Deployment Guide–Nov 2021

Deploy an AI-Ready Enterprise Platform on VMware vSphere 7 with VMware Tanzu Kubernetes Grid Service

An End-to-end Approach to the Setup of the Machine Learning Infrastructure

Table of Contents

Introduction

Overview

This deployment guide illustrates the prerequisites and processes for setting up the newly introduced AI-Ready Enterprise Platform from NVIDIA and VMware. It includes NVIDIA Enterprise AI Suite, VMware vSphere® 7 Update 3a and VMware vCenter® 7 Update 3a, and vSphere with VMware Tanzu® Kubernetes Grid™ Service.

This document describes two deployment scenarios:

- Deploy a Tanzu Kubernetes Grid Cluster or stand-alone VM with vGPU. This is a basic scenario for users to start with to run AI workloads with minimal configurations on vSphere with Tanzu. The Tanzu Kubernetes Grid Cluster worker nodes are configured with a traditional VMXNET3 virtual network adapter and all the advanced vSphere features including VMware vSphere vMotion® (vSphere vMotion), VMware vSphere High Availability (vSphere HA), VMware vSphere Distributed Resource Scheduler™ (DRS).
- Deploy a Tanzu Kubernetes Grid Cluster or stand-alone VM with both vGPU and RoCE backed passthrough NIC. This is an advanced scenario that requires additional hardware and configuration steps. The Tanzu Kubernetes Grid Cluster worker nodes are configured with the RoCE backed passthrough NIC to enhance the network performance between nodes to improve the distributed Machine Learning performance; however, it does not support vSphere vMotion as a tradeoff.

AI-Ready Platform from NVIDIA and VMware

VMware and NVIDIA have partnered together to unlock the power of AI for every business, by delivering an end-to-end enterprise platform optimized for AI workloads. This integrated platform delivers best-in-class AI software, the NVIDIA Enterprise AI suite, optimized and exclusively certified to run on the industry’s leading virtualization platform, VMware vSphere, with industry-leading accelerated servers that have been NVIDIA-Certified. This platform accelerates the speed at which developers can build AI and high-performance data analytics for their business, enables organizations to scale modern workloads on the same VMware vSphere infrastructure they have already invested in, and delivers enterprise-class manageability, security, and availability.

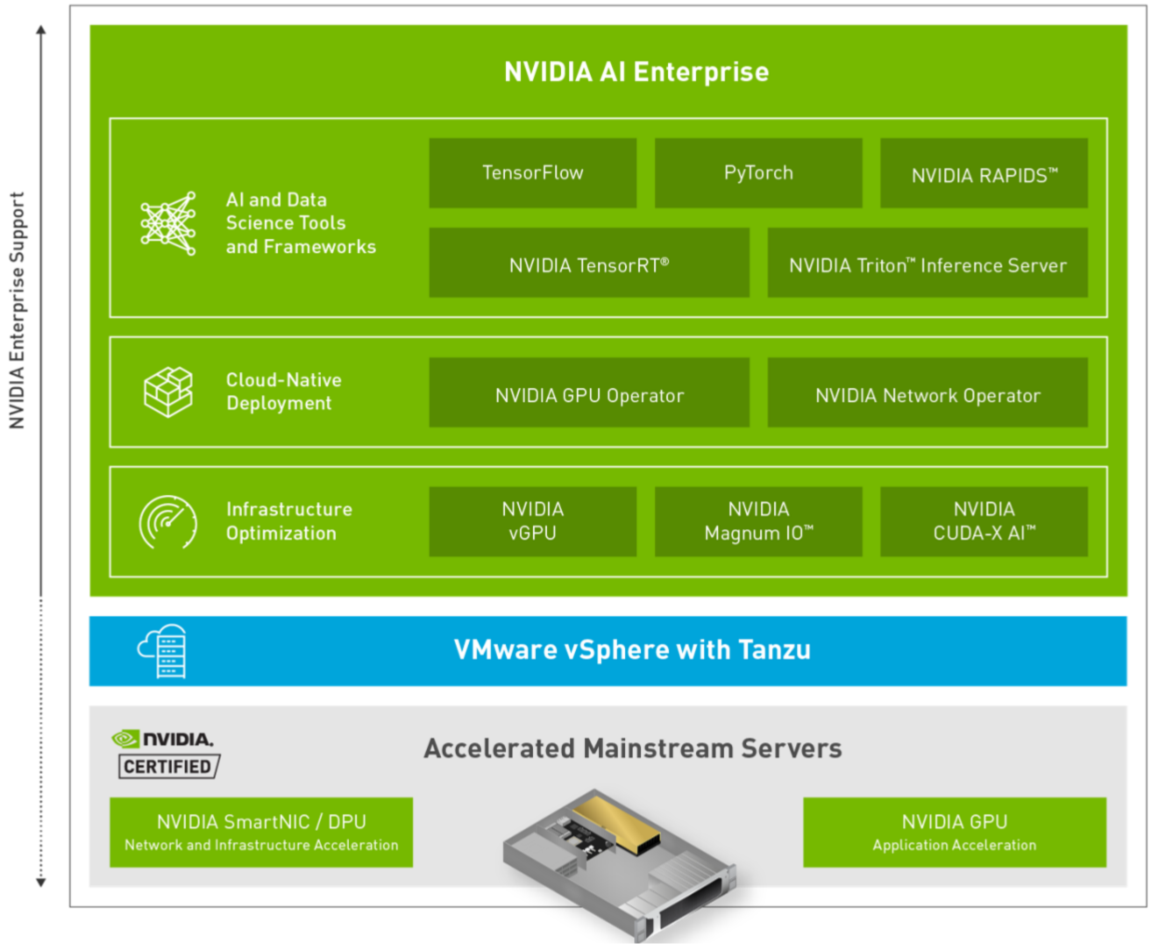


Figure 1. AI-Ready Enterprise Platform

Intended Audience

The primary audience for this document is the VMware administrators who are tasked with setting up a GPU-based VMware vSphere with Tanzu Kubernetes Grid Service for the machine learning practitioners, the data scientists, or application architects. The document is also targeted at the developers who will be using the Tanzu Kubernetes Grid Service contained in vSphere with Tanzu to create a Tanzu Kubernetes Grid Cluster or stand-alone VM with vGPU and the IT Manager in charge of such a data center as well as the application architects who need to understand the infrastructure to support a working and virtualized machine learning application.

End-User Goals

The goals of a data center project team that would use this document can be broken down into the following main areas:

- To set up an artificial intelligence or machine learning (AI/ML) infrastructure on vSphere with Tanzu Kubernetes Grid Service with virtual GPUs associated with one or more worker nodes or stand-alone VMs in Tanzu Kubernetes Grid Cluster.
- In a more advanced use case, to create a Tanzu Kubernetes Grid Cluster or stand-alone VM with vGPU and RoCE backed passthrough NIC configured to enhance the performance of distributed Machine Learning.

VMware vSphere 7 Update 3a

To help customers deliver AI and developer-ready infrastructure, VMware has enhanced the Virtual Machine (VM) Service that was *introduced in Update 2a* in April, 2021. With vSphere 7 Update 3a, developers and DevOps teams will be able to use Kubernetes commands to provision VMs on hosts or Tanzu Kubernetes Grid Clusters with vGPUs. This will help customers build and run their AI apps on GPU-enabled hardware using a self-service model. When combined with all the GPU-related enhancements we *introduced in vSphere 7 Update 2*, customers will have a lot of power at their fingertips.

NVIDIA AI Enterprise

The NVIDIA AI Enterprise Suite is an end-to-end, cloud-native, suite of AI and data science applications and frameworks, optimized and exclusively certified by NVIDIA to run on VMware vSphere with NVIDIA-Certified Systems. It includes key enabling technologies and software from NVIDIA for rapid deployment, management, and scaling of AI workloads in the modern hybrid cloud. NVIDIA Enterprise AI is licensed and supported by NVIDIA. For more information, see the *NVIDIA AI Enterprise* webpage.

NVIDIA vGPU

NVIDIA vGPU enables multiple VMs to have simultaneous, direct access to a single physical GPU, or GPUs can be aggregated within a single VM. By doing so, NVIDIA vGPU provides VMs with high-performance compute, application compatibility, cost-effectiveness, and scalability since multiple VMs can be customized to specific tasks that may demand GPU compute or memory. For more information, see the *NVIDIA vGPU* webpage.

NVIDIA Multi-Instance GPU (MIG)

The new NVIDIA Multi-Instance GPU (MIG) feature allows the NVIDIA A100 or A30 GPUs, based on the NVIDIA Ampere architecture, to be securely partitioned into up to seven separate GPU compute slices for Compute Unified Device Architecture (CUDA) applications, providing multiple users with separate GPU resources for optimal GPU utilization. This feature is particularly beneficial for workloads that do not fully saturate the GPU's compute capacity and therefore users may want to run different workloads in parallel to maximize utilization.

MIG allows multiple vGPUs (and thereby VMs) to run in parallel on a single A100 or A30, while preserving the isolation guarantees that vGPU provides. For more information about GPU partitioning using vGPU and MIG, refer to the *technical brief*.

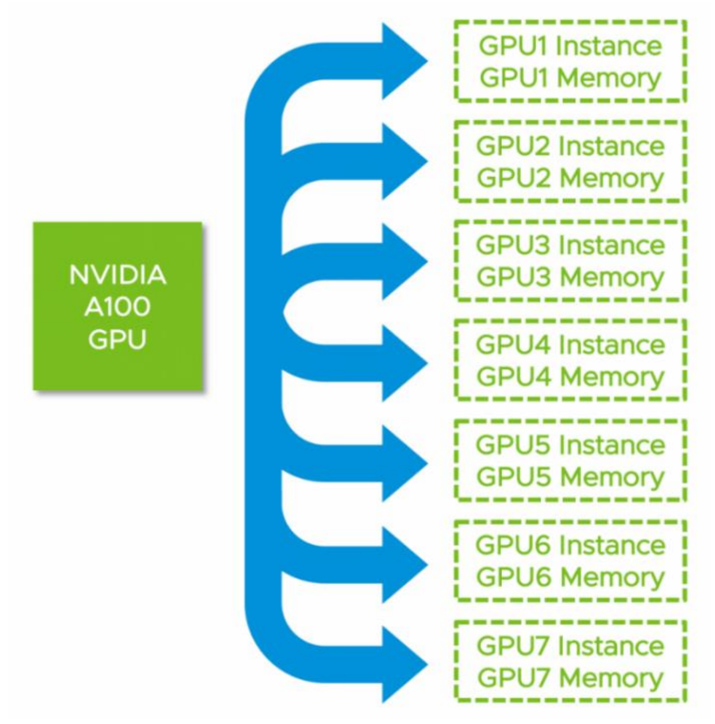

NVIDIA A100 GPU
GPU1 Instance GPU1 Memory
GPU2 Instance GPU2 Memory
GPU3 Instance GPU3 Memory
GPU4 Instance GPU4 Memory
GPU5 Instance GPU5 Memory
GPU6 Instance GPU6 Memory
GPU7 Instance GPU7 Memory

Figure 2. NVIDIA A100 Multi-Instance GPU

For more information, see *Multi-Instance GPU*.

Multi-Node (Distributed) Learning

Many data scientists are looking to scale compute resources to reduce the time it takes to complete the training of a neural network and produce results in real-time. Taking a Multi-GPU approach brings scientists closer to achieving a breakthrough as they can more rapidly experiment with different neural networks and algorithms. To maximize the throughput, multi-node learning takes the approach by using two or more worker nodes of Tanzu Kubernetes Grid Cluster on different VMware ESXi hosts with GPU installed, with each worker node assigned a virtual GPU. These worker nodes can transfer data across a network with TCP or Remote Direct Memory Access (RDMA) protocol.

Deploy a Tanzu Kubernetes Grid Cluster with vGPU Access

Deployment Prerequisites

Hardware

Server

Minimum three servers that are approved on both the *VMware Hardware Compatibility List* and the *NVIDIA Virtual GPU Certified Servers List* is needed.

GPU

Minimum one NVIDIA GPU installed in one of the servers:

- Ampere class GPU (A100, A40, A30, or A10) (A100 and A30 are MIG capable, recommended)
- Turing class GPU (T4)
- More supported GPUs can be found *here*

Software

Download the Software

- VMware vSphere Hypervisor (Six) 7.0 Update 3, *ensure to patch the 7.0 Update 3a to avoid potential PSOD*
- *VMware vCenter Server 7.0 Update 3a*
- *NVIDIA AIE ESXi Driver VIB file 470.63 or NVIDIA vGPU software 13.1 (or later version for VMware vSphere 7.0)*

Obtain the Licenses

Obtain vSphere Licenses

Connect with your NVIDIA/VMware Account Manager or go *here* to obtain the VMware vSphere licenses.

Obtain NVIDIA vGPU Licenses:

Follow this *link* to find the suitable approach to obtain the NVIDIA vGPU license.

Environment Preparation

IP Addresses Preparation

Table 1. IP Addresses

Purpose	Number of IP Addresses to Prepare
vCenter	1
vSphere Hypervisor Management	1 per Server
vSphere Hypervisor vMotion	1 per Server
NVIDIA License Server	1
vSphere vSAN (OPTIONAL)	1 per Server

vSphere Environment Preparation

- VMware vCenter Server 7.0 Update 3a or later is configured to manage the ESXi servers. We recommend running it as a VM on a separate host to the above three hosts that will form the vSphere with Tanzu Supervisor cluster. For the vCenter Server installation procedures and details, see *About vCenter Server Installation and Setup*.
- VMware vSphere Hypervisor 7.0 Update 3 or later is installed on each of the servers. For the Hypervisor installation procedures and details, see *About VMware ESXi Installation and Setup*.
- Furthermore, *About VMware vCenter Server and Host Management* provides information about vCenter and ESXi server management and configuration.
- Also, we recommend using VMware vSAN as the first-tier storage for VM placement. For more information, see *About vSAN Planning and Deployment*.

NTP Server

An NTP Server should be used to synchronize the time for vCenter, Hypervisor Server(s), and NVIDIA License Server. For details of configuring NTP Server on vCenter and Hypervisor Server, see *Configure the System Time Zone and Time Synchronization Settings* and *Editing the Time Configuration Settings of a Host*.

If there is no NTP Server available, those links above also have the information about configuring time and time zone manually.

NVIDIA License Server

If the ***Cloud License Service*** hosted by NVIDIA is not an option, the NVIDIA License Server could be installed on the servers above as a VM or could be configured on a separate physical host. Follow the instruction *here* to set up the NVIDIA Delegated License Service (DLS) License Server.

System Requirements for Setting vSphere with Tanzu

There are three options to configure the network of vSphere with Tanzu, each of them has different system and network requirements. We follow one of the following items that are applicable to the configuration as a requirement:

- *System Requirements for Setting Up vSphere with Tanzu with NSX-T Data Center*
- *System Requirements for Setting Up vSphere with Tanzu with vSphere Networking and HA Proxy Load Balancer*
- *System Requirements for Setting Up vSphere with Tanzu with vSphere Networking and NSX Advanced Load Balancer*

Hardware Settings

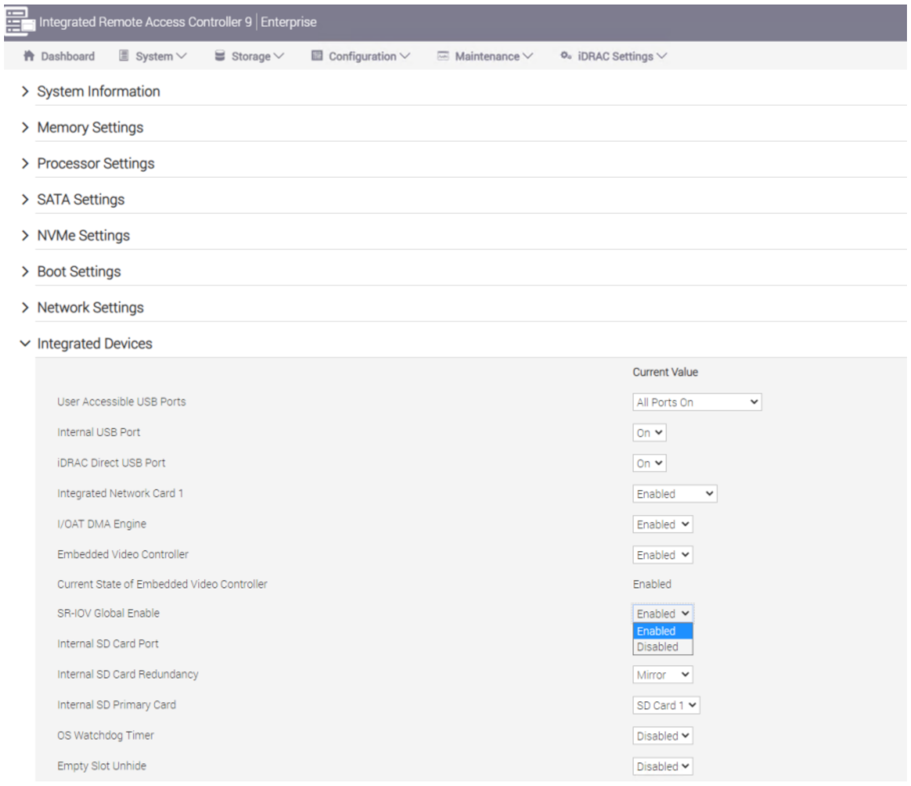
Enable vGPU Based on *Ampere Architecture GPU*

To use a vGPU based on the Ampere architecture GPU, VT-D/IOMMU and Global SR-IOV are required to be enabled. The VT-D/IOMMU setting, also known as Virtualization Technology when applied to processor settings in the server's BIOS, is also a *prerequisite to run 64-bit VM on vSphere*.

A GPU Instance in a MIG setup is associated with an SR-IOV Virtual Function at vGPU boot-up time.

Global SR-IOV

To enable the "Global SR-IOV" feature at the vSphere host BIOS level through the server management console, or iDRAC, below is an image of how this is done in the case of a Dell R740xd server, using the iDRAC interface. This may be different on your host servers, depending on your vendor.



