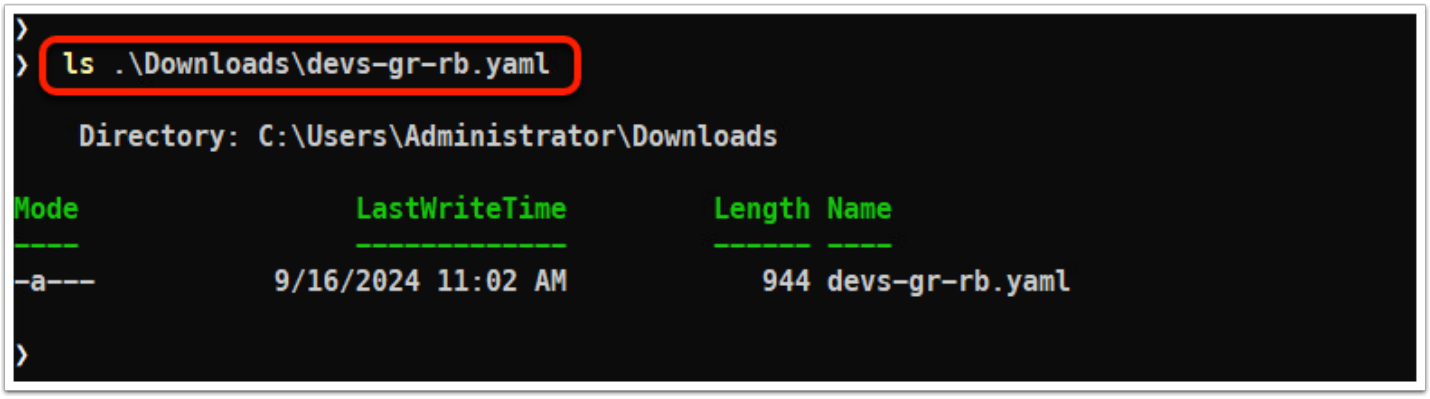


Your configuration file should look exactly as the screenshot above.

Save the file!

Double check you have the manifest in place:

Create a RBAC for the Developer's Account




```
ls .\Downloads\devs-gr-rb.yaml
```

You are now ready to implement the role in Terminal:

Create a RBAC for the Developer's Account

```
❯
❯ kubectl apply -f  .\Downloads\devs-gr-rb.yaml
rolebinding.rbac.authorization.k8s.io/rolebind-cluster-group-devs created
❯
```

```
kubectl apply -f  .\Downloads\devs-gr-rb.yaml
```

Test Permissions for the Developer

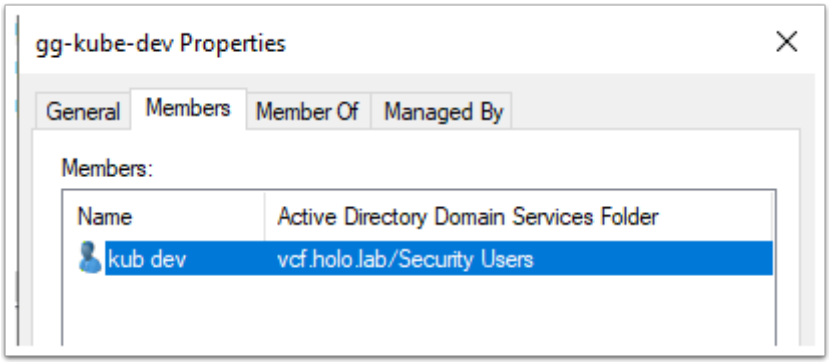
In Terminal, log out as **administrator@vsphere.local**

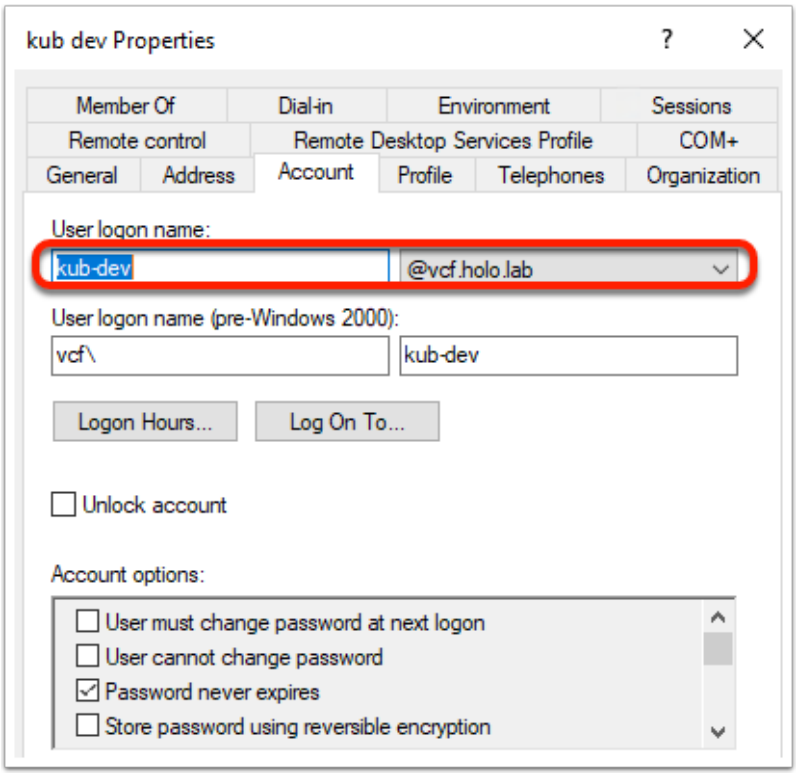
```
❯
❯ kubectl vsphere logout
Logged out of all vSphere namespaces.
❯
```

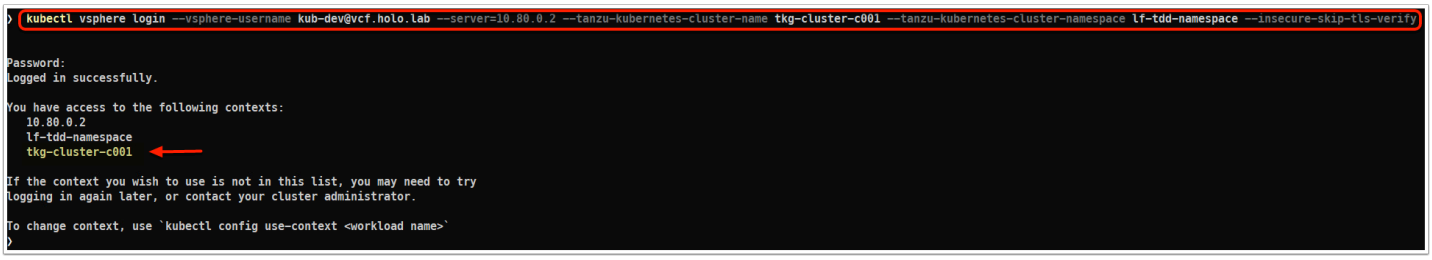
```
kubectl vsphere logout
```

Log in as an SSO user, member of the AD group you configured rolebinding earlier.