	Current Value
User Accessible USB Ports	All Ports On
Internal USB Port	On
iDRAC Direct USB Port	On
Integrated Network Card 1	Enabled
I/OAT DMA Engine	Enabled
Embedded Video Controller	Enabled
Current State of Embedded Video Controller	Enabled
SR-IOV Global Enable	Enabled [✓ Enabled] Disabled
Internal SD Card Port	
Internal SD Card Redundancy	Mirror
Internal SD Primary Card	SD Card 1
OS Watchdog Timer	Disabled
Empty Slot Unhide	Disabled

Figure 3. Enable Global SR-IOV in IDRAC

vSphere Settings

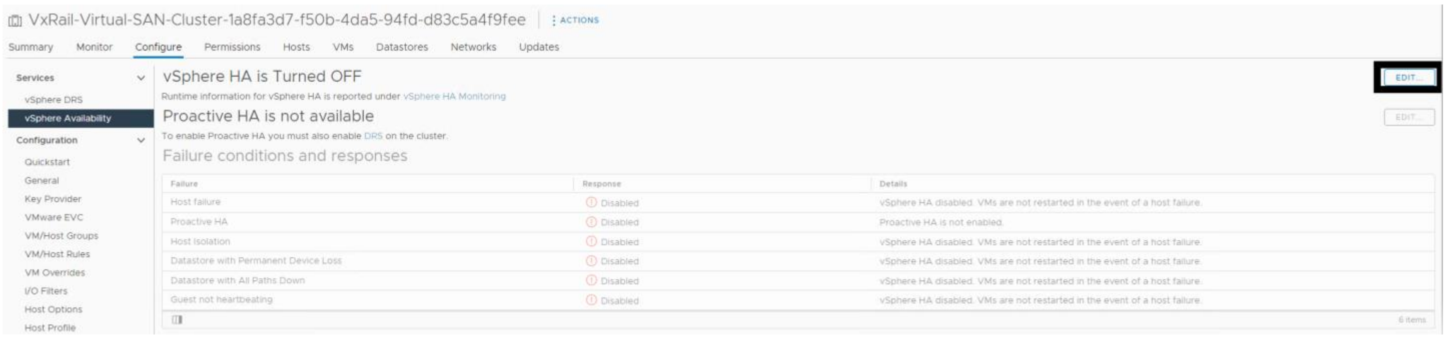
The vSphere settings should be configured for both vSphere with Tanzu and NVIDIA vGPU.

Pre-requisites for Configuring vSphere with Tanzu on a vSphere Cluster

Configure vSphere Availability

vSphere HA needs to be enabled on the target vSphere Cluster that will be used as the **Supervisor Cluster** in vSphere with Tanzu. To do so:

1. Log in to vCenter using the vSphere Client as an administrator user.
2. Click **Hosts and Clusters**.
3. Find the vSphere cluster that manages the ESXi hosts.
4. Navigate to **Configure** -> **Services** -> **vSphere Availability** and click **EDIT…**



vSphere HA is Turned OFF

Runtime information for vSphere HA is reported under vSphere HA Monitoring

Proactive HA is not available

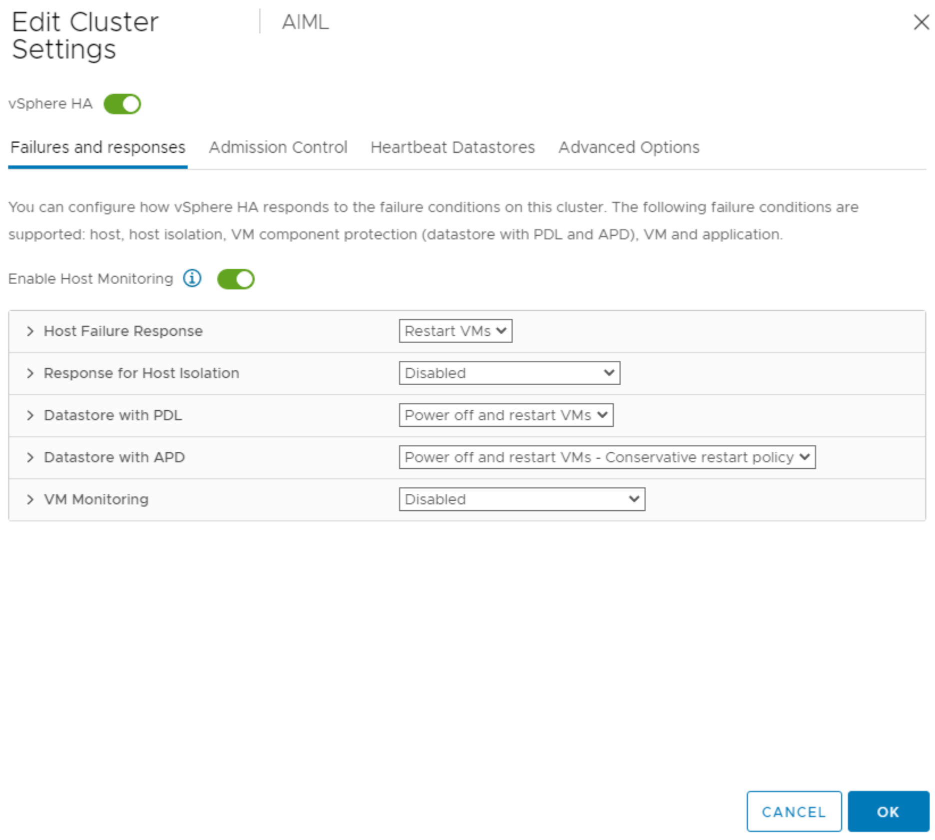
To enable Proactive HA you must also enable DRS on the cluster.

Failure conditions and responses

Failure	Response	Details
Host failure	Disabled	vSphere HA disabled. VMs are not restarted in the event of a host failure.
Proactive HA	Disabled	Proactive HA is not enabled.
Host Isolation	Disabled	vSphere HA disabled. VMs are not restarted in the event of a host failure.
Datastore with Permanent Device Loss	Disabled	vSphere HA disabled. VMs are not restarted in the event of a host failure.
Datastore with All Paths Down	Disabled	vSphere HA disabled. VMs are not restarted in the event of a host failure.
Guest not heartbeating	Disabled	vSphere HA disabled. VMs are not restarted in the event of a host failure.

Figure 4. Edit vSphere Availability Settings

5. Switch the toggle to the right for **vSphere HA** then click **OK**.



You can configure how vSphere HA responds to the failure conditions on this cluster. The following failure conditions are supported: host, host isolation, VM component protection (datastore with PDL and APD), VM and application.

Host Failure Response	Restart VMs
Response for Host Isolation	Disabled
Datastore with PDL	Power off and restart VMs
Datastore with APD	Power off and restart VMs - Conservative restart policy
VM Monitoring	Disabled

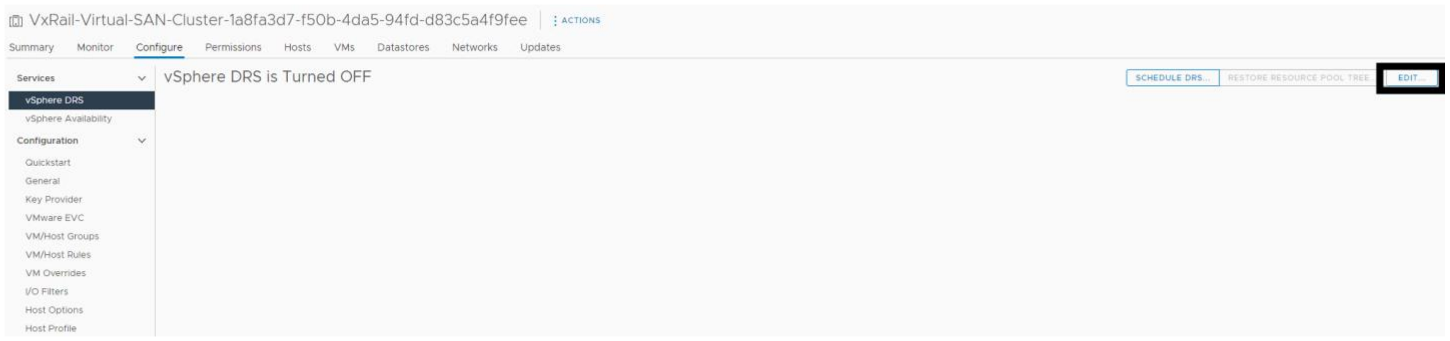
Figure 5. Enable vSphere HA

Configure vSphere DRS

vSphere DRS needs to be enabled and configured to be in **Fully Automated** mode on the target vSphere Cluster that will be used as the **Supervisor Cluster**.

To do so:

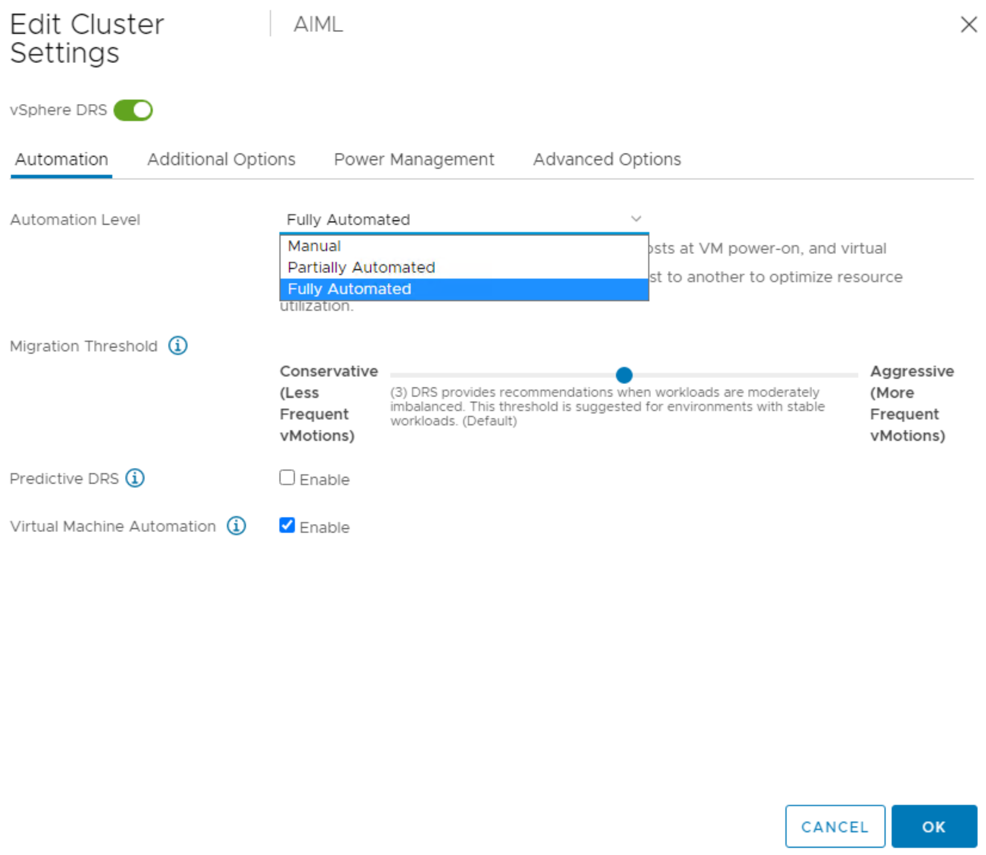
1. Log in to vCenter using the vSphere Client as an administrator user.
2. Click **Hosts and Clusters**.
3. Find the vSphere cluster that manages the ESXi hosts.
4. Navigate to **Configure** -> **Services** -> **vSphere DRS** and click **EDIT.**


VxRail-Virtual-SAN-Cluster-1a8fa3d7-f50b-4da5-94fd-d83c5a4f9fee

vSphere DRS is Turned OFF

Figure 6. Edit vSphere DRS Settings

5. Select **Fully Automated**, then click **OK.**


Edit Cluster Settings | AIML

vSphere DRS

Automation | Additional Options | Power Management | Advanced Options

Automation Level: Fully Automated (Manual / Partially Automated / Fully Automated)

...sts at VM power-on, and virtual ...st to another to optimize resource utilization.

Migration Threshold: Conservative (Less Frequent vMotions) — Aggressive (More Frequent vMotions)

(3) DRS provides recommendations when workloads are moderately imbalanced. This threshold is suggested for environments with stable workloads. (Default)

Predictive DRS: ☐ Enable

Virtual Machine Automation: ☑ Enable

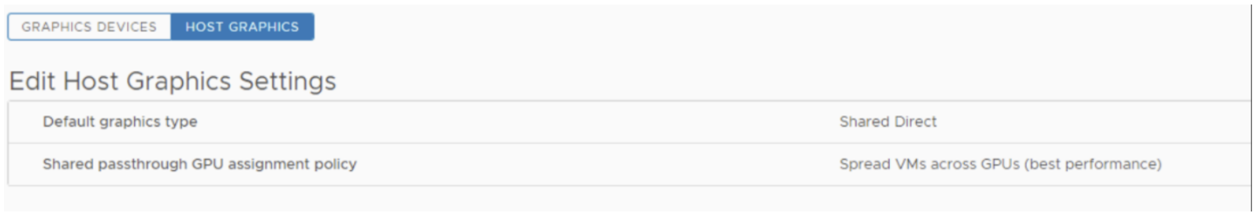
Figure 7. Configure DRS Automation Level to Fully Automated

Enable vGPU

Set the GPU Device to vGPU Mode Using the vSphere Host Graphics Setting

A GPU card can be configured in one of two modes: vSGA (shared virtual graphics) and vGPU. The NVIDIA GPU card should be configured with vGPU mode. This is specifically for use of the GPU in compute workloads, such as in machine learning or high-performance computing applications.

1. Log in to vCenter using the vSphere Client as an administrator user.
2. Click **Hosts and Clusters.**
3. Find the hosts with GPU installed within the targe vSphere cluster, for each of the hosts.
4. Navigate to **Configure** -> **Hardware** -> **Graphics**:
   - Under the **HOST GRAPHICS** tab, set
     - **Default graphics type** to **Shared Direct**
     - **Shared passthrough GPU assignment policy** to **Spread VMs across GPUs (best performance)**



Edit Host Graphics Settings

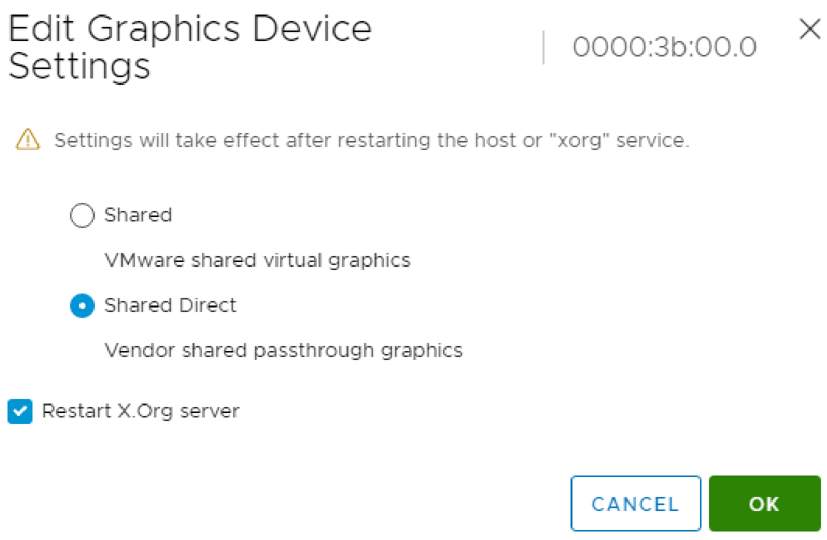
Default graphics type	Shared Direct
Shared passthrough GPU assignment policy	Spread VMs across GPUs (best performance)

Figure 8. Edit Host Graphics Setting

NOTE: If you choose to do this setting on the host server's command line, using the **esxcli** command, the option is called "Shared Passthru". Example: esxcli graphics host set –default-type SharedPassthru. "Shared Direct" as seen in the vSphere Client and SharedPassthru at the command-line level are the same.

The following steps might be needed if there are more than one physical GPUs on the host

- Under the **GRAPHICS DEVICES** tab, edit each GPU settings as follows and click **OK:**
  - **Shared Direct** is selected.
  - **Restart X Org server** is checked.



Edit Graphics Device Settings | 0000:3b:00.0

Settings will take effect after restarting the host or "xorg" service.

( ) Shared
VMware shared virtual graphics

(●) Shared Direct
Vendor shared passthrough graphics

[✓] Restart X.Org server

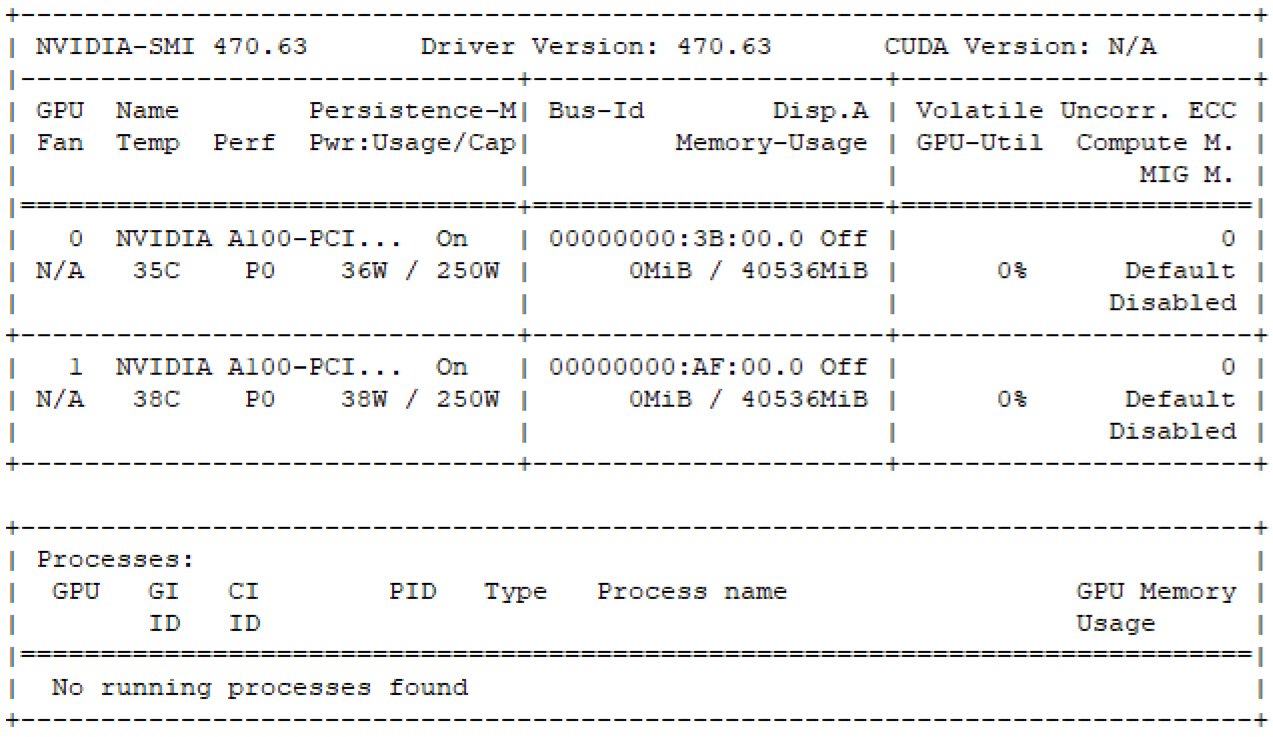
Figure 9. Edit Graphic Device Setting

Install the NVIDIA Virtual GPU Manager

Before VMs enabled for NVIDIA vGPU can be configured, the NVIDIA Virtual GPU Manager (sometimes called the "vGPU Host Driver") must be installed into the ESXi hypervisor. The process for installing the NVIDIA Virtual GPU Manager is described below.