Test Permissions for the Developer








```
kubectl vsphere login --vsphere-username kub-dev@vcf.holo.lab --server=10.80.0.2 --
tanzu-kubernetes-cluster-name tkg-cluster-c001 --tanzu-kubernetes-cluster-namespace lf-
tdd-namespace --insecure-skip-tls-verify
```

username: `kub-dev@vcf.holo.lab`

password: `VMware123!`

Test the permissions of the objects in the cluster

Change working context to the workload TKG cluster

Test Permissions for the Developer

```
kubectx tkg-cluster-c001
```

Test permissions

Test Permissions for the Developer

```
❯ kubectl get ns
NAME                                STATUS   AGE
cert-manager                        Active   27h
default                             Active   28h
dev-ns                              Active   49m
kube-node-lease                     Active   28h
kube-public                         Active   28h
kube-system                         Active   28h
secretgen-controller                Active   28h
tito                                Active   25h
tkg-system                          Active   28h
velero-vsphere-plugin-backupdriver  Active   28h
vmware-system-antrea                Active   28h
vmware-system-auth                  Active   28h
vmware-system-cloud-provider        Active   28h
vmware-system-csi                   Active   28h
vmware-system-tkg                   Active   28h
```

```
kubectl get ns
```

```
❯ kubectl get nodes -o wide
NAME                                                        STATUS   ROLES           AGE   VERSION                  INTERNAL-IP   EXTERNAL-IP   OS-IMAGE             KERNEL-VERSION       CONTAINER-RUNTIME
tkg-cluster-c001-qbqkc-k2qc8                                Ready    control-plane   2d3h  v1.29.4+vmware.3-fips.1  10.244.0.66   <none>        Ubuntu 22.04.4 LTS   5.15.0-112-generic   containerd://1.6.31
tkg-cluster-c001-tkg-cluster-nodepool-c001-jcvkd-c6ksj-n2vhq  Ready    <none>          2d3h  v1.29.4+vmware.3-fips.1  10.244.0.67   <none>        Ubuntu 22.04.4 LTS   5.15.0-112-generic   containerd://1.6.31
```

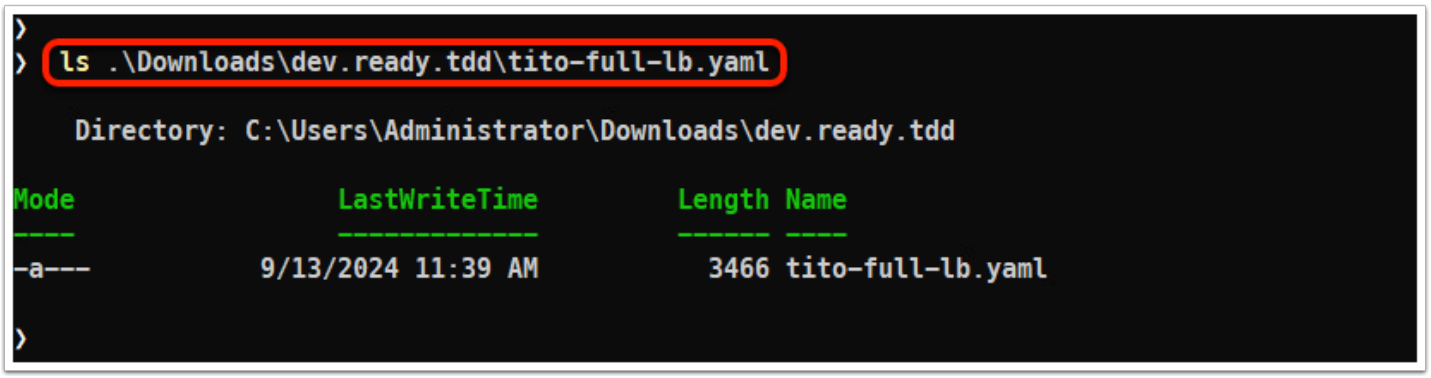
```
kubectl get nodes -o wide
```

```
❯ kubectl get pods -A
NAMESPACE                       NAME                                                                READY   STATUS      RESTARTS       AGE
cert-manager                    cert-manager-7f545657f4-tnplt                                       1/1     Running     0              2d2h
cert-manager                    cert-manager-cainjector-56564f68dd-stmvn                            1/1     Running     0              2d2h
cert-manager                    cert-manager-webhook-6855846455-qqfk4                               1/1     Running     0              2d2h
kube-system                     antrea-agent-2p8hw                                                  2/2     Running     0              2d3h
kube-system                     antrea-agent-4zgzs                                                  2/2     Running     0              2d3h
kube-system                     antrea-controller-69f8dc4689-58fn5                                  1/1     Running     0              2d3h
kube-system                     coredns-66d8b77bb7-5v4r9                                            1/1     Running     0              2d3h
kube-system                     coredns-66d8b77bb7-qpw8x                                            1/1     Running     0              2d3h
kube-system                     docker-registry-tkg-cluster-c001-qbqkc-k2qc8                        1/1     Running     0              2d3h
kube-system                     docker-registry-tkg-cluster-c001-tkg-cluster-nodepool-c001-jcvkd-c6ksj-n2vhq  1/1     Running     0              2d3h
kube-system                     etcd-tkg-cluster-c001-qbqkc-k2qc8                                   1/1     Running     0              2d3h
kube-system                     kube-apiserver-tkg-cluster-c001-qbqkc-k2qc8                         1/1     Running     0              2d3h
kube-system                     kube-controller-manager-tkg-cluster-c001-qbqkc-k2qc8                1/1     Running     0              2d3h
kube-system                     kube-proxy-2n74j                                                    1/1     Running     0              2d3h
kube-system                     kube-proxy-gq8zk                                                    1/1     Running     0              2d3h
kube-system                     kube-scheduler-tkg-cluster-c001-qbqkc-k2qc8                         1/1     Running     0              2d3h
kube-system                     metrics-server-5d7798ccbc-84p8s                                     1/1     Running     0              2d3h
kube-system                     snapshot-controller-86d9b86fb4-4mkkj                                1/1     Running     0              2d3h
secretgen-controller            secretgen-controller-6f866c4756-6rrz5                               1/1     Running     0              2d3h
tkg-system                      kapp-controller-56bd5797bd-sjb42                                    2/2     Running     0              2d3h
tkg-system                      tanzu-capabilities-controller-manager-6466f94878-7h757              1/1     Running     0              2d3h
vmware-system-antrea            register-placeholder-skjvj                                          0/1     Completed   0              2d3h
vmware-system-auth              guest-cluster-auth-svc-nztjv                                        1/1     Running     0              2d3h
vmware-system-cloud-provider    guest-cluster-cloud-provider-59758d68c9-q8x2p                       1/1     Running     0              2d3h
vmware-system-csi               vsphere-csi-controller-fb55d5746-zxl7c                              7/7     Running     0              2d3h
vmware-system-csi               vsphere-csi-node-l5xks                                              3/3     Running     0              2d3h
vmware-system-csi               vsphere-csi-node-ssc9v                                              3/3     Running     5 (2d3h ago)   2d3h
```

```
kubectl get pods -A
```