For each ESXi server with a GPU device installed:

1. Enter maintenance mode.
2. Install the NVIDIA AIE vSphere Installation Bundle (VIB):
   a. SSH into ESXi server with the administrator privilege
   b. Remove any older version of NVIDIA Virtual GPU Manager vSphere Installation Bundle if installed:
   ```
   $ esxcli software vib remove -n `esxcli software vib list | grep "NVIDIA-AIE\|NVIDIA-
   VMware_ESXi" | awk '{print $1}'`
   ```
   c. Install the NVIDIA Virtual GPU Manager, run the following commands:
   ```
   $ esxcli software vib install -v Absolute_Path_of_Directory_of_the_VIB_File/NVIDIA-AIE*.vib
   ```
3. Exit maintenance mode.
4. Verify that the NVIDIA kernel driver can successfully communicate with the physical GPUs in the system by running the **nvidia-smi** command without any options.
   ```
   $ nvidia-smi
   ```
   If successful, the above command lists all the GPUs in the system.



```
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 470.63       Driver Version: 470.63       CUDA Version: N/A      |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                               |                      |               MIG M. |
|===============================+======================+======================|
|   0  NVIDIA A100-PCI...  On   | 00000000:3B:00.0 Off |                    0 |
| N/A   35C    P0    36W / 250W |      0MiB / 40536MiB |      0%      Default |
|                               |                      |             Disabled |
+-------------------------------+----------------------+----------------------+
|   1  NVIDIA A100-PCI...  On   | 00000000:AF:00.0 Off |                    0 |
| N/A   38C    P0    38W / 250W |      0MiB / 40536MiB |      0%      Default |
|                               |                      |             Disabled |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
|  No running processes found                                                 |
+-----------------------------------------------------------------------------+
```

Figure 10. nvidia-smi Output

If not, refer to the *Virtual GPU Software User Guide* for troubleshooting.

Change vGPU Scheduling (OPTIONAL)

If the generation of GPU(s) installed is after the *Maxwell architecture,* and MIG is not configured, we recommend changing the vGPU scheduling policy from **Best effort scheduling** (Default) to **Fixed share scheduling** or **Equal share scheduling** for optimal performance since AI workloads are typically long-running operations.

For detailed procedures of changing the vGPU scheduling policy, see *here*.

Enable Multi-Instance GPU or MIG (OPTIONAL)

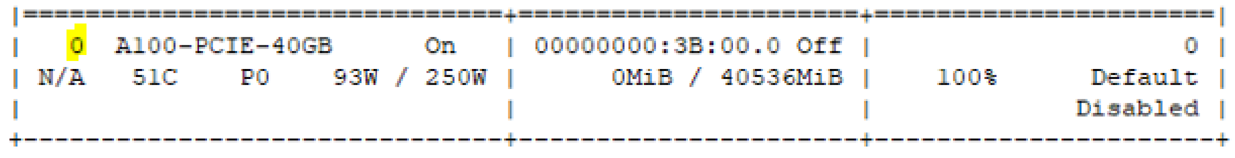
Pre-requisites:

- NVIDIA Ampere series GPU is installed
- *Global SR-IOV is enabled*
- *NVIDIA Virtual GPU Manager version R460.xx or later is installed*

By default, MIG is deactivated at the host server level. We may enable MIG on a per-GPU basis. The following steps describe how to check or change the MIG setting on the ESXi server with NVIDIA A100 GPU installed:

1. Enter maintenance mode.

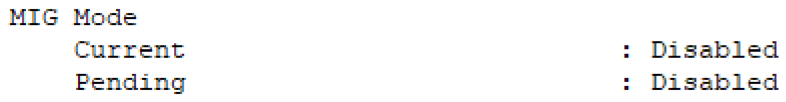
2. SSH into ESXi server with administrator privilege and find the GPU ID. If there are multiple GPUs in the host, find the GPU ID of the one to be checked by running `$ nvidia-smi`, and the number highlighted is the GPU ID; if there is only one GPU in the host, the GPU ID is 0.



```
|===============================+======================+======================|
|   0  A100-PCIE-40GB      On   | 00000000:3B:00.0 Off |                    0 |
| N/A   51C    P0    93W / 250W |      0MiB / 40536MiB |    100%      Default |
|                               |                      |             Disabled |
+-------------------------------+----------------------+----------------------+
```

Figure 11. Find GPU ID

3. MIG setting can be found by running:
`$ nvidia-smi -i [GPU-ID] -q | grep -A 2 "MIG Mode"`



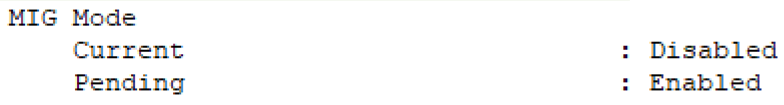
```
    MIG Mode
        Current                           : Disabled
        Pending                           : Disabled
```

Figure 12. MIG Mode is Deactivated

Current: Disabled means MIG is deactivated.
Pending: Disabled means there is no change to enable MIG to be made yet.

4. To enable MIG, run
`$ nvidia-smi -i [GPU-ID] -mig 1`
NOTE: If no GPU ID is specified, MIG mode is applied to all the GPUs on the system.

5. Recheck the MIG setting:
`$ nvidia-smi -i [GPU-ID] -q | grep -A 2 "MIG Mode"`



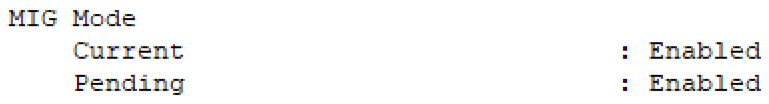
```
    MIG Mode
        Current                           : Disabled
        Pending                           : Enabled
```

Figure 13. MIG Mode Set to Enabled, Pending Reboot

Showing MIG setting is pending change and will be applied after rebooting the host.

6. Reboot the host. This is done to perform a GPU reset. Certain processes may be active on the GPU causing the reset to be delayed, so a reboot is a safe option to take here.

7. Recheck the MIG setting is enabled.



```
    MIG Mode
        Current                           : Enabled
        Pending                           : Enabled
```

Figure 14. MIG Mode is Enabled after Rebooting

NOTE: The MIG setting will persist across any subsequent reboots of the host server.

vSphere with Tanzu Kubernetes Settings

Create the Supervisor Cluster

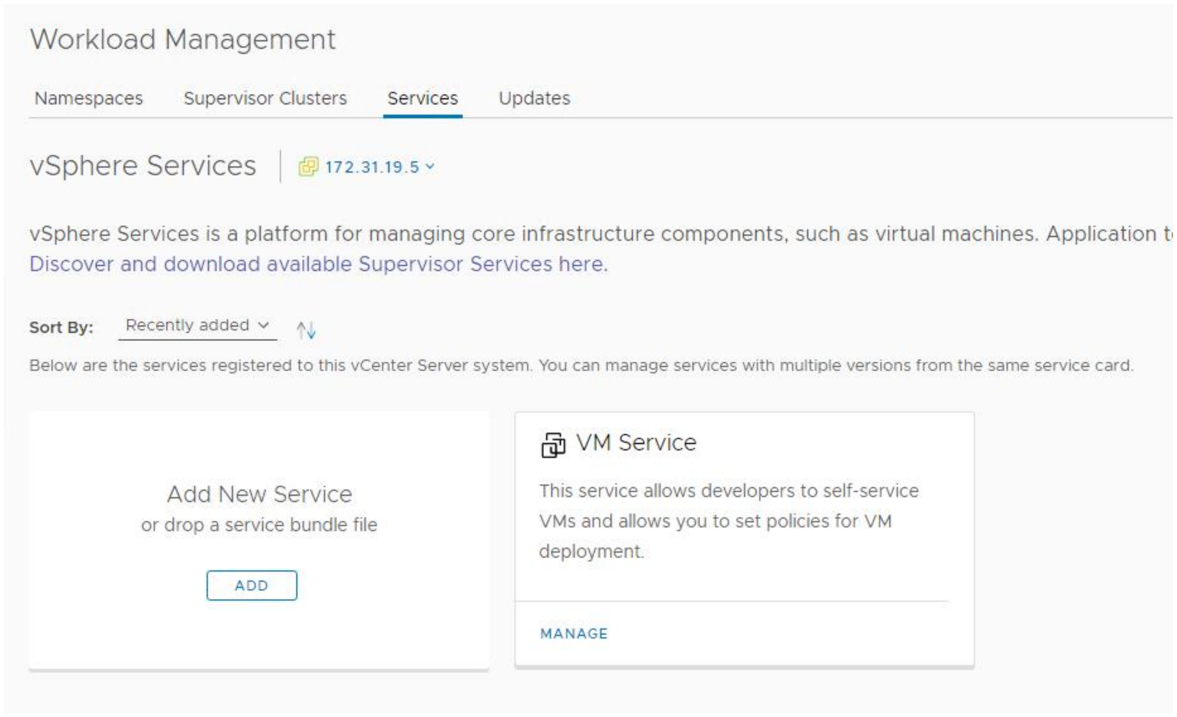
To enable a vSphere cluster for Workload Management, a Supervisor Cluster needs to be created. The pre-requisites listed *here* should be met and the steps can be found *here*.

NOTE: The content library for the Supervisor Cluster must be subscribed to *https://wp-content.vmware.com/v2/latest/lib.json*

Configure the VM Service

Once the Workload Management is enabled, the **VM Service** can be configured by the following steps:

1. In the console of Workload Management, click on **Namespaces** on the left, then click the **Services** tab on the right.
2. Click **Manage** in the **VM Service** box.
3. The next steps are to configure **VM Classes** and **Content Libraries** for the **VM Service**.



vSphere Services is a platform for managing core infrastructure components, such as virtual machines. Application t

Discover and download available Supervisor Services here.

Below are the services registered to this vCenter Server system. You can manage services with multiple versions from the same service card.

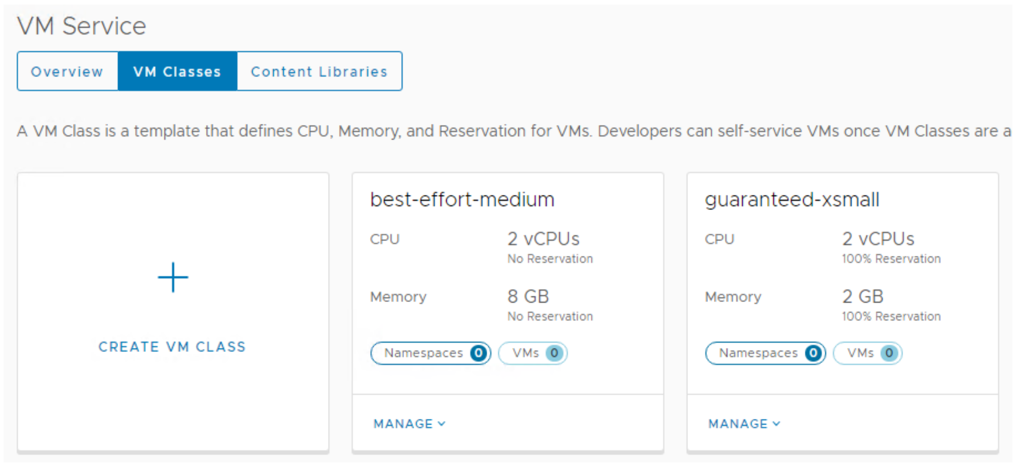
This service allows developers to self-service VMs and allows you to set policies for VM deployment.

Figure 15. Manage VM Classes using the VM Service

Create a VM Class with vGPU

In the **VM Service** console, click on the second tab **VM Classes** to browse the existing VM classes. By default, in vSphere with Tanzu, there are eight VM Classes that are named starting with **best-effort,** which means there is no reservation for the allocated CPU and memory, and there are another eight VM classes that are named starting with **guaranteed,** which means the CPU and memory allocated are 100% reserved.

1. To create the VM Class configured with vGPU, click on the box with **+** icon.



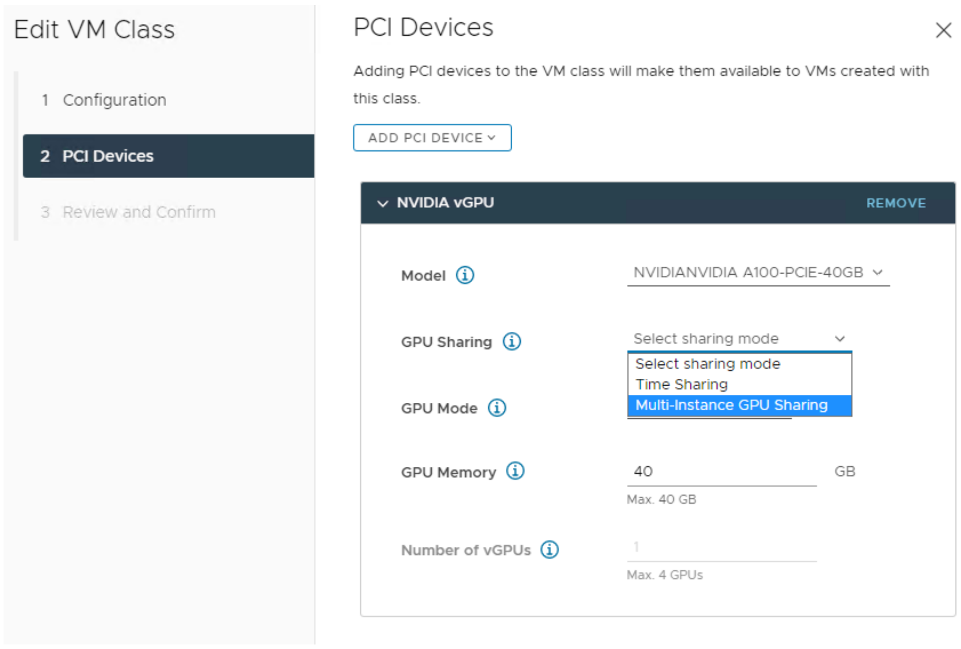
A VM Class is a template that defines CPU, Memory, and Reservation for VMs. Developers can self-service VMs once VM Classes are a

best-effort-medium
CPU 2 vCPUs No Reservation
Memory 8 GB No Reservation
Namespaces 0 VMs 0
MANAGE

guaranteed-xsmall
CPU 2 vCPUs 100% Reservation
Memory 2 GB 100% Reservation
Namespaces 0 VMs 0
MANAGE

Figure 16. Create VM Class

2. Specify a meaningful name for the VM Class; for example, 4cpu-8gram-a100-mig-20c, which means the vGPU that any new VM of this VM class is configured with a vGPU for an NVIDIA A100 with MIG enabled with 20GB GPU memory buffer.
3. Configure the **vCPU** and **Memory**, the **CPU Resource Reservation** can be optionally specified to reserve the CPU resource for the VM.
4. Choose **Yes** for the **PCI Devices,** and the **Memory Resource Reservation** would be automatically reserved for 100% in this case.
5. Move to the second page by clicking **NEXT** and it asks for the PCI Device selection. Expand the drop-down list **ADD PCI DEVICE**, **NVIDIA VGPU** should be chosen.
6. From the **Model** drop-down list, all the physical GPUs installed in the Supervisor Cluster will be listed. For this example, **NVIDIA A100-PCIE-40GB** should be selected.
7. After the model is chosen, **GPU Sharing** mode should be selected too. For this example, choose **Multi-Instance GPU Sharing**. If the **Model** chosen is not **MIG** capable or enabled, then **Time Sharing** would be the only option.



Adding PCI devices to the VM class will make them available to VMs created with this class.

Model: NVIDIANVIDIA A100-PCIE-40GB
GPU Sharing: Select sharing mode
GPU Mode
GPU Memory: 40 GB
Max. 40 GB
Number of vGPUs: 1
Max. 4 GPUs

Figure 17. Configure vGPU for VM Class

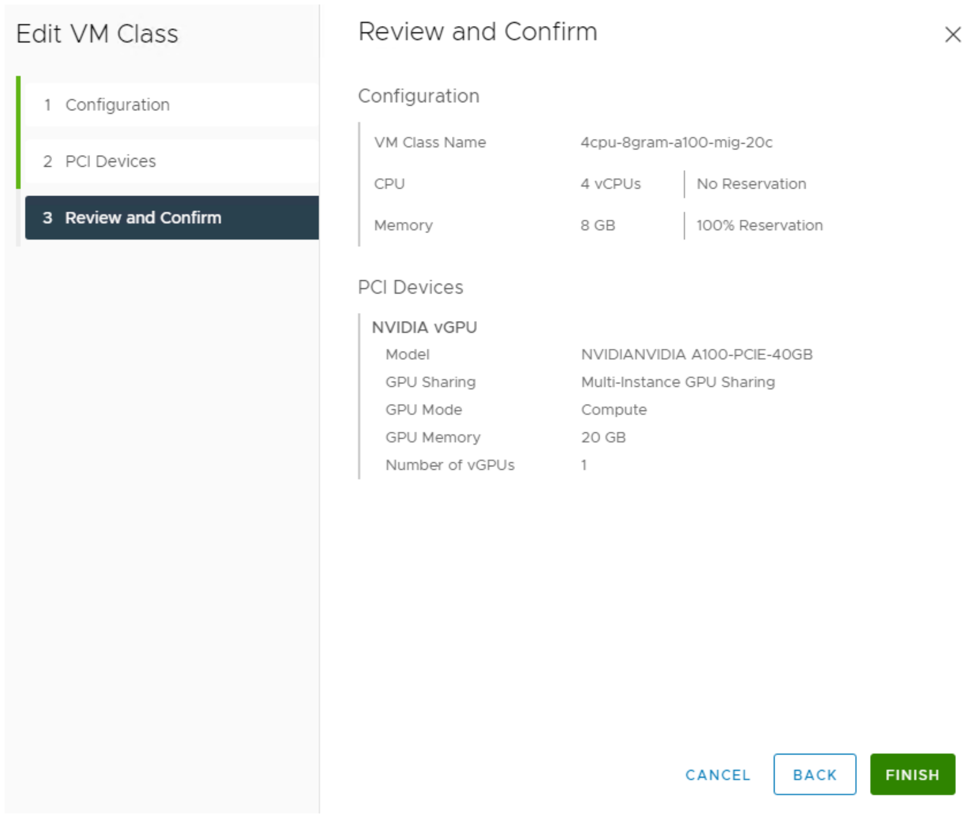
8. The items in the GPU Mode drop-down list are defined by the version of NVIDIA Virtual GPU Manager installed on the ESXi server, in this case, **Compute** should be chosen.
9. Specify the **GPU Memory** for the vGPU, using **NVIDIA A100 PCIE-40GB** as an example:
   - If **Multi-Instance GPU Sharing** (MIG) is chosen for the **GPU Sharing** mode, the legal numbers that can be specified are: **5**, **10**, **20**, **40** to represent 5GB, 10GB, 20GB, and 40GB. If the number specified is not one of these legal numbers, the system would automatically allocate the minimum legal number that can satisfy the memory requirement. For example, if 7 is given, the system would allocate 10GB as the GPU memory buffer to this VM class automatically.
   - If **Time Sharing** is chosen for the GPU Sharing mode, the legal numbers that can be specified are: **4**, **5**, **8**, **10**, **20,** and **40**, to represent 4GB, 5GB, 8GB, 10GB, 20GB, and 40GB of the GPU memory buffer. Similarly, if the number specified is not one of these numbers, the minimum legal number that can satisfy the requirement would be automatically allocated.

   NOTE: When configuring NVIDIA A100 MIG vGPU with Virtual Machine, there are two kinds of profiles with 20GB memory buffer, A100-3-20C and A100-4-20C, meaning 20GB memory buffer with 3 compute slices and 4 compute slices. To simplify the configuration for Tanzu Kubernetes Grid users, when configuring 20GB for memory buffer, **it only configures 20GB memory buffer with 3 compute slices**.

10. The last item **Number of vGPUs** can specify a number that is greater than **1** and equal or less than **4** if all the following conditions are met:
    - The vGPU is configured with **Time Sharing** mode.
    - The **GPU Memory** is specified at the maximum value for this GPU model.

    NOTE: If the number specified is greater than 1, please check if there are more than one identical physical GPUs installed in any of the ESXi hosts in the Supervisor Cluster.

    In this example, this item is greyed-out because the **MIG** mode is chosen, and **GPU Memory** is not specified at the maximum value.
11. Review the configuration and click **FINISH** if everything looks good.



Configuration

VM Class Name	4cpu-8gram-a100-mig-20c	
CPU	4 vCPUs	No Reservation
Memory	8 GB	100% Reservation

PCI Devices

NVIDIA vGPU

Model	NVIDIANVIDIA A100-PCIE-40GB
GPU Sharing	Multi-Instance GPU Sharing
GPU Mode	Compute
GPU Memory	20 GB
Number of vGPUs	1

Figure 18. Review the VM Class

After the VM Class is successfully created, there will be a banner on the top.

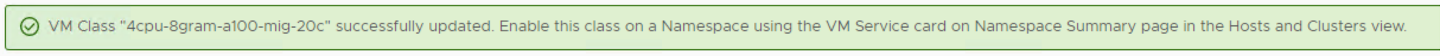

VM Class "4cpu-8gram-a100-mig-20c" successfully updated. Enable this class on a Namespace using the VM Service card on Namespace Summary page in the Hosts and Clusters view.

Figure 19. VM Class is Successfully Created

If the GPU model or GPU MIG setting is not identical in the **Supervisor Cluster**; for example, the **Supervisor Cluster** has both NVIDIA A100 GPU and NVIDIA V100s GPU installed, or some of the NVIDIA A100 GPUs have MIG enabled and some are disabled, repeat the steps above to create multiple VM classes for different GPU models or settings.

Create Local Content Library for VM Service (OPTIONAL)

In addition to a Tanzu Kubernetes Grid Cluster, Tanzu Kubernetes Grid Service can also provide stand-alone VM lifecycle management to the developers with **VM Service**. To enable that, in the **VM Service** console, click on the third tab **Content Libraries** to browse the content libraries added to any **Namespace** through the **VM Service** card on the Namespace Summary page. Click the box with **+** icon in it and it will redirect to the **Content Libraries** console and follow the instruction *here* to create a local content library for VM Service.

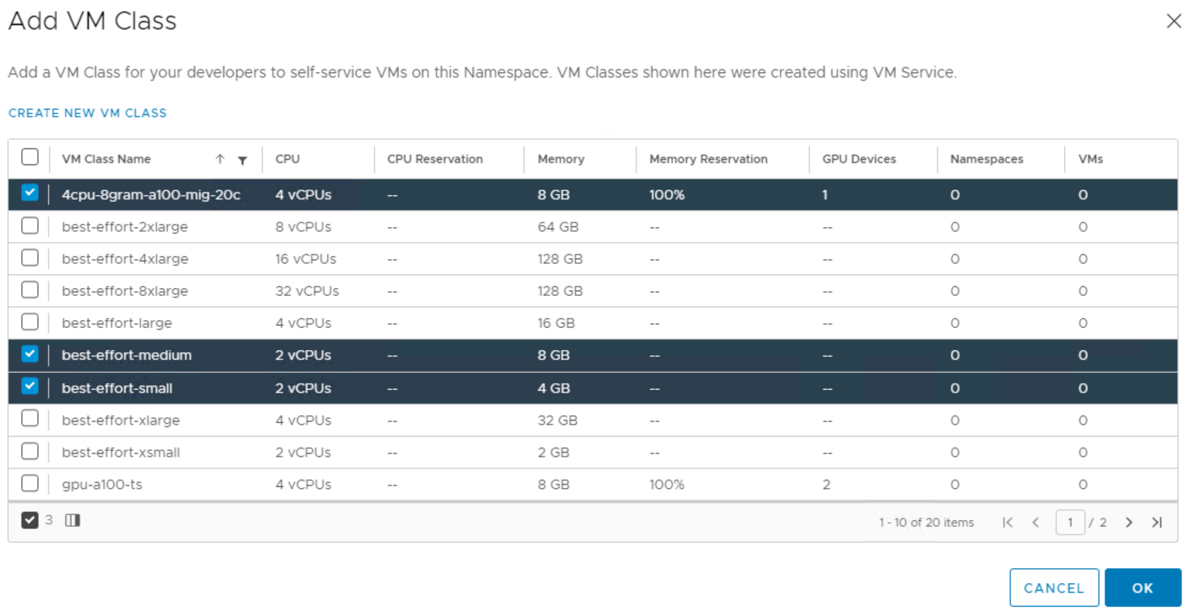
Configure Namespace

To configure the Namespace on the Supervisor Cluster:

1. Follow the instruction *here* to create the **Namespace** on the **Supervisor Cluster**, and configure the **Permissions**, **Storage**, **Capacity,** and **Usage**.

2. Follow the instruction of **Associate the Content Library with the vSphere Namespace** in this *document* to configure the Namespace for vSphere with Tanzu Kubernetes releases if there is no content library associated with the Namespace.

3. To add VM Classes to this **Namespace**, click **ADD VM CLASS** in the **VM Service** box under the Namespace **Summary** console, and check the VM Classes needed to be added, make sure adding the VM Class with vGPU configured that was created *earlier*, and also add some default VM Classes like best-effort-small and best-effort-medium.


Add VM Class

Add a VM Class for your developers to self-service VMs on this Namespace. VM Classes shown here were created using VM Service.

CREATE NEW VM CLASS

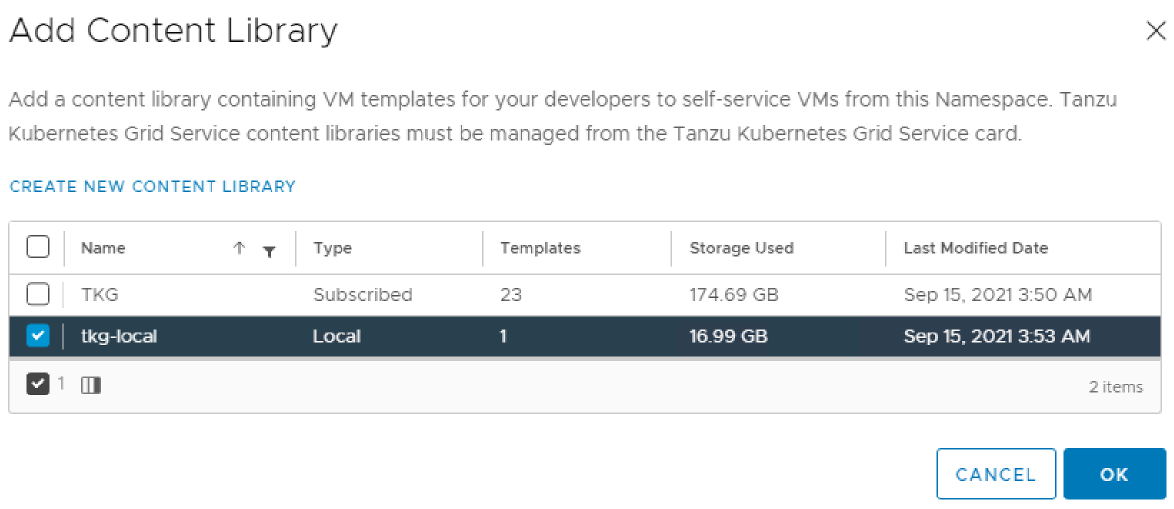
	VM Class Name	CPU	CPU Reservation	Memory	Memory Reservation	GPU Devices	Namespaces	VMs
✓	4cpu-8gram-a100-mig-20c	4 vCPUs	--	8 GB	100%	1	0	0
☐	best-effort-2xlarge	8 vCPUs	--	64 GB	--	--	0	0
☐	best-effort-4xlarge	16 vCPUs	--	128 GB	--	--	0	0
☐	best-effort-8xlarge	32 vCPUs	--	128 GB	--	--	0	0
☐	best-effort-large	4 vCPUs	--	16 GB	--	--	0	0
✓	best-effort-medium	2 vCPUs	--	8 GB	--	--	0	0
✓	best-effort-small	2 vCPUs	--	4 GB	--	--	0	0
☐	best-effort-xlarge	4 vCPUs	--	32 GB	--	--	0	0
☐	best-effort-xsmall	2 vCPUs	--	2 GB	--	--	0	0
☐	gpu-a100-ts	4 vCPUs	--	8 GB	100%	2	0	0

✓ 3   1 - 10 of 20 items   1 / 2

CANCEL OK

Figure 20. Associate VM Class into Namespace

4. Optionally, to add the local content library created *earlier* to this Namespace, click **ADD CONTENT LIBRARY** in the **VM Service** box under the Namespace **Summary** console, and check the local content library created *earlier* and click **OK**.



Add Content Library

Add a content library containing VM templates for your developers to self-service VMs from this Namespace. Tanzu Kubernetes Grid Service content libraries must be managed from the Tanzu Kubernetes Grid Service card.

CREATE NEW CONTENT LIBRARY

	Name	Type	Templates	Storage Used	Last Modified Date
☐	TKG	Subscribed	23	174.69 GB	Sep 15, 2021 3:50 AM
✓	tkg-local	Local	1	16.99 GB	Sep 15, 2021 3:53 AM

1 — 2 items

Figure 21. Associate Local Content Library to Namespace

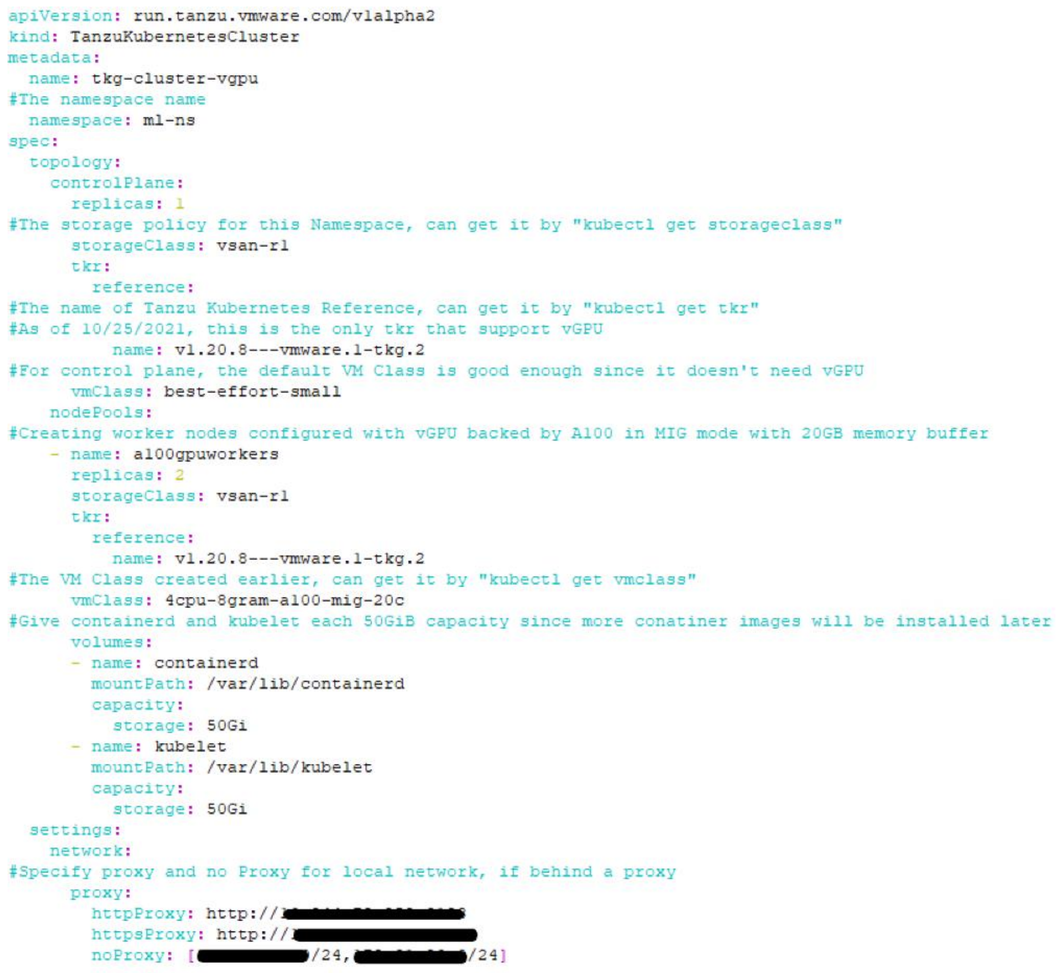
Provision Tanzu Kubernetes Grid Cluster with vGPU

Prerequisites:

- A *VM Class* is created and configured with vGPU.
- The Tanzu Kubernetes Grid Content Library has downloaded image **ob-18807685-tkgs-ova-ubuntu-2004-v1.20.8---vmware.1-tkg.2.** You can check this by using the **kubectl get vmimage** command while you are logged in to the Supervisor cluster.
- The *Namespace is properly configured* and the VM Classes with vGPU are associated into the Namespace**.**

Follow these steps to create a Tanzu Kubernetes Grid Cluster:

1. *Download and install the Kubernetes CLI Tools*.
2. Connect to the Supervisor Cluster using the tool installed: **kubectl,** follow *Configure Secure Login for vSphere with Tanzu Clusters* and *Connect to the Supervisor Cluster as a vCenter Single Sign-On User*.
3. Use `$kubectl config use-context <example-context-name>` to switch to the **Namespace** created *earlier*.
4. Follow the instructions *here* to prepare the Tanzu Kubernetes Grid Cluster YAML file, and this *example* can be referred to. Make sure to use **Tanzu Kubernetes Grid Service v1alpha2 API.** There are some YAML file examples on *Github*.