Deploy a Workload as a Developer

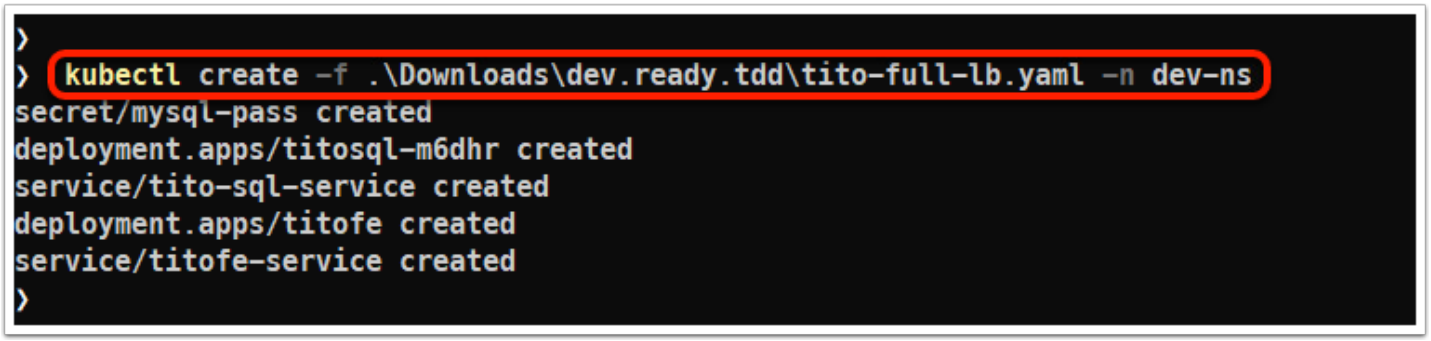
Check you have access to the manifest




```
ls .\Downloads\dev.ready.tdd\tito-full-lb.yaml
```

Deploy the TITO app in the corresponding namsespace `dev-ns` on the workload TKG cluster

Deploy a Workload as a Developer



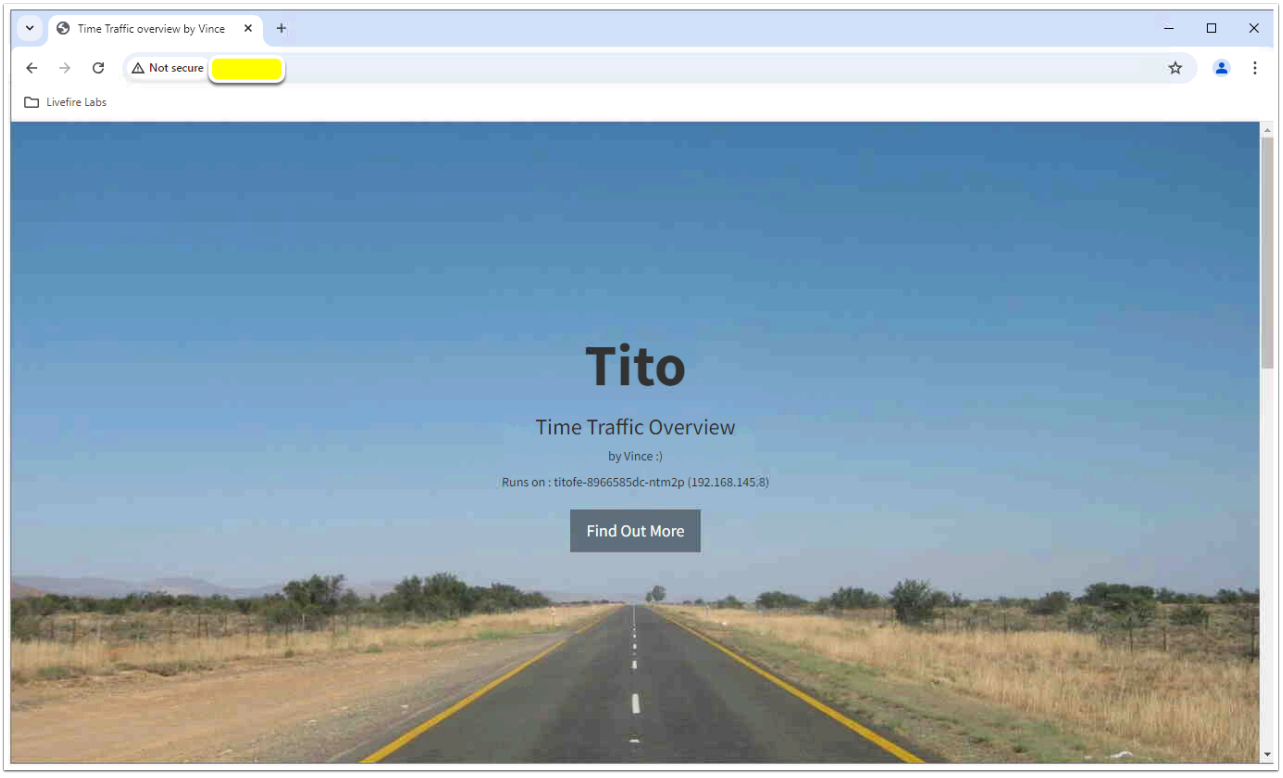

```
kubectl create -f .\Downloads\dev.ready.tdd\tito-full-lb.yaml -n dev-ns
```

In a couple of minutes you should have the app running

Deploy a Workload as a Developer

```
❯
❯ kubectl get all -n dev-ns
NAME                                 READY   STATUS    RESTARTS   AGE
pod/titofe-8966585dc-7wp9g           1/1     Running   0          2m28s
pod/titofe-8966585dc-ntm2p           1/1     Running   0          2m28s
pod/titosql-m6dhr-55b4d79fc8-k5vr6   1/1     Running   0          2m28s

NAME                          TYPE           CLUSTER-IP      EXTERNAL-IP   PORT(S)        AGE
service/tito-sql-service      ClusterIP      10.98.175.205   <none>        3306/TCP       2m29s
service/titofe-service        LoadBalancer   10.104.134.0    10.80.0.5     80:30779/TCP   2m28s

NAME                              READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/titofe            2/2     2            2           2m28s
deployment.apps/titosql-m6dhr     1/1     1            1           2m29s

NAME                                         DESIRED   CURRENT   READY   AGE
replicaset.apps/titofe-8966585dc             2         2         2       2m28s
replicaset.apps/titosql-m6dhr-55b4d79fc8     1         1         1       2m29s
❯
```

```
kubectl get all -n dev-ns
```



You can explore the deployed **Tito** application, via your browser at `http://<EXTERNAL IP FROM titofe-service LoadBanlancer>`

Post- Deployment: Troubleshooting

Post Deployment Basic Troubleshooting

1. Task description and objectives

As part of initial troubleshooting you are tasked to perform a thorough health-check of the environment components, to ensure that no issues exist

2. Post Deployment Initial Troubleshooting

You need to:

Collect and inspect Supervisor logs

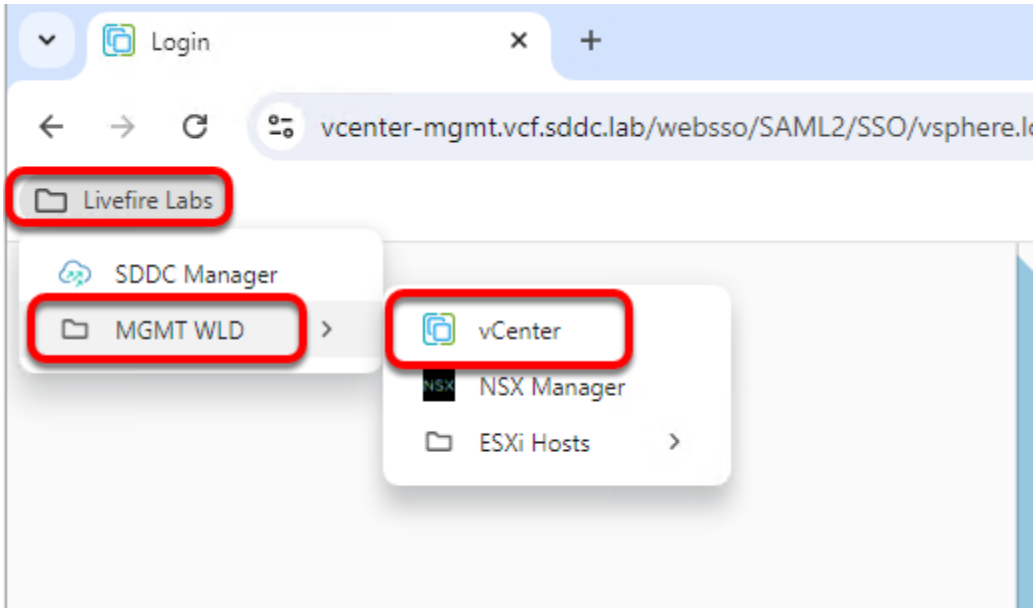
Collect and inspect ClusterAPI logs

Inspect the health of Supervisor's TKG components

Verify SSH access to Supervisor

2.1. Get the Supervisor Logs

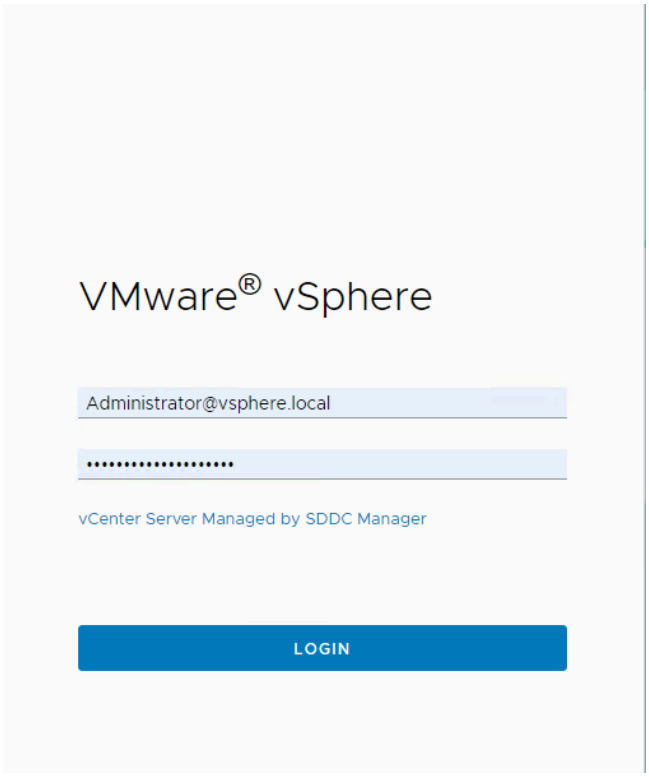
Add text



Open Chrome

Open the Livefire Labs Bookmark Folder

Choose --> **MGMT WLD** --> **vCenter**



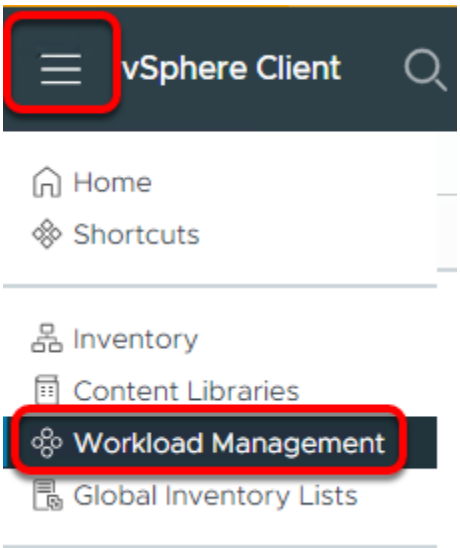
Login:

User:

```
Administrator@vsphere.local
```

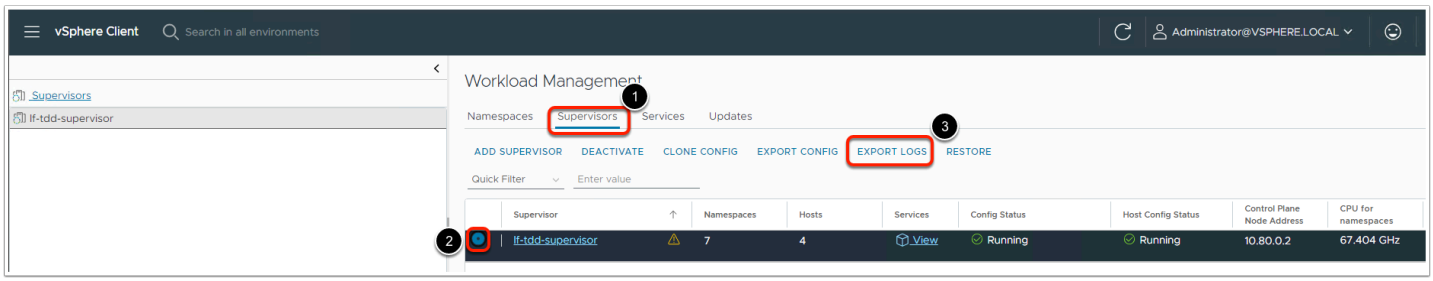
Password:

```
VMware123!VMware123!
```

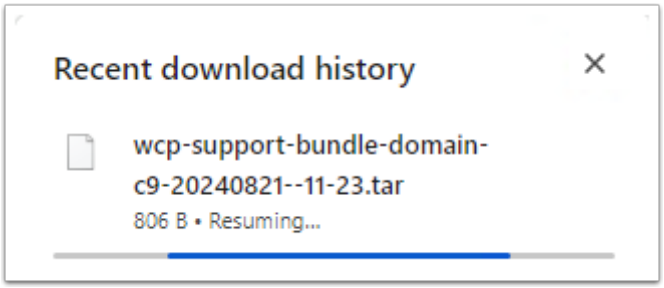


Open the elipses Menu icon

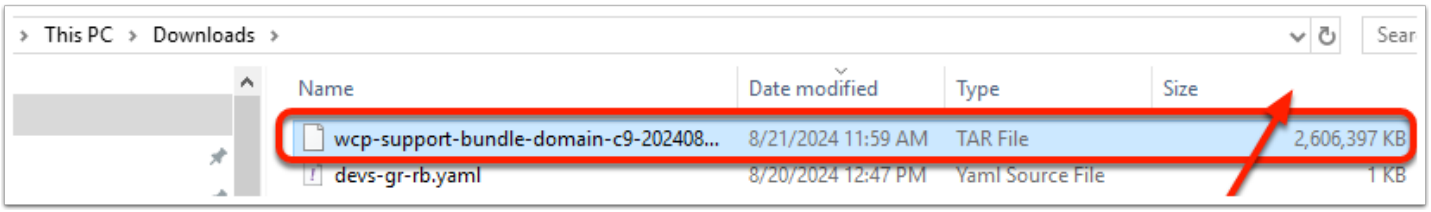
Choose **Workload Management**



1. Click the **Supervisors** tab
2. Select the **lf-tdd-supervisor**
3. Click **EXPORT LOGS**



The logs are downloaded to the **Downloads** folder



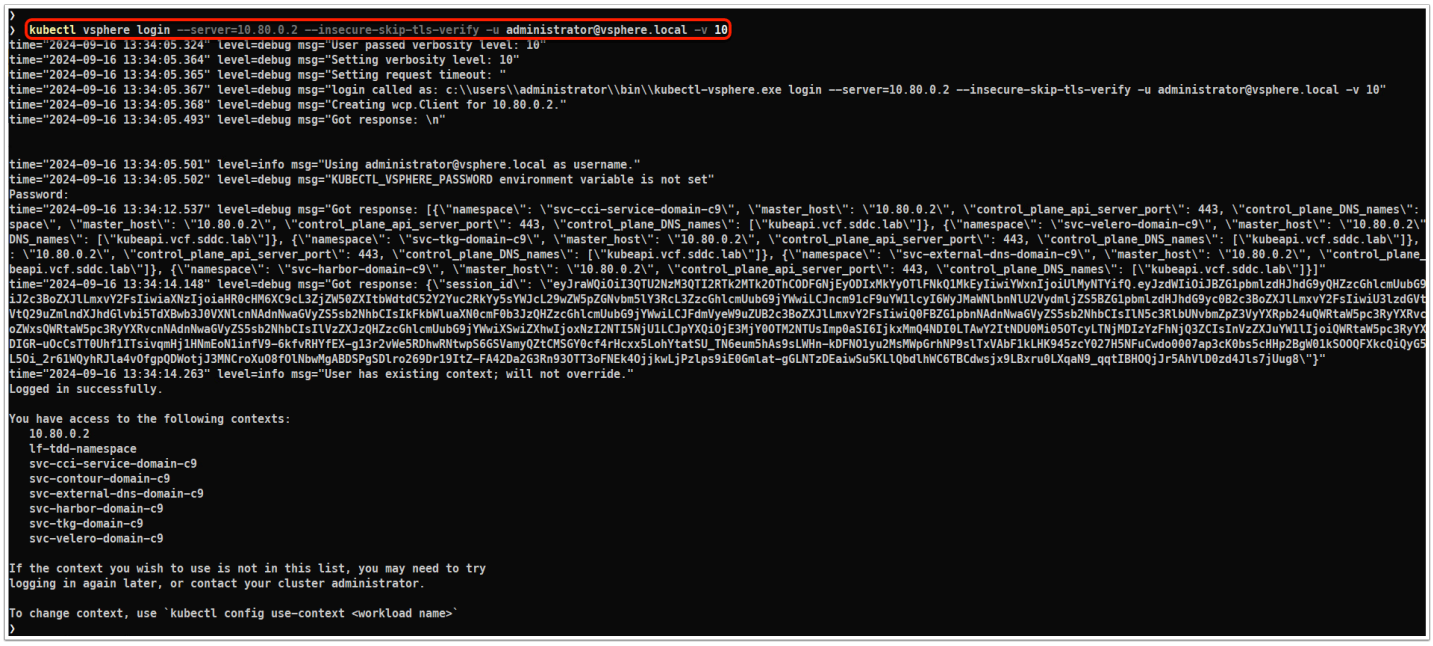
IMPORTANT !!!