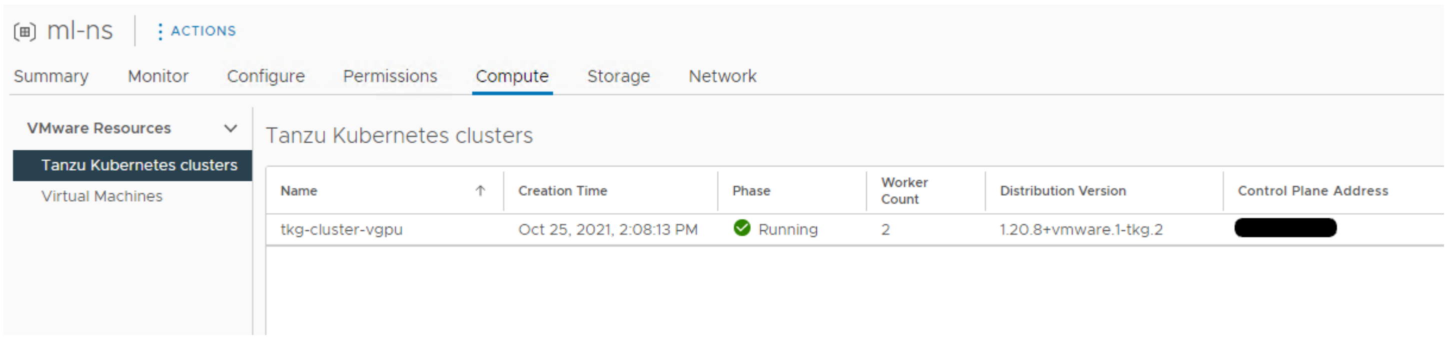


```
apiVersion: run.tanzu.vmware.com/v1alpha2
kind: TanzuKubernetesCluster
metadata:
  name: tkg-cluster-vgpu
#The namespace name
  namespace: ml-ns
spec:
  topology:
    controlPlane:
      replicas: 1
#The storage policy for this Namespace, can get it by "kubectl get storageclass"
      storageClass: vsan-r1
      tkr:
        reference:
#The name of Tanzu Kubernetes Reference, can get it by "kubectl get tkr"
#As of 10/25/2021, this is the only tkr that support vGPU
          name: v1.20.8---vmware.1-tkg.2
#For control plane, the default VM Class is good enough since it doesn't need vGPU
      vmClass: best-effort-small
    nodePools:
#Creating worker nodes configured with vGPU backed by A100 in MIG mode with 20GB memory buffer
    - name: a100gpuworkers
      replicas: 2
      storageClass: vsan-r1
      tkr:
        reference:
          name: v1.20.8---vmware.1-tkg.2
#The VM Class created earlier, can get it by "kubectl get vmclass"
      vmClass: 4cpu-8gram-a100-mig-20c
#Give containerd and kubelet each 50GiB capacity since more conatiner images will be installed later
      volumes:
      - name: containerd
        mountPath: /var/lib/containerd
        capacity:
          storage: 50Gi
      - name: kubelet
        mountPath: /var/lib/kubelet
        capacity:
          storage: 50Gi
  settings:
    network:
#Specify proxy and no Proxy for local network, if behind a proxy
      proxy:
        httpProxy: http://[redacted]
        httpsProxy: http://[redacted]
        noProxy: [[redacted]/24,[redacted]/24]
```

Figure 22. Tanzu Kubernetes Grid Cluster Definition

5. Run $kubectl apply -f <tkg_cluster_yaml_file_name> to provision the Tanzu Kubernetes cluster.

6. After a few minutes, run $kubectl get tkc <tkg_cluster_name> to ensure that the new Tanzu Kubernetes Grid Cluster is ready. This can also be checked from the vSphere web client by choosing the **Namespace**, and navigating to **Compute**, the cluster information will be shown under **Tanzu Kubernetes clusters**.


ml-ns | ACTIONS

Summary Monitor Configure Permissions Compute Storage Network

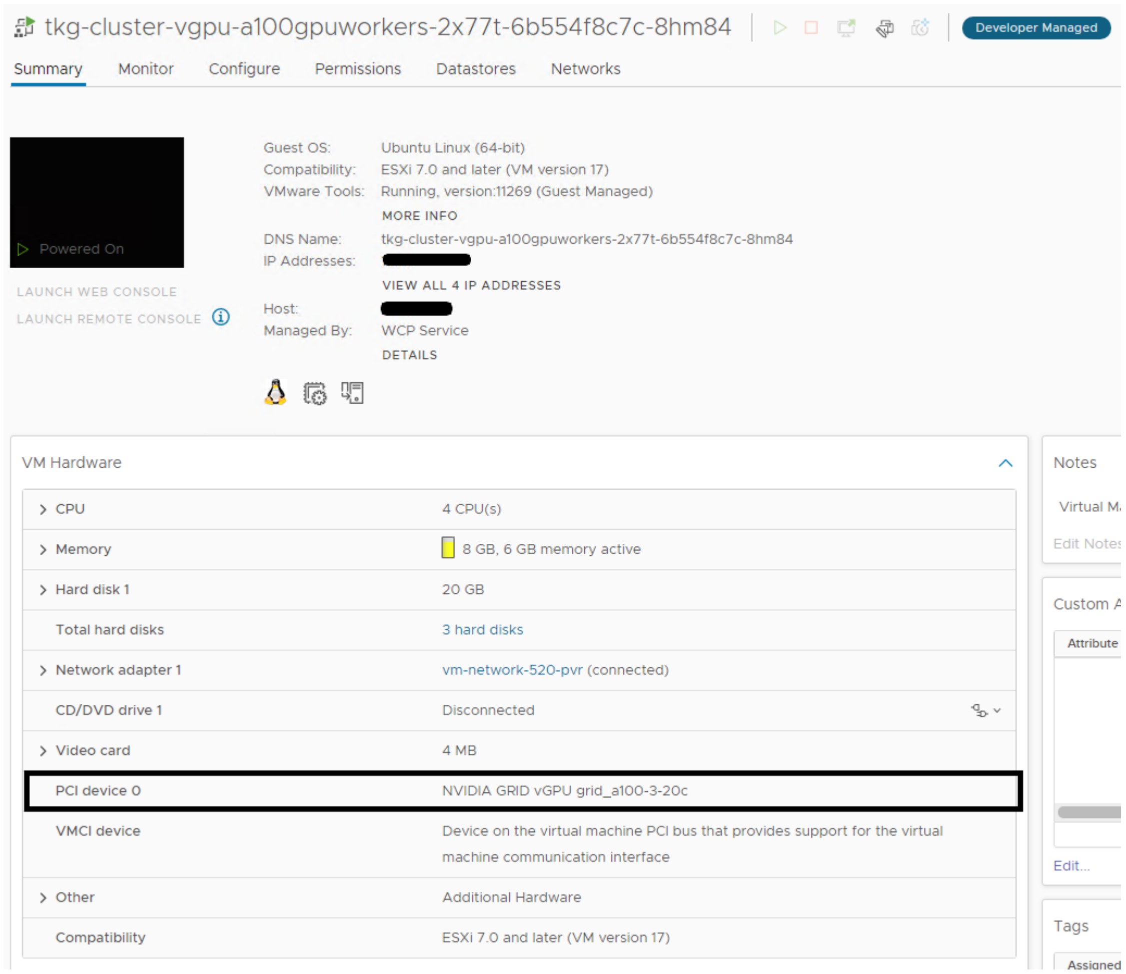
VMware Resources
Tanzu Kubernetes clusters
Virtual Machines

Tanzu Kubernetes clusters

Name	Creation Time	Phase	Worker Count	Distribution Version	Control Plane Address
tkg-cluster-vgpu	Oct 25, 2021, 2:08:13 PM	✓ Running	2	1.20.8+vmware.1-tkg.2	[redacted]

Figure 23. Tanzu Kubernetes Grid Cluster is Successfully Created

7. The Tanzu Kubernetes cluster with two worker nodes is provisioned, and each worker node is configured with one vGPU, and each vGPU has 3 compute slices and 20GB memory buffer; for example, a valid MIG-backed vGPU profile.


tkg-cluster-vgpu-a100gpuworkers-2x77t-6b554f8c7c-8hm84 | Developer Managed

Summary Monitor Configure Permissions Datastores Networks

Powered On
LAUNCH WEB CONSOLE
LAUNCH REMOTE CONSOLE

Guest OS:	Ubuntu Linux (64-bit)
Compatibility:	ESXi 7.0 and later (VM version 17)
VMware Tools:	Running, version:11269 (Guest Managed)
	MORE INFO
DNS Name:	tkg-cluster-vgpu-a100gpuworkers-2x77t-6b554f8c7c-8hm84
IP Addresses:	[redacted]
	VIEW ALL 4 IP ADDRESSES
Host:	[redacted]
Managed By:	WCP Service
	DETAILS

VM Hardware

CPU	4 CPU(s)
Memory	8 GB, 6 GB memory active
Hard disk 1	20 GB
Total hard disks	3 hard disks
Network adapter 1	vm-network-520-pvr (connected)
CD/DVD drive 1	Disconnected
Video card	4 MB
PCI device 0	NVIDIA GRID vGPU grid_a100-3-20c
VMCI device	Device on the virtual machine PCI bus that provides support for the virtual machine communication interface
Other	Additional Hardware
Compatibility	ESXi 7.0 and later (VM version 17)

Notes
Virtual M
Edit Notes
Custom A
Attribute
Edit...
Tags
Assigned

Figure 24. Worker Nodes Configure with vGPU

Next Step

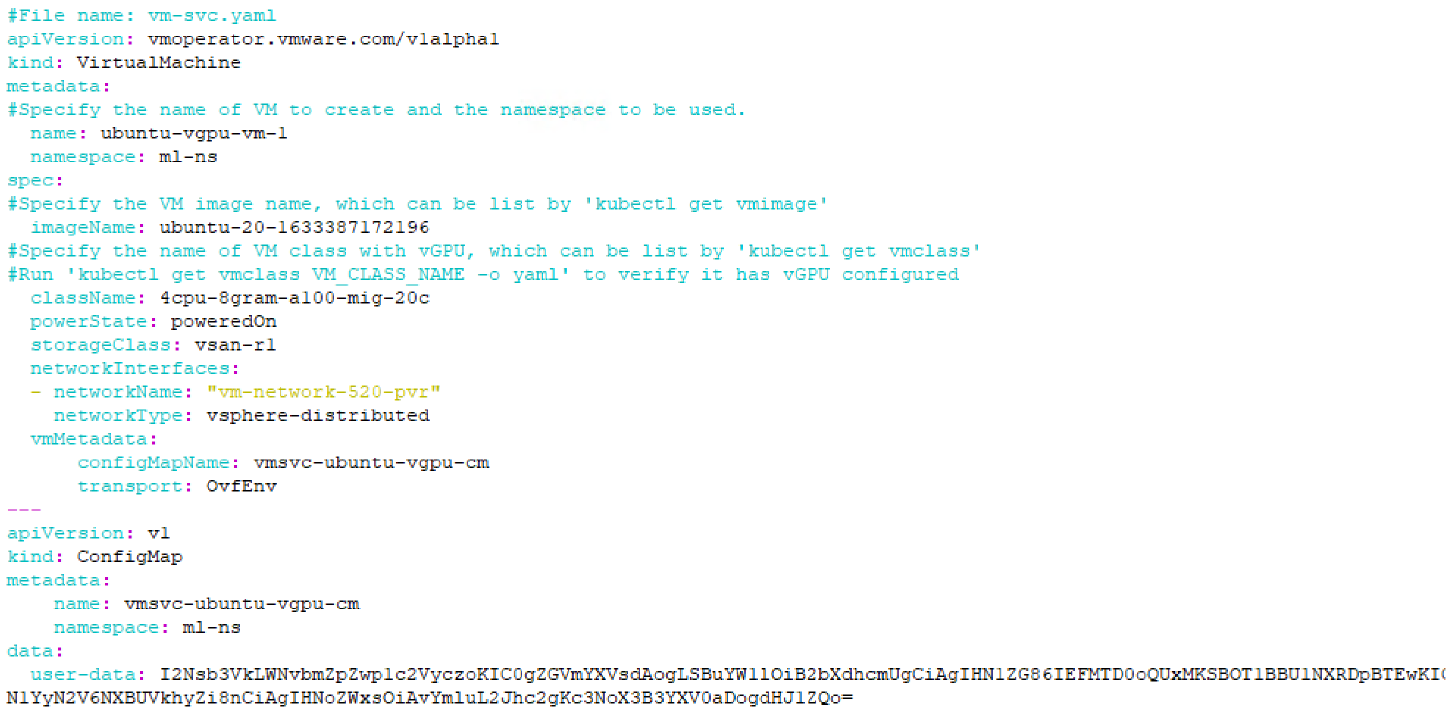
When the Tanzu Kubernetes Grid Cluster configured with vGPU is ready, log out from the Namespace by $kubectl logout and *re-login into the Tanzu Kubernetes Grid Cluster*, then follow the instructions to install the *NVIDIA GPU Operator* with NVIDIA AI Enterprise and then run GPU applications.

Provision Stand-alone VM with vGPU

Prerequisites:

1. Download **ubuntu-20-1633387172196.ova** or **centos-stream-8-vmservice-v1alpha1-1633387532874.ova** from *VMware Marketplace* and import to the *local content library*.
2. The *local content library* has been added to the VM Service
3. *VM Class with vGPU is created* and *associated with the Namespace*.

Follow the instruction to *Deploy a Virtual Machine with vGPU in vSphere with Tanzu*. When preparing the VM YAML file, the name of *VM Class with vGPU configured* should be specified to **spec.className;** and the name of VM image imported to the local content library should be used for **spec.imageName**.

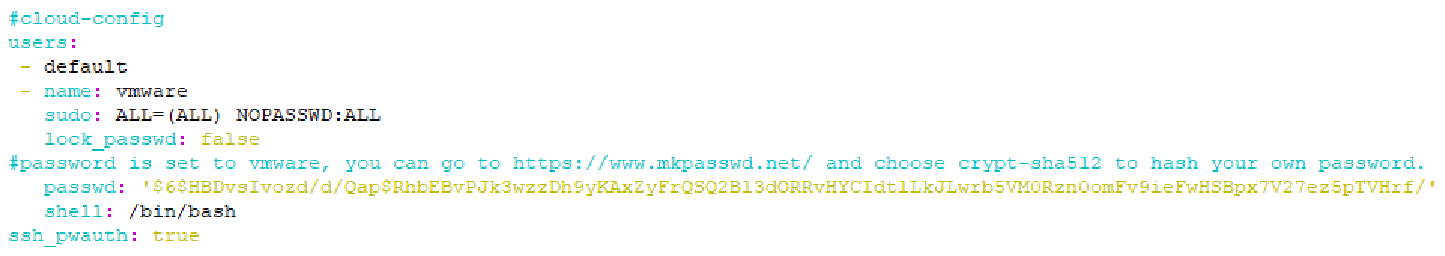


```
#File name: vm-svc.yaml
apiVersion: vmoperator.vmware.com/v1alpha1
kind: VirtualMachine
metadata:
#Specify the name of VM to create and the namespace to be used.
  name: ubuntu-vgpu-vm-1
  namespace: ml-ns
spec:
#Specify the VM image name, which can be list by 'kubectl get vmimage'
  imageName: ubuntu-20-1633387172196
#Specify the name of VM class with vGPU, which can be list by 'kubectl get vmclass'
#Run 'kubectl get vmclass VM_CLASS_NAME -o yaml' to verify it has vGPU configured
  className: 4cpu-8gram-a100-mig-20c
  powerState: poweredOn
  storageClass: vsan-r1
  networkInterfaces:
  - networkName: "vm-network-520-pvr"
    networkType: vsphere-distributed
  vmMetadata:
      configMapName: vmsvc-ubuntu-vgpu-cm
      transport: OvfEnv
---
apiVersion: v1
kind: ConfigMap
metadata:
    name: vmsvc-ubuntu-vgpu-cm
    namespace: ml-ns
data:
  user-data: I2Nsb3VkLWNvbmZpZwplc2VyczoKIC0gZGVmYXVsdAogLSBuYW1lOiB2bXdhcmUgCiAgIHN1ZG86IEFMTD0oQUxMKSBOT1BBU1NXRDpBTEwKI
N1YyN2V6NXBUVkhyZi8nCiAgIHNoZWxsOiAvYmluL2Jhc2gKc3NoX3B3YXV0aDogdHJ1ZQo=
```

Figure 25. Stand-alone VM Definition

The value of **data.user-data** is base64 encoded cloud-config file, by running $cat <cloud-config_yaml_file> | base64 -w 0.

In the cloud-config.yaml, both the username and password values have been set to **vmware** in the example below to access the VM via SSH. The YAML file examples can be found on *Github*.



```
#cloud-config
users:
 - default
 - name: vmware
   sudo: ALL=(ALL) NOPASSWD:ALL
   lock_passwd: false
#password is set to vmware, you can go to https://www.mkpasswd.net/ and choose crypt-sha512 to hash your own password.
   passwd: '$6$HBDvsIvozd/d/Qap$RhbEBvPJk3wzzDh9yKAxZyFrQSQ2Bl3dORRvHYCIdtlLkJLwrb5VM0Rzn0omFv9ieFwHSBpx7V27ez5pTVHrf/'
   shell: /bin/bash
ssh_pwauth: true
```

Figure 26. VM User-data Definition

Run $kubectl apply -f <vm_yaml_file_name> to create the stand-alone VM, wait for few minutes, and run $kubectl get vm <vm_name> -o yaml | grep vmIp to ensure the VM is running, and IP address is ready.

Next Step

When the stand-alone VM configured with vGPU is ready:

1. The VM created can be accessed using SSH. Follow the *instructions* to install the NVIDIA vGPU Driver
2. The VM created can be accessed using SSH. Install the machine learning applications.

Multi-Node Machine Learning Using Tanzu Kubernetes Grid with vGPU and RoCE

Deploying a distributed or multi-node machine learning platform on Tanzu Kubernetes Grid can easily use multiple GPUs from different ESXi hosts. This is done by provisioning multiple worker nodes, each with vGPU(s) configured on them. Faster network speeds become more critical to bring the performance of distributed machine learning applications to the next level.

RDMA over Converged Ethernet (RoCE) backed networking provides an efficient, low latency, light-weight transport and enables faster application completion, better server utilization, and higher scalability, which is highly recommended for multi-node (distributed) learning.

To deploy a machine learning platform with Tanzu Kubernetes Grid Service with RoCE, the worker nodes configured with vGPU should also be configured with a RoCE-backed passthrough Network Interface Card (NIC). In this case, the worker nodes of the Tanzu Kubernetes Grid Cluster do not support vSphere vMotion. This is the tradeoff of enhancing the network performance by leveraging RoCE.

All the ESXi servers with GPU installed also need to meet the following pre-requisites and configurations.

Deployment Prerequisites

In addition to the *Deployment Prerequisites* in the Deploy Tanzu Kubernetes Grid with vGPU chapter, at least one NVIDIA Network Interface Card or other RoCE capable Network Interface Card should be installed on each ESXi host that is equipped with a GPU.

NVIDIA Network Interface Card

It's recommended to install one NVIDIA ConnectX-5, ConnectX-6, or Bluefield-2 network card in each of the GPU servers. These NVIDIA network adapters are RDMA and Address Translation Services (ATS) capable, which can enable RoCE to potentially maximize the performance of distributed Machine Learning across multiple Tanzu Kubernetes cluster worker nodes on different ESXi servers.

Hardware Settings

In addition to the *hardware settings* in the Deploy Tanzu Kubernetes Grid with vGPU chapter, all the ESXi servers with GPU and NVIDIA NIC installed need to meet the following configurations.

Configure NVIDIA Network Interface Card and Switch for RoCEv2

Ensure the *recommended NVIDIA Network Interface Card* or other RoCE capable Network Adapter Installed and the firmware of the network interface card is up-to-date, and follow the procedures *here* to configure Lossless RoCE for NVIDIA Onyx Switches, if the intermediate network switch is from other vendors, see the user manual.

PCIe Root Complex Placement (Recommended)

To achieve the best performance for multi-node learning, it is recommended that both the network adapter and the GPU be physically located on the same PCIe IO root complex.

vSphere Settings

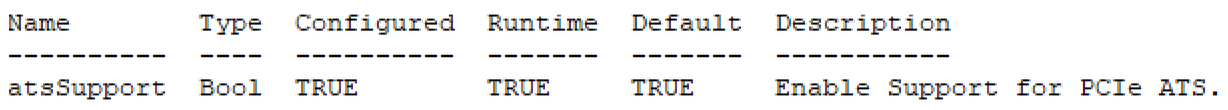
Implementing distributed machine learning infrastructure with RoCE involves two or more Tanzu Kubernetes cluster worker nodes running on different ESXi hosts with GPUs and NVIDIA NICs installed., In addition to the *vSphere settings* in the Deploy Tanzu Kubernetes Grid with vGPU chapter, the requirements and settings below should be done on each ESXi host involved.