This procedure takes significant time, as the logs are being extracted AND compressed at the backend in a TAR file, which comes with a relatively large size!

You don't need to wait for its completion, just move on to the next exercise

2.2. Get the ClusterAPI Logs

Use CLI and log in to Supervisor

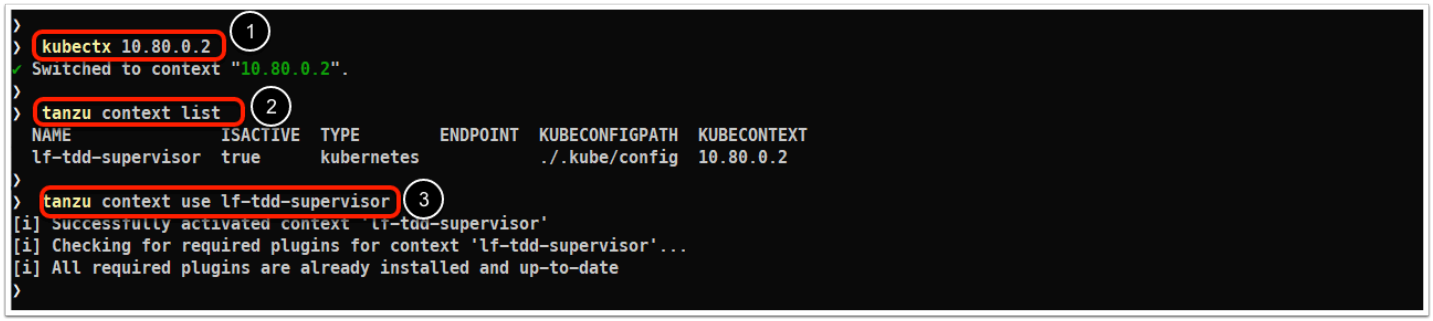



```
kubectl vsphere login --server=10.80.0.2 -u administrator@vsphere.local --insecure-skip-tls-verify -v 10
```

password: `VMware123!VMware123!`

This command passes the UserID **-u administrator@vsphere.local** as well as passing verbose level 10 to give us more debugging info (**-v 10**) in case we need it.

As the vCenter root certificate is not installed on your windows desktop machine, the **--insecure-skip-tls-verify** flag is required when you log in to the Supervisor

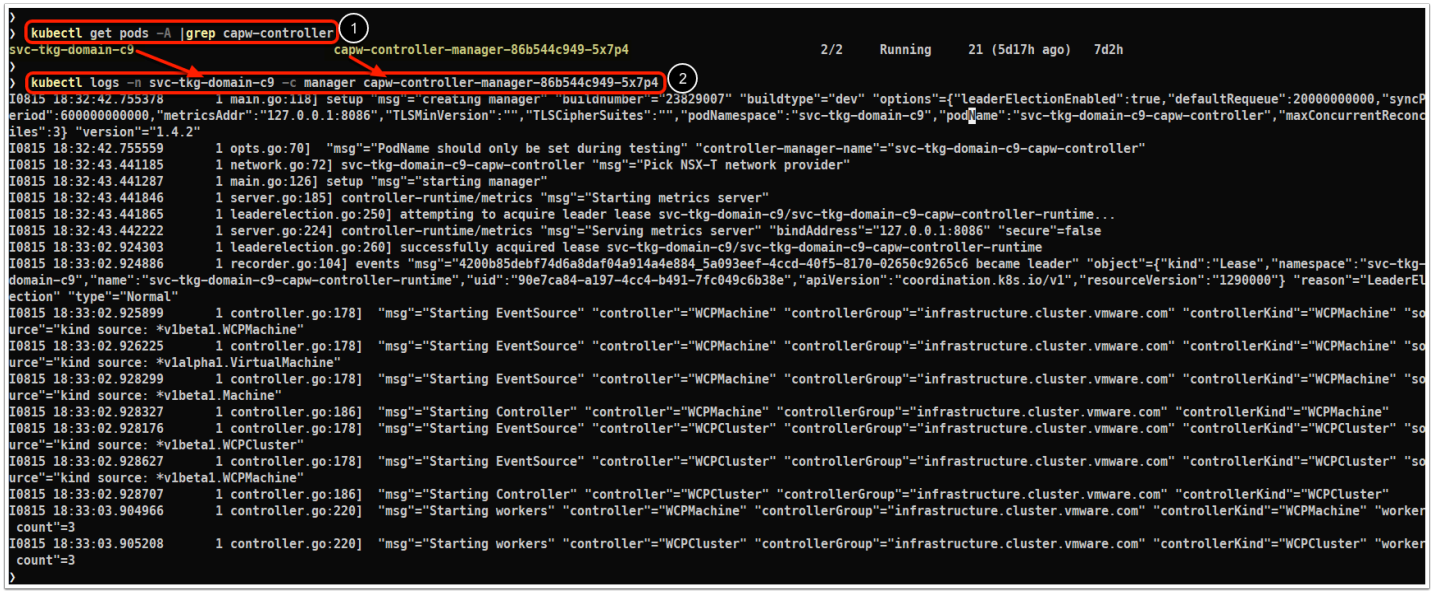



```
kubectx 10.80.0.2
```

```
tanzu context list
```

```
tanzu context use lf-tdd-supervisor
```

Examine the ClusterAPI logs



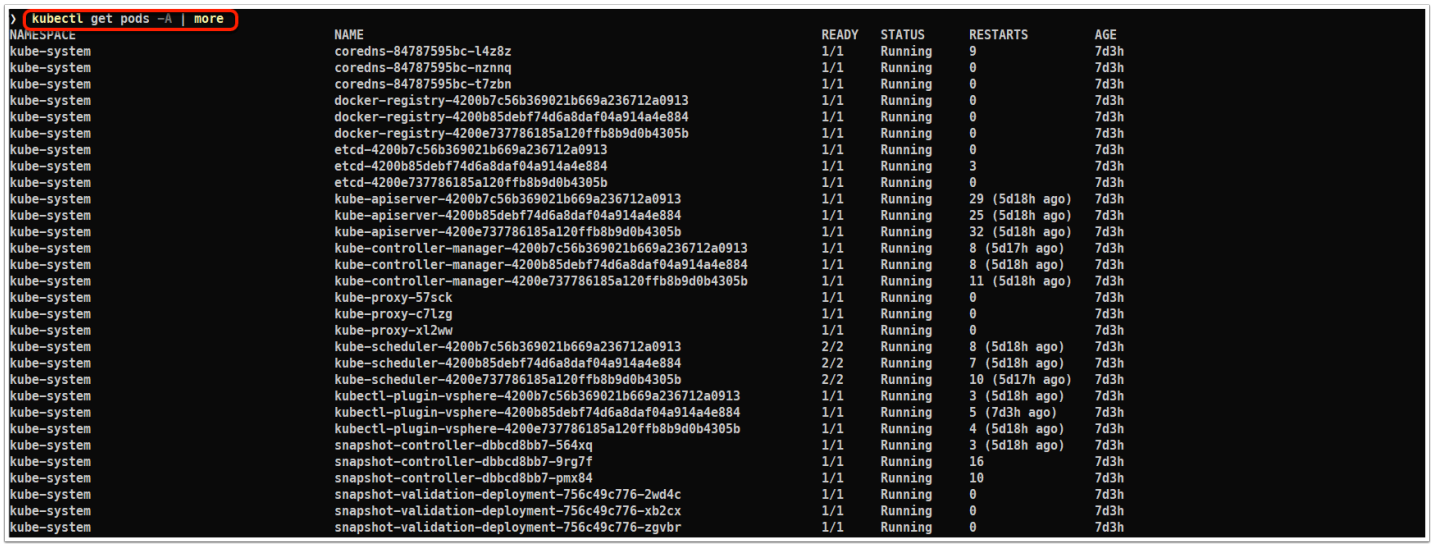
1. Get the **name** of the `capw-controller-manager` **pod** and the **namespace** it is running in
2. Get the logs from it

```
kubectl get pods -A |grep capw-controller
```

```
kubectl logs -n svc-tkg-domain-c9 -c manager capw-controller-manager-86b544c949-5x7p4
```

2.3. Get the Health Status for TKG Components on the Supervisor

You can get the Health Status of TKG components simply inspecting if all pods are in Running state




```
kubectl get pods -A | more
```

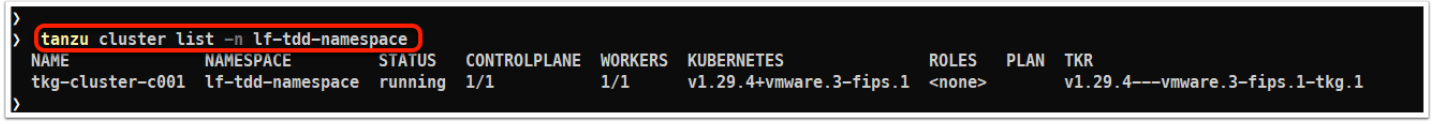
Pressing the space bar on your keyboard will pass the information to you on portions.

However, you can simply filter out the pods that are NOT in a Running state:


```
kubectl get pods -A | grep -v Running
```

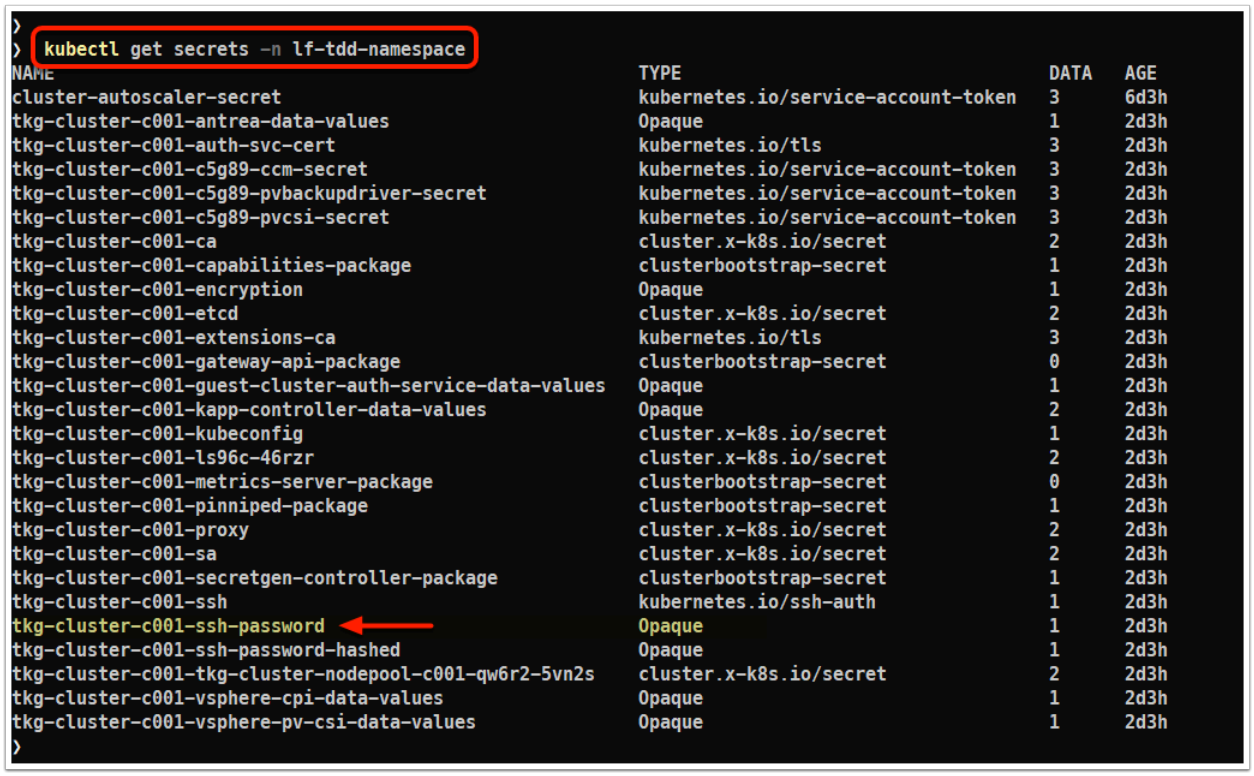
2.4. Get the password for SSH connection

Let's check which TKG workload clusters are available in the relevant namespace




```
tanzu cluster list -n lf-tdd-namespace
```

Use the terminal to identify the secret that contains the SSH password for the cluster nodes




```
kubectl get secrets -n lf-tdd-namespace
```