Verify ATS (Address Translation Service) Setting

VMware vSphere 7 Update 2 enhances PCIe device-to-device communication through a PCIe mechanism named Address Translation Service (ATS). ATS allows local caching of each PCIe device's virtual-to-physical address mapping. This speeds up the local peer-to-peer device communication within a host. It therefore enhances the rate at which data travels from the GPU to the network card locally (bypassing the CPU and main memory). That network traffic then goes on to another GPU on another host, with the same ATS speed-up on the remote side. VMware vSphere 7 Update 3 has enabled ATS in the ESXi by default, to verify that:

1. SSH into ESXi server with the administrator privilege.
2. Verify the ATS setting by running:
   ```
   $ esxcli system settings kernel list -o atsSupport
   ```

The following figure illustrates the ATS is enabled by default.



```
Name         Type  Configured  Runtime  Default  Description
----------   ----  ----------  -------  -------  -----------
atsSupport   Bool  TRUE        TRUE     TRUE     Enable Support for PCIe ATS.
```

Figure 27. ATS is Enabled by Default

If it shows **False** for the **Runtime**, follow the steps below to enable it:

1. Enter maintenance mode.
2. Run the following command to enable ATS:
   ```
   $ esxcli system settings kernel set -s atsSupport -v TRUE
   ```
3. Reboot the ESXi server and re-verify the ATS mode.
4. Exit maintenance mode.

Verify the GPU and Network Adapter (or HCA: Host Channel Adapter) Placement

To verify whether the GPU and Network Adapter are connected to the same PCIe Root Complex, SSH into the ESXi server with administrator privilege and run the following command to get the PCI information and redirect it to a file,

```
$ lspci -e > lspci.out
```

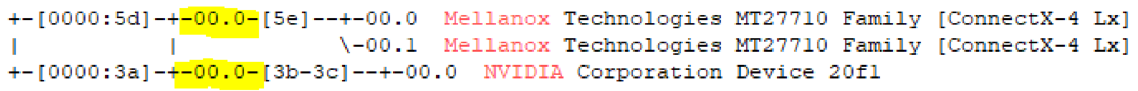
Since ESXi does not support "lspci -t", the output file needs to be placed into another Linux server and then run:

```
$ lspci -tvv -F lspci.out > lspciTree.out
```

Then run the following command to find out the root complex placement:

```
$ grep -e "Mellanox\|NVIDIA" lspciTree.out
```

The output below is an example showing that both GPU and the HCA are connected to the same root complex (00.0) which is highlighted, and we can find out the network adapter and GPU's PCI Bus ID (5e and 3b).



```
+-[0000:5d]-+-00.0-[5e]--+-00.0  Mellanox Technologies MT27710 Family [ConnectX-4 Lx]
|           |             \-00.1  Mellanox Technologies MT27710 Family [ConnectX-4 Lx]
+-[0000:3a]-+-00.0-[3b-3c]--+-00.0  NVIDIA Corporation Device 20f1
```

Figure 28. Find Root Complex Placement

Configure the Network Adapter to be in Passthrough mode:

1. Log in to vCenter using the vSphere Client as an administrator user.
2. Click **Hosts and Clusters.**
3. Find the hosts with GPU and RoCE enabled, for each of the hosts.
4. Navigate to **Configure -> Hardware -> PCI Devices**.

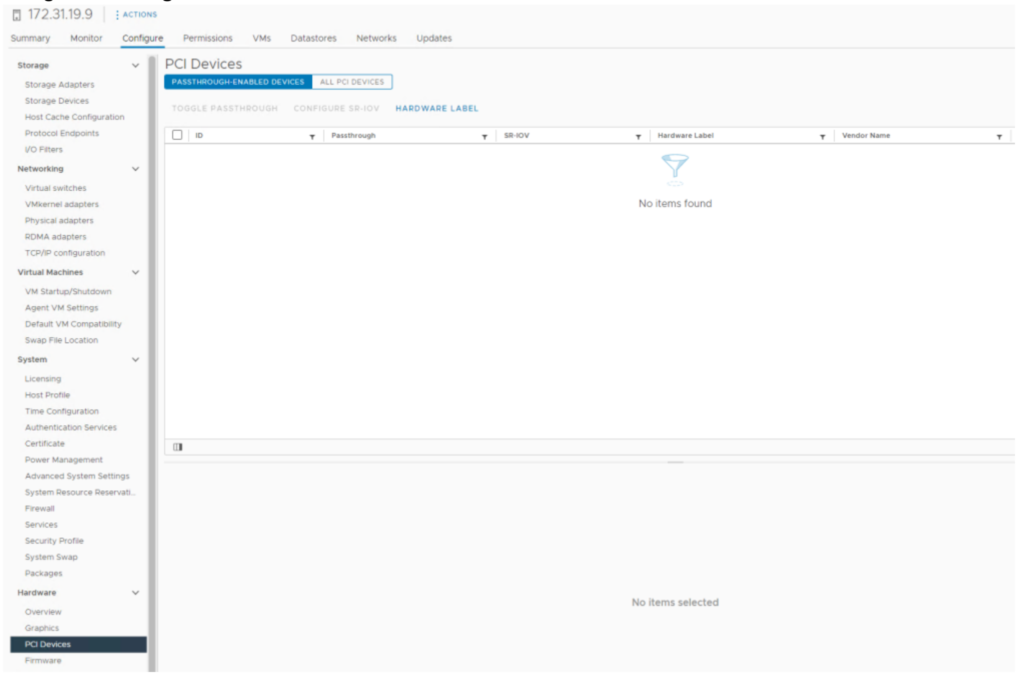
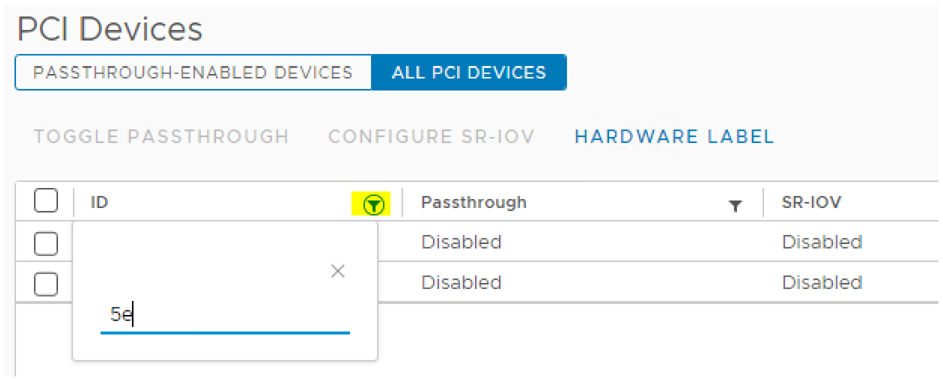


Figure 29. Configure Passthrough for PCI Devices

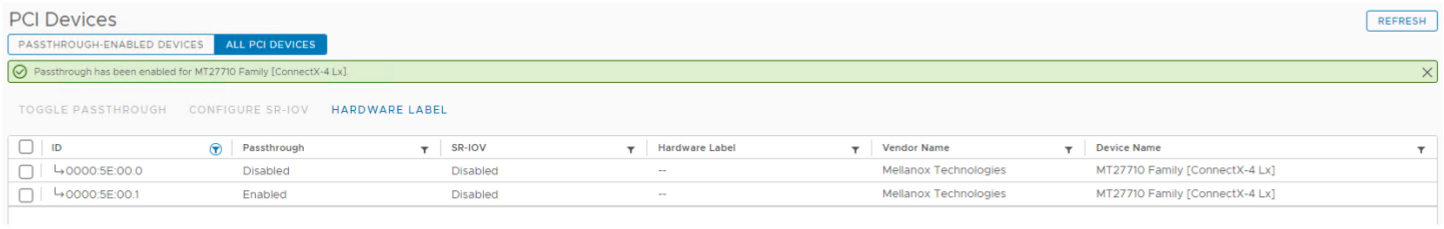
5. Click the **ALL PCI DEVICES** tab.
6. Filter the ID using the PCI Bus ID found above (5e). Make sure it is not being used by any ESX/ESXi Device, check the box on the left, and click **TOGGLE PASSTHROUGH** to set it to passthrough**.**



ID	Passthrough	SR-IOV
5e	Disabled	Disabled
	Disabled	Disabled

Figure 30. Select NVIDIA Network Adapter to Passthrough

7. There will be a banner coming up when the network adapter is set to passthrough successfully.



Passthrough has been enabled for MT27710 Family [ConnectX-4 Lx].

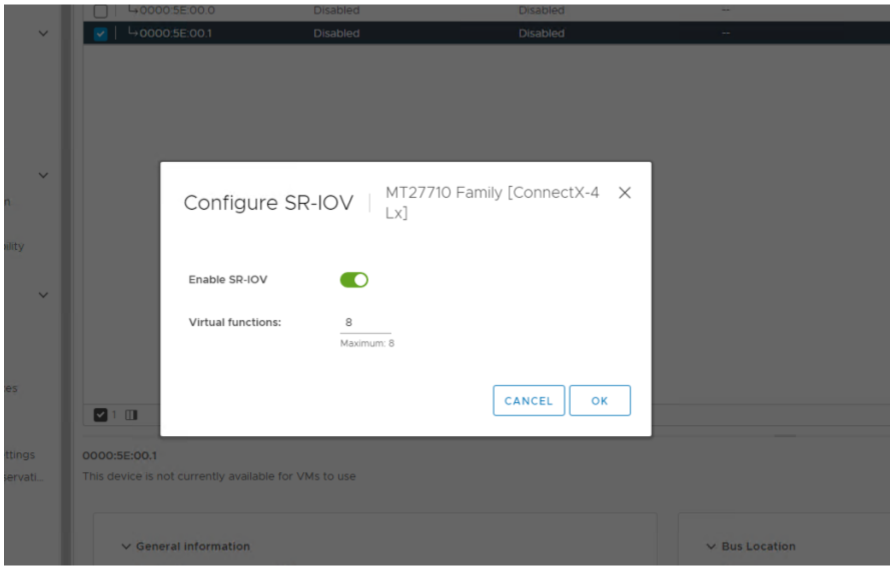
ID	Passthrough	SR-IOV	Hardware Label	Vendor Name	Device Name
↳0000:5E:00.0	Disabled	Disabled	--	Mellanox Technologies	MT27710 Family [ConnectX-4 Lx]
↳0000:5E:00.1	Enabled	Disabled	--	Mellanox Technologies	MT27710 Family [ConnectX-4 Lx]

Figure 31. Set NIC to Passthrough

Configure SR-IOV for a NIC

If the NVIDIA Network Adapter supports SR-IOV, it can be enabled on the network adapter from the vSphere Client. Enabling SR-IOV on the network adapter generates multiple Virtual Functions. Those Virtual Functions can behave like dedicated device functions while sharing the same network adapter, and each Virtual Function can be set to passthrough and then assigned to one virtual machine or Tanzu Kubernetes cluster worker node.

To enable SR-IOV on the network adapter, follow the steps above from step. 1 to step. 5 above. Then filter the ID using the PCI Bus ID found above (5e), make sure it is not being used by any ESX/ESXi Device, check the box on the left, and click **CONFIGURE SR-IOV**, in the pop-up box, switch the toggle to the **right** and set the number of Virtual Functions to generate and click **OK.** The max number of Virtual Functions are set in the firmware.



0000:5E:00.0	Disabled	Disabled
0000:5E:00.1	Disabled	Disabled

Configure SR-IOV | MT27710 Family [ConnectX-4 Lx]

Enable SR-IOV: on

Virtual functions: 8

Maximum: 8

CANCEL OK

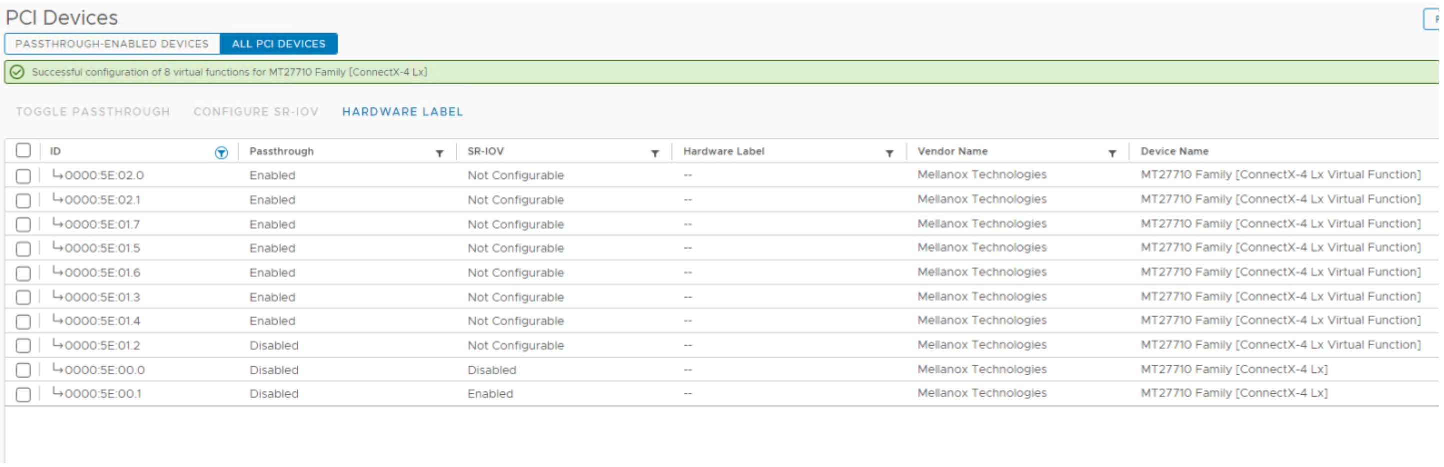
0000:5E:00.1

This device is not currently available for VMs to use

Figure 32. Configure SR-IOV for the NIC

NOTE: To be able to see the **Enable SR-IOV** option from vSphere client, the SR-IOV must be activated on the NVIDIA network adapter firmware, follow the *link* to activate SR-IOV if **Enable SR-IOV** option is not available from vSphere client.

There will be a banner coming up when the Virtual Functions are successfully configured, and by default, those Virtual Functions are already set to passthrough automatically.



PCI Devices

Successful configuration of 8 virtual functions for MT27710 Family [ConnectX-4 Lx]

ID	Passthrough	SR-IOV	Hardware Label	Vendor Name	Device Name
0000:5E:02.0	Enabled	Not Configurable	--	Mellanox Technologies	MT27710 Family [ConnectX-4 Lx Virtual Function]
0000:5E:02.1	Enabled	Not Configurable	--	Mellanox Technologies	MT27710 Family [ConnectX-4 Lx Virtual Function]
0000:5E:01.7	Enabled	Not Configurable	--	Mellanox Technologies	MT27710 Family [ConnectX-4 Lx Virtual Function]
0000:5E:01.5	Enabled	Not Configurable	--	Mellanox Technologies	MT27710 Family [ConnectX-4 Lx Virtual Function]
0000:5E:01.6	Enabled	Not Configurable	--	Mellanox Technologies	MT27710 Family [ConnectX-4 Lx Virtual Function]
0000:5E:01.3	Enabled	Not Configurable	--	Mellanox Technologies	MT27710 Family [ConnectX-4 Lx Virtual Function]
0000:5E:01.4	Enabled	Not Configurable	--	Mellanox Technologies	MT27710 Family [ConnectX-4 Lx Virtual Function]
0000:5E:01.2	Disabled	Not Configurable	--	Mellanox Technologies	MT27710 Family [ConnectX-4 Lx Virtual Function]
0000:5E:00.0	Disabled	Disabled	--	Mellanox Technologies	MT27710 Family [ConnectX-4 Lx]
0000:5E:00.1	Disabled	Enabled	--	Mellanox Technologies	MT27710 Family [ConnectX-4 Lx]

Figure 33. NIC VFs Generated and set to Passthrough

vSphere with Tanzu Kubernetes Grid Settings

Configure VM Service

Create VM Classes with vGPU and Passthrough NIC

Prerequisites:

- At least one RoCE capable network adapter is installed on the ESXi host that equipped with GPU.

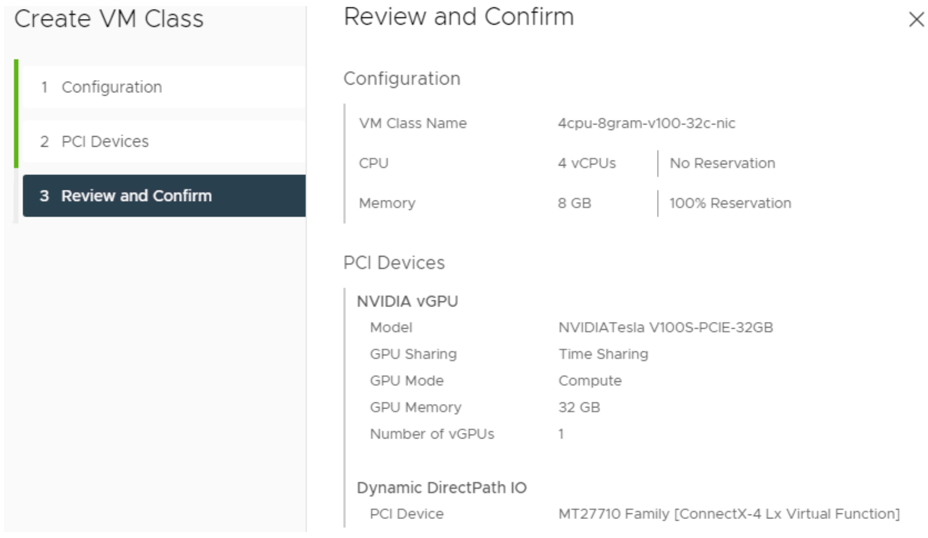
- The *network adapter is configured to passthrough*, if the network adapter has SR-IOV enabled, its Virtual Functions are configured to passthrough.
- *NVIDIA Virtual GPU Manager is installed* on all the ESXi host.

To create a VM Class configured with vGPU and passthrough NIC, follow the steps of Create VM Class with vGPU, but on the second page of the **VM Class** creation console, make the additional configurations:

1. On the second page of the **VM Class** creation dialog, after the vGPU is configured, expand the drop-down list **ADD PCI DEVICE** again and choose **Dynamic DirectPath IO**.

2. From the **PCI Devices** drop-down list in the **Dynamic DirectPath IO** box, the network adapter configured as passthrough will be listed, if the network adapter has SR-IOV configured and the Virtual Functions are set to passthrough, the Virtual Function will be listed too. Choose the network adapter and then click **NEXT** to review the settings.

NOTE: If the number or model of GPUs, or if the model or SR-IOV setting of the NICs installed on the ESXi hosts are not identical, double check on each of the ESXi hosts to ensure the VM Class configuration is compatible with at least one ESXi host.



Configuration

VM Class Name	4cpu-8gram-v100-32c-nic	
CPU	4 vCPUs	No Reservation
Memory	8 GB	100% Reservation

PCI Devices

NVIDIA vGPU

Model	NVIDIATesla V100S-PCIE-32GB
GPU Sharing	Time Sharing
GPU Mode	Compute
GPU Memory	32 GB
Number of vGPUs	1

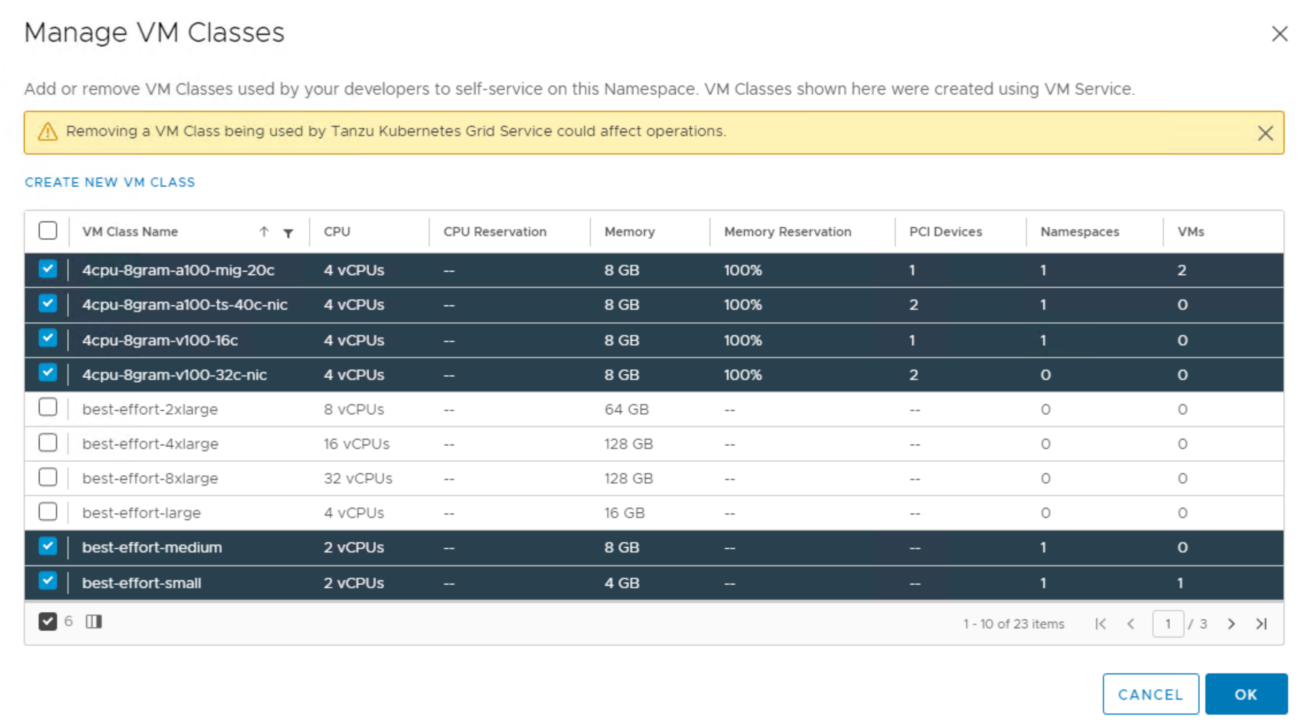
Dynamic DirectPath IO

PCI Device	MT27710 Family [ConnectX-4 Lx Virtual Function]

Figure 34. Review Configuration of the VM Class

Configure Namespace

Add the *VM Class with vGPU and passthrough NIC configured* to the **Namespace**, click **ADD VM CLASS** in the **VM Service** box under the Namespace **Summary** console, and check the newly created VM Classes.


Manage VM Classes

Add or remove VM Classes used by your developers to self-service on this Namespace. VM Classes shown here were created using VM Service.

Removing a VM Class being used by Tanzu Kubernetes Grid Service could affect operations.

CREATE NEW VM CLASS

	VM Class Name	CPU	CPU Reservation	Memory	Memory Reservation	PCI Devices	Namespaces	VMs
✓	4cpu-8gram-a100-mig-20c	4 vCPUs	--	8 GB	100%	1	1	2
✓	4cpu-8gram-a100-ts-40c-nic	4 vCPUs	--	8 GB	100%	2	1	0
✓	4cpu-8gram-v100-16c	4 vCPUs	--	8 GB	100%	1	1	0
✓	4cpu-8gram-v100-32c-nic	4 vCPUs	--	8 GB	100%	2	0	0
	best-effort-2xlarge	8 vCPUs	--	64 GB	--	--	0	0
	best-effort-4xlarge	16 vCPUs	--	128 GB	--	--	0	0
	best-effort-8xlarge	32 vCPUs	--	128 GB	--	--	0	0
	best-effort-large	4 vCPUs	--	16 GB	--	--	0	0
✓	best-effort-medium	2 vCPUs	--	8 GB	--	--	1	0
✓	best-effort-small	2 vCPUs	--	4 GB	--	--	1	1

Figure 35. Associate VM Class to the Namespace

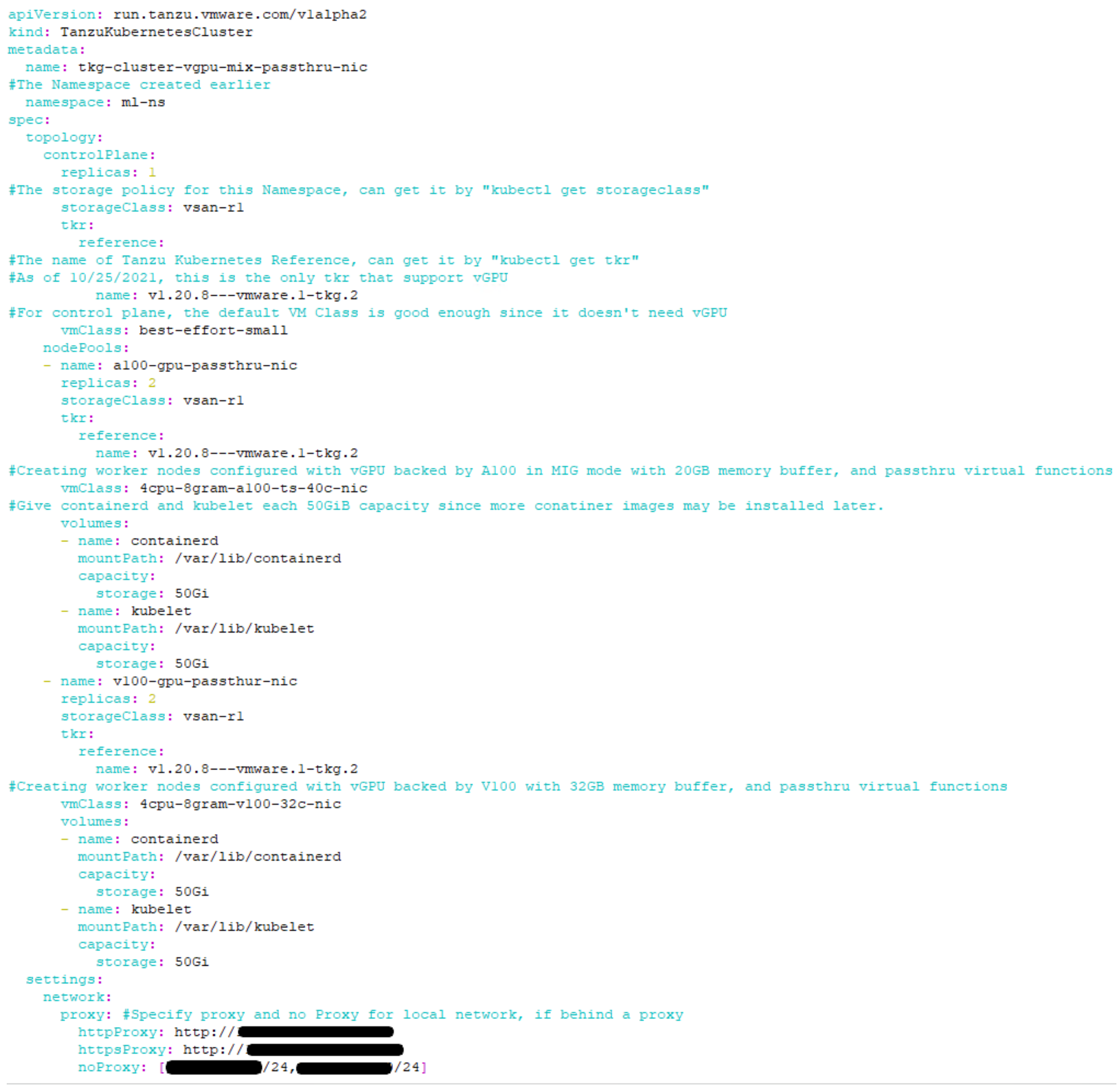
Provision Tanzu Kubernetes Grid Cluster with vGPU and Passthrough NIC

Prerequisites:

- *VM Class configured with vGPU and passthrough NIC* is created and added into the **Namespace**.
- The Tanzu Kubernetes Grid Content Library has downloaded image **ob-18807685-tkgs-ova-ubuntu-2004-v1.20.8---vmware.1-tkg.2**.

To create a Tanzu Kubernetes Grid Cluster, follow the following steps:

1. *Download and install the Kubernetes CLI Tools*.
2. Connect to the Supervisor Cluster using the tool installed: **kubectl,** follow *Configure Secure Login for vSphere with Tanzu Clusters* and *Connect to the Supervisor Cluster as a vCenter Single Sign-On User*.
3. Use $kubectl config use-context <example-context-name> to switch to the **Namespace** created *earlier*.
4. Follow the instruction *here* to prepare the Tanzu Kubernetes Grid Cluster YAML file, and refer to this *example*. Make sure to use **Tanzu Kubernetes Grid Service v1alpha2 API.** There are some YAML file examples on *Github*.

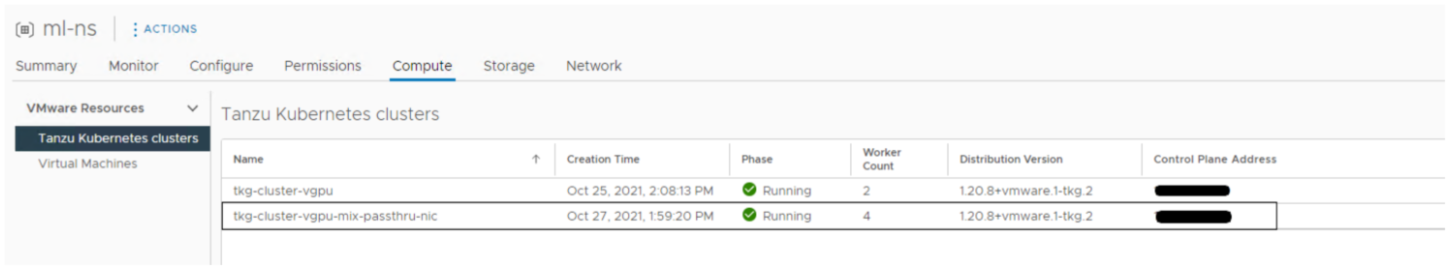


```
apiVersion: run.tanzu.vmware.com/v1alpha2
kind: TanzuKubernetesCluster
metadata:
  name: tkg-cluster-vgpu-mix-passthru-nic
#The Namespace created earlier
  namespace: ml-ns
spec:
  topology:
    controlPlane:
      replicas: 1
#The storage policy for this Namespace, can get it by "kubectl get storageclass"
      storageClass: vsan-r1
      tkr:
        reference:
#The name of Tanzu Kubernetes Reference, can get it by "kubectl get tkr"
#As of 10/25/2021, this is the only tkr that support vGPU
          name: v1.20.8---vmware.1-tkg.2
#For control plane, the default VM Class is good enough since it doesn't need vGPU
      vmClass: best-effort-small
    nodePools:
    - name: a100-gpu-passthru-nic
      replicas: 2
      storageClass: vsan-r1
      tkr:
        reference:
          name: v1.20.8---vmware.1-tkg.2
#Creating worker nodes configured with vGPU backed by A100 in MIG mode with 20GB memory buffer, and passthru virtual functions
      vmClass: 4cpu-8gram-a100-ts-40c-nic
#Give containerd and kubelet each 50GiB capacity since more conatiner images may be installed later.
      volumes:
      - name: containerd
        mountPath: /var/lib/containerd
        capacity:
          storage: 50Gi
      - name: kubelet
        mountPath: /var/lib/kubelet
        capacity:
          storage: 50Gi
    - name: v100-gpu-passthur-nic
      replicas: 2
      storageClass: vsan-r1
      tkr:
        reference:
          name: v1.20.8---vmware.1-tkg.2
#Creating worker nodes configured with vGPU backed by V100 with 32GB memory buffer, and passthru virtual functions
      vmClass: 4cpu-8gram-v100-32c-nic
      volumes:
      - name: containerd
        mountPath: /var/lib/containerd
        capacity:
          storage: 50Gi
      - name: kubelet
        mountPath: /var/lib/kubelet
        capacity:
          storage: 50Gi
  settings:
    network:
      proxy: #Specify proxy and no Proxy for local network, if behind a proxy
        httpProxy: http://[redacted]
        httpsProxy: http://[redacted]
        noProxy: [[redacted]/24,[redacted]/24]
```

Figure 36. Tanzu Kubernetes Grid Cluster Definition with vGPU and Passthrough NIC

5. Run $kubectl apply -f <tkg_cluster_yaml_file_name> to provision the Tanzu Kubernetes cluster.

6. After a few minutes, run $kubectl get tkc <tkg_cluster_name> to ensure the cluster is ready, or this could be checked from vSphere web client by choosing the **Namespace**, and navigating it to **Compute**, the cluster information will be shown under **Tanzu Kubernetes clusters**.

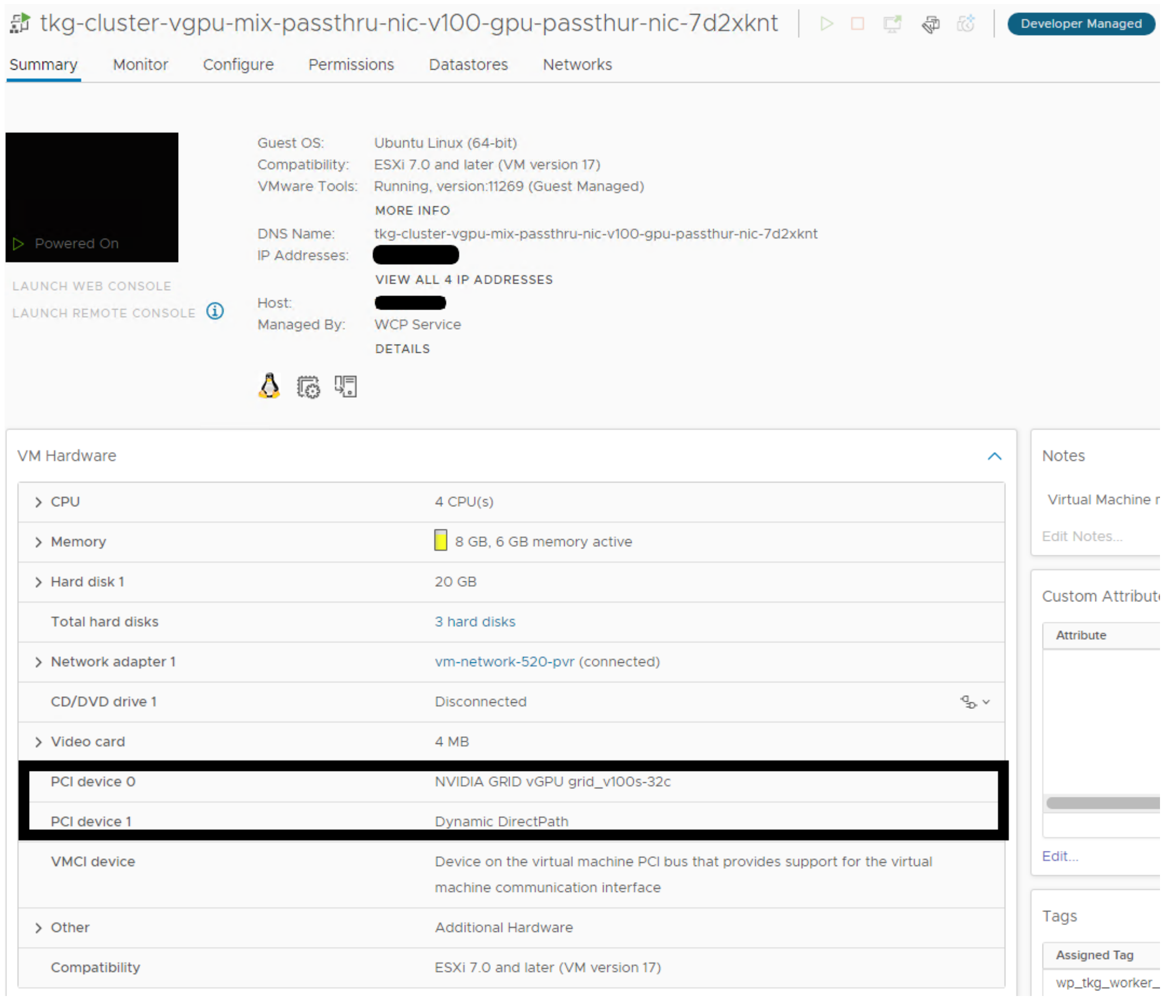


Tanzu Kubernetes clusters

Name	Creation Time	Phase	Worker Count	Distribution Version	Control Plane Address
tkg-cluster-vgpu	Oct 25, 2021, 2:08:13 PM	Running	2	1.20.8+vmware.1-tkg.2	[redacted]
tkg-cluster-vgpu-mix-passthru-nic	Oct 27, 2021, 1:59:20 PM	Running	4	1.20.8+vmware.1-tkg.2	[redacted]

Figure 37. Tanzu Kubernetes Grid Cluster with vGPU and Passthrough NIC Created

7. The Tanzu Kubernetes cluster with four worker nodes is provisioned, two worker nodes are configured with vGPU backed by an A100 and a passthrough NIC Virtual Function; another two are configured with a vGPU backed by V100 and a passthrough NIC Virtual Function.