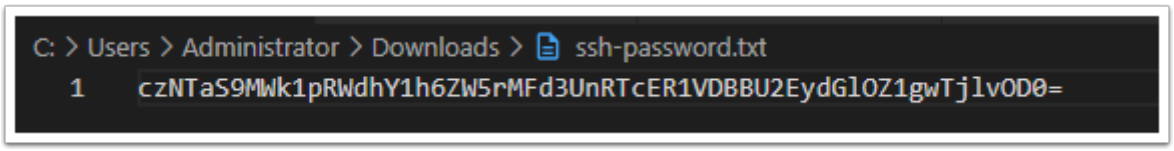
That sohuld be `tkg-cluster-c001-ssh-password`

Now you can retrieve the `ssh-passwordkey` value from the secret:


```
kubectl get secrets -n lf-tdd-namespace tkg-cluster-c001-ssh-password -o yaml | grep
"ssh-passwordkey"
```

Record the value in VCD and save it like C:\Downloads**ssh-password.txt** file

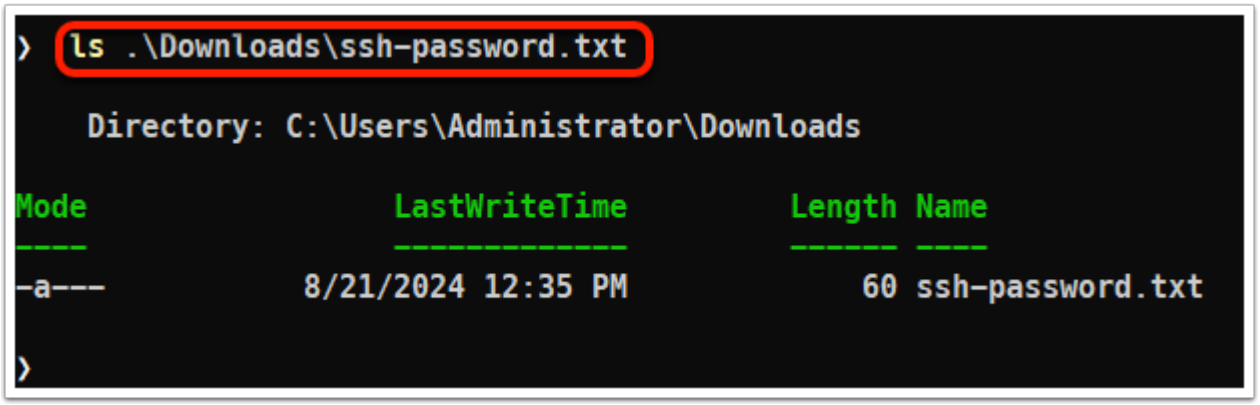

```
code C:\Users\Administrator\Downloads\ssh-password.txt
```



Just as an example:

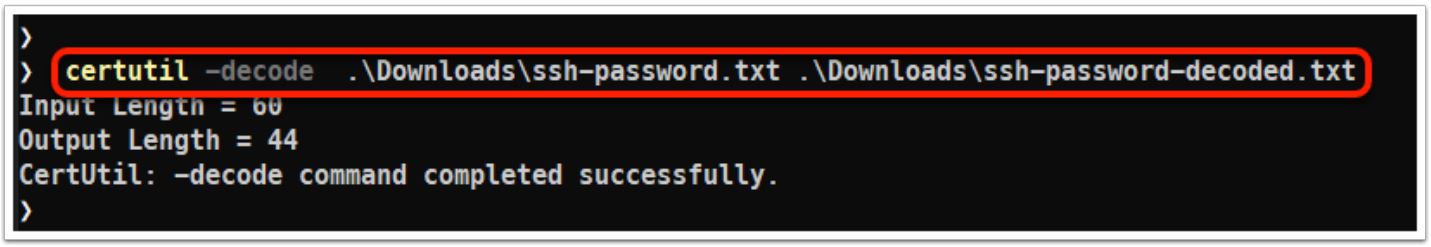
ssh-passwordkey:
czNTaS9MWk1pRWdhY1h6ZW5rMFd3UnRTcER1VDBBU2EydGlOZ1gwTjlvOD0=

Double check you the file is saved:




```
ls .\Downloads\ssh-password.txt
```

Decode this password in an output file

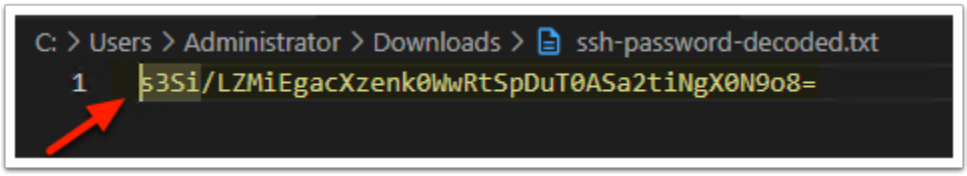



```
certutil -decode .\Downloads\ssh-password.txt .\Downloads\ssh-password-decoded.txt
```

Open the decoded password in VCD


```
code .\Downloads\ssh-password-decoded.txt
```

Okay, you have the SSH password now:



Next, you can get the IP addresses of the TKG workload cluster nodes:


```
kubectl get vspheremachines -n lf-tdd-namespace -l cluster.x-k8s.io/cluster-name=tkg-cluster-c001 -o wide
```

You can SSH to the target cluster node as the `vmware-system-user` with the password you decoded in previous steps.

2.5. Create a Linux Jump Host VM

To SSH a workload cluster nodes using a password, you can create a Jump Box VM that connects to the workload network and the the management of the frontend network for SSH tunneling.

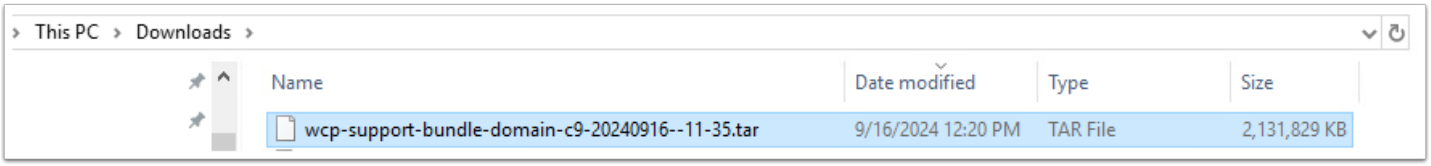
The instructions use PhotonOS which you can download here: https://github.com/vmware/photon/wiki/Downloading-Photon-OS.

This exercise won't take place in the current TDD Lab session.

2.6. End of the chapter

By now, the supervisor logs from section 2.1 should got downloaded :-)

You can check and see the TAR file



This is the end of your Lab activities!