Guest OS:	Ubuntu Linux (64-bit)
Compatibility:	ESXi 7.0 and later (VM version 17)
VMware Tools:	Running, version:11269 (Guest Managed)
DNS Name:	tkg-cluster-vgpu-mix-passthru-nic-v100-gpu-passthur-nic-7d2xknt
IP Addresses:	[redacted]
Host:	[redacted]
Managed By:	WCP Service

VM Hardware

CPU	4 CPU(s)
Memory	8 GB, 6 GB memory active
Hard disk 1	20 GB
Total hard disks	3 hard disks
Network adapter 1	vm-network-520-pvr (connected)
CD/DVD drive 1	Disconnected
Video card	4 MB
PCI device 0	NVIDIA GRID vGPU grid_v100s-32c
PCI device 1	Dynamic DirectPath
VMCI device	Device on the virtual machine PCI bus that provides support for the virtual machine communication interface
Other	Additional Hardware
Compatibility	ESXi 7.0 and later (VM version 17)

Figure 38. Worker Nodes Configured with vGPU and Passthrough NIC

Next Step

When the Tanzu Kubernetes Grid Cluster configured with vGPU and passthrough NIC is ready, log out from the Namespace by $kubectl logout and *re-login into the Tanzu Kubernetes Grid Cluster*, then follow the instructions to install the *NVIDIA GPU Operator* and the *NVIDIA Network Adapter Operator* with NVIDIA AI Enterprise and then run GPU applications.

Create a Stand-alone VM with vGPU and Passthrough NIC

Pre-requisites:

1. Download **ubuntu-20-1633387172196.ova** or **centos-stream-8-vmservice-v1alpha1-1633387532874.ova** from *VMware Marketplace* and import it to the *local content library*.
2. The *local content library* has been added to the VM Service
3. *VM Class with vGPU and passthrough NIC is created* and *associated with the Namespace*.

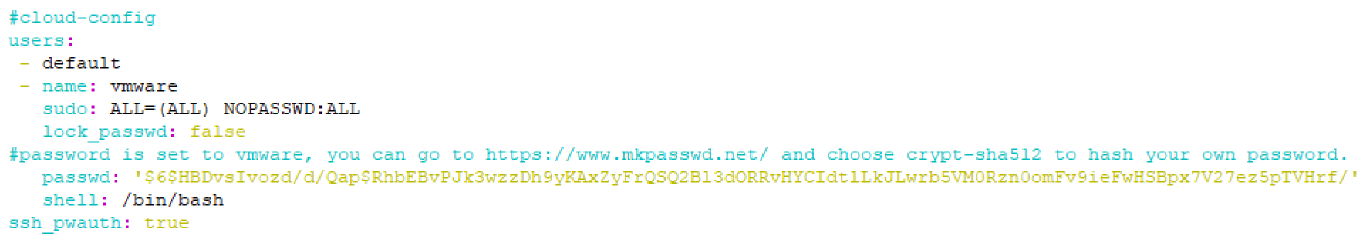
Follow the instruction to *Deploy a Virtual Machine with vGPU in vSphere with Tanzu*. When preparing the VM YAML file, the name of *VM Class with vGPU and passthrough NIC configured* should be specified to **spec.className;** and the name of the VM image imported to the local content library should be used for **spec.imageName**.

```
#File name: vm-svc-gpu-nic.yaml
apiVersion: vmoperator.vmware.com/v1alpha1
kind: VirtualMachine
metadata:
#Specify the name of VM to create and the namespace to be used.
  name: ubuntu-vgpu-v100-32c-nic
  namespace: ml-ns
spec:
#Specify the VM image name, which can be list by 'kubectl get vmimage'
  imageName: ubuntu-20-1633387172196
#Specify the name of VM class with vGPU, which can be list by 'kubectl get vmclass'
#Run 'kubectl get vmclass VM_CLASS_NAME -o yaml' to verify it has vGPU configured
  className: 4cpu-8gram-v100-32c-nic
  powerState: poweredOn
  storageClass: vsan-r1
  networkInterfaces:
  - networkName: "vm-network-520-pvr"
    networkType: vsphere-distributed
  vmMetadata:
      configMapName: vmsvc-ubuntu-vgpu-cm
      transport: OvfEnv
---
apiVersion: v1
kind: ConfigMap
metadata:
    name: vmsvc-ubuntu-vgpu-cm
    namespace: ml-ns
data:
  user-data: I2Nsb3VkLWNvbmZpZwplc2VyczoKIC0gZGVmYXVsdAogLSBuYW1lOiB2bXdhcmUgCiAgIHN1ZG86IEFMT
N1YyN2V6NXBUVkhyZi8nCiAgIHNoZWxsOiAvYmluL2Jhc2gKc3NoX3B3YXV0aDogdHJ1ZQo=
```

Figure 39. Stand-alone VM with vGPU and Passthrough NIC Definition

The value of **data.user-data** is base64 encoded cloud-config file, by running $cat <cloud-config_yaml_file> | base64 -w 0

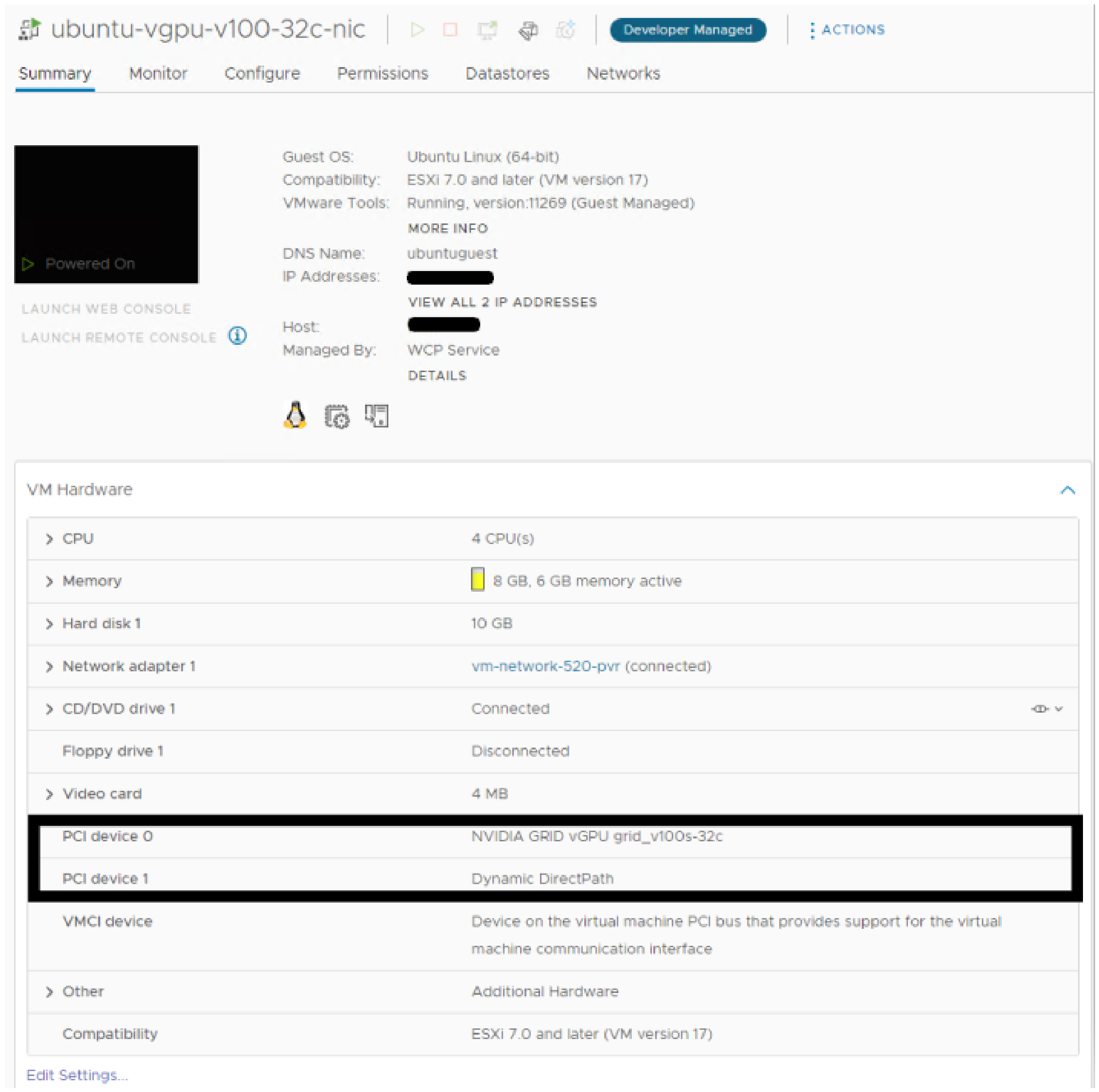
In the cloud-config.yaml, both the username and password have been set to **vmware** in the example below to access the VM via SSH. The YAML examples can be found on *Github*.



```
#cloud-config
users:
 - default
 - name: vmware
   sudo: ALL=(ALL) NOPASSWD:ALL
   lock_passwd: false
#password is set to vmware, you can go to https://www.mkpasswd.net/ and choose crypt-sha512 to hash your own password.
   passwd: '$6$HBDvsIvozd/d/Qap$RhbEBvPJk3wzzDh9yKAxZyFrQSQ2Bl3dORRvHYCIdtlLkJLwrb5VM0Rzn0omFv9ieFwHSBpx7V27ez5pTVHrf/'
   shell: /bin/bash
ssh_pwauth: true
```

Figure 40. Stand-alone VM User-data Definition

Run $kubectl apply -f <vm_yaml_file_name> to create the stand-alone VM, wait for a few minutes, and run $kubectl get vm <vm_name> -o yaml | grep vmIp to ensure the VM is running, and the IP address is ready. From vSphere client, it can be seen the Virtual Machine configured with vGPU is successfully created and running.



Guest OS:	Ubuntu Linux (64-bit)
Compatibility:	ESXi 7.0 and later (VM version 17)
VMware Tools:	Running, version:11269 (Guest Managed)
DNS Name:	ubuntuguest
IP Addresses:	[redacted]
Host:	[redacted]
Managed By:	WCP Service

CPU	4 CPU(s)
Memory	8 GB, 6 GB memory active
Hard disk 1	10 GB
Network adapter 1	vm-network-520-pvr (connected)
CD/DVD drive 1	Connected
Floppy drive 1	Disconnected
Video card	4 MB
PCI device 0	NVIDIA GRID vGPU grid_v100s-32c
PCI device 1	Dynamic DirectPath
VMCI device	Device on the virtual machine PCI bus that provides support for the virtual machine communication interface
Other	Additional Hardware
Compatibility	ESXi 7.0 and later (VM version 17)

Figure 41. Worker Nodes Configured with vGPU and Passthrough NIC

Next Step

When the stand-alone VM configured with vGPU and passthrough NIC is ready:

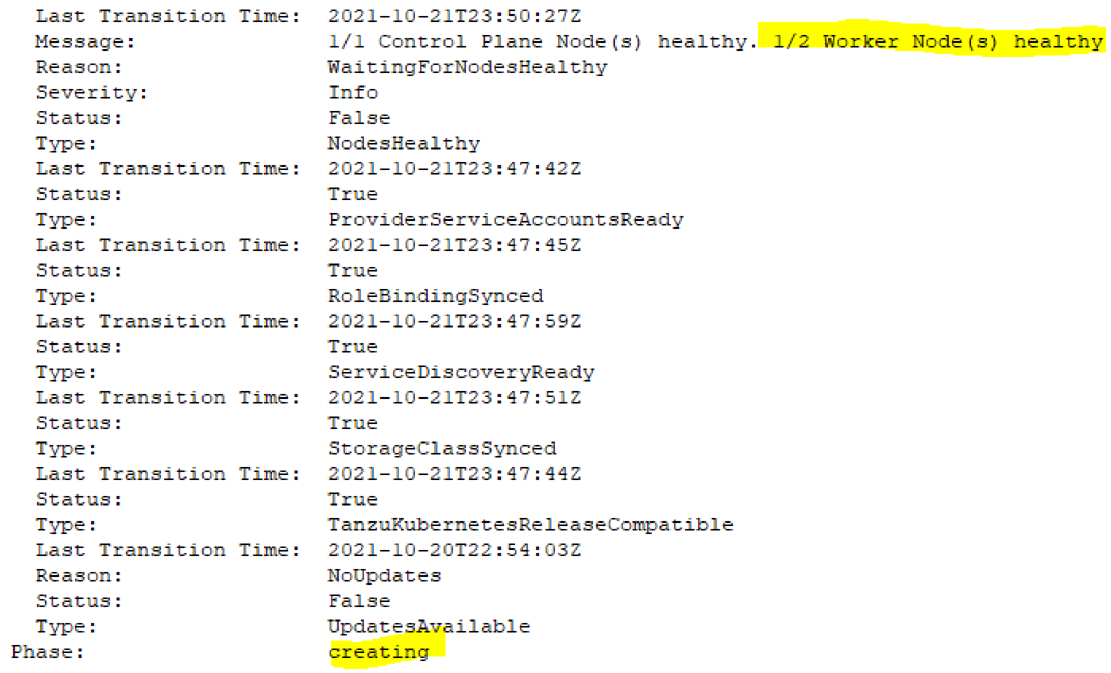
1. The VM created can be accessed by SSH. Follow the *instruction* to install the NVIDIA vGPU Driver.
2. The VM created can be accessed by SSH. Install the Network Adapter driver.
3. The VM created can be accessed by SSH. Install the machine learning applications.

Troubleshooting

Symptom: Failed to Create a Tanzu Kubernetes Grid Cluster or Stand-alone VM with a vGPU profile or Passthrough NIC

When provisioning a Tanzu Kubernetes Grid Cluster or stand-alone VM that is configured with vGPU, the provisioning may fail due to insufficient GPU resources. In this case, the provisioning of the Tanzu Kubernetes Grid Cluster or stand-alone VM may be hanging. This can be checked by running $kubectl get tkc <tkg_cluster_name> , if the **AGE** is more than 10 minutes and the **READY** is still false, then the Tanzu Kubernetes Grid Cluster should be further investigated. Follow the steps below to do so. If provisioning a stand-alone VM is hanging, go to step. 3 directly:

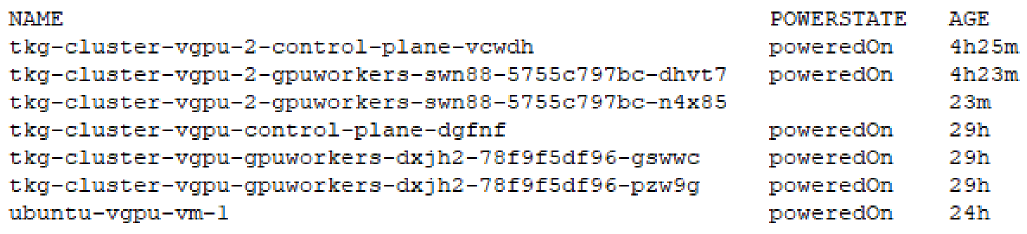
1. Run $kubectl describe tkc <tkg_cluster_name> to check if the **Phase** is still **Creating** and how many worker nodes are available.



```
    Last Transition Time:  2021-10-21T23:50:27Z
    Message:               1/1 Control Plane Node(s) healthy. 1/2 Worker Node(s) healthy
    Reason:                WaitingForNodesHealthy
    Severity:              Info
    Status:                False
    Type:                  NodesHealthy
    Last Transition Time:  2021-10-21T23:47:42Z
    Status:                True
    Type:                  ProviderServiceAccountsReady
    Last Transition Time:  2021-10-21T23:47:45Z
    Status:                True
    Type:                  RoleBindingSynced
    Last Transition Time:  2021-10-21T23:47:59Z
    Status:                True
    Type:                  ServiceDiscoveryReady
    Last Transition Time:  2021-10-21T23:47:51Z
    Status:                True
    Type:                  StorageClassSynced
    Last Transition Time:  2021-10-21T23:47:44Z
    Status:                True
    Type:                  TanzuKubernetesReleaseCompatible
    Last Transition Time:  2021-10-20T22:54:03Z
    Reason:                NoUpdates
    Status:                False
    Type:                  UpdatesAvailable
  Phase:                   creating
```

Figure 42. Check Tanzu Kubernetes Grid Cluster Status

2. If the cluster is still in **Creating** phase and not all the worker nodes are available, run $kubectl get vm to find out which worker nodes (VMs) are not available.



```
NAME                                                  POWERSTATE   AGE
tkg-cluster-vgpu-2-control-plane-vcwdh                poweredOn    4h25m
tkg-cluster-vgpu-2-gpuworkers-swn88-5755c797bc-dhvt7  poweredOn    4h23m
tkg-cluster-vgpu-2-gpuworkers-swn88-5755c797bc-n4x85               23m
tkg-cluster-vgpu-control-plane-dgfnf                  poweredOn    29h
tkg-cluster-vgpu-gpuworkers-dxjh2-78f9f5df96-gswwc    poweredOn    29h
tkg-cluster-vgpu-gpuworkers-dxjh2-78f9f5df96-pzw9g    poweredOn    29h
ubuntu-vgpu-vm-1                                      poweredOn    24h
```

Figure 43. One Worker Node Not Powered On

3. After finding the VM(s) that are not powered on, run $kubectl describe vm <VM_name> to check the details of the problematic VM. At the bottom of the **Events** section, the "**No host is compatible with the virtual machine**." section means that the **Supervisor Cluster** is running out of GPU resources.

4. In this case, talk to the vSphere administration team to release some GPU resources or reduce the size of the Tanzu Kubernetes Grid Cluster or stand-alone VM.

Reference

- *vSphere 7 Update 3a*

- *NVIDIA vGPU*
- *NVIDIA vGPU User Guide*
- *Multi-Instance GPU*
- *NVIDIA AI Enterprise Software Suite*

About the Authors

Chen Wei, Staff Solutions Architect in the VMware Cloud Infrastructure Business Group, wrote this document with contributions from the following members:

- Justin Murray, Staff Technical Marketing Architect, VMware
- Catherine Xu, Manager of Application Solutions Architecture team in the Cloud Infrastructure Business Group, VMwar