SUSE Rancher Stack

NVIDIA DGX Testing and Deployment Guide



Table of Contents

Obiective	3
Purpose	3
Background	3
Test Environment	3
Rationalization:	3
Virtualized Infrastructure:	3
Management Node:	3
Worker Nodes:	4
Tasks and Test Elements Demonstration	4
Build Rancher Kubernetes Management Cluster (pre-requisite step)	4
Install and deploy RKE2 cluster as worker nodes.	6
Deploy NVIDIA GPU operator and enable Multi-Instance GPU (MIG)	9
Install and deploy Longhorn (persistent storage).	12
Deploy accelerated workload on NVIDIA DGX worker nodes	13
Wrap-Up	. 16

Objective

Purpose

This document outlines installation and test steps completed by SUSE to certify the successful deployment of its Enterprise Container Management (ECM) software stack with the NVIDIA DGX as a Kubernetes target node that can be used for the deployment of accelerated workloads leveraging NVIDIA GPUs. SUSE Enterprise Container Management products deployed during the testing and validation process include:

- Rancher Kubernetes Management
- Rancher RKE2 Kubernetes distribution
- Longhorn Kubernetes Persistent Storage

As part of this qualification effort, the following tests were completed:

Test/Task	Complete Y/N
Build Rancher Cluster (pre-requisite)	Y
Install and deploy RKE2 cluster as worker nodes	Y
Deploy NVIDIA GPU operator and enable Multi-Instance GPU	Y
Install and deploy Longhorn (persistent storage)	Y
Deploy accelerated workload on NVIDIA DGX worker nodes	Y

Table 1. Test/qualification checklist.

The rest of this paper will document the steps followed and results obtained while completing the tasks outlined in Table 1.

Background

As NVIDIA states in its own <u>website</u>, "The NVIDIA DGX[™]-Ready Software program features enterprise-grade MLOPs solutions that accelerate AI workflows and improve deployment, accessibility, and utilization of AI infrastructure. **DGX-Ready Software is tested and certified for use on DGX systems, helping you get the most out of your AI platform investment**."

Test Environment

Rationalization:

Since the purpose of the environment is to demonstrate the successful integration of the NVIDIA DGX into an existing Kubernetes cluster, we kept the rest of the infrastructure to a minimal configuration that would be easy to reproduce by anyone else having the following:

- At least one physical node that would support virtualization for the non-DGX components.
- The NVIDIA DGX system itself.

Virtualized Infrastructure:

SUSE's <u>Harvester</u> hyperconverged infrastructure solution. Harvester is built on Kubernetes and utilizes the latest cloud-native solutions including LongHorn and Kubevirt.

With this setup, we created four virtual machines:

- One virtual machine to host the Rancher management server.
- Three virtual machines to create an RKE2 cluster acting as part of the worker node infrastructure.

Management Node:



We kept it simple and used a single node and followed the instructions as outlined under the **Deploying Rancher Server** guide under the *manual install* section (which addresses onprem setups). The virtual machine's OS was <u>SUSE Linux Enterprise Micro</u>.

Worker Nodes:

Three virtual machines used to build an RKE2 cluster with <u>SUSE Linux Enterprise Micro</u> used as their OS.

Two NVIDIA DGX (physical nodes) added to the Kubernetes cluster for deployment of GPUbased workloads.

Tasks and Test Elements Demonstration

Build Rancher Kubernetes Management Cluster (pre-requisite step)

Having a working Rancher Kubernetes Management cluster to complete all other activities is a pre-requisite. A discussion on best practices for building and deploying the Rancher cluster itself is beyond the scope of this document. A good starting point for those interested would be the Rancher documentation at: <u>https://ranchermanager.docs.rancher.com/</u>

In our scenario, we opted to install SUSE's Harvester Hyperconverged Infrastructure Solution and deploy virtual machines that would support the Rancher Kubernetes Management cluster. Visit <u>https://docs.harvesterhic.io/</u> for installation and configuration information.

Steps followed:

• Build four virtual machines – One virtual machine for the Rancher management environment and three virtual machines to act as worker nodes in an RKE2 cluster.

Harvester							All Namespaces			· 🔍
Dashboard Hosts		Virtual Ma	chines							Create
Virtual Machines										
Volumes		× Stop	O Restart	⊥ Download YAML	🛎 Delete	È	=	Filter		
Images										
Namespaces		○ State ♀	Name 🖓		CPU 🗘	Memory 🗘	IP Address	Node ♡	Age♀	
Networks	~									
Backup & Snapshot	~	Namespace: def	ault							
Monitoring & Logging	~	Running	test5	Console	✓ 4	16 Gi	10.141.0.176	harv-1	5 days	1
Adjunced	~	Running	test4	Console	~ 4	32 Gi	10.10.0.220 (210.141.0.173 (2	harv-1	61 days	÷
		Running	test3	Console	~ 4	32 Gi	10.10.0.219 🗍 10.141.0.175 📄	harv-1 📴	61 days	÷
		Running	test2	Console	✓ 4	32 Gi	10.10.0.218 🗍 10.141.0.174 🗍	harv-1 📴	61 days	÷
		Running	test1	Console	v 12	64 Gi	10.10.0.217 (210.141.0.172 (2	harv-1 📴	62 days	÷

Figure 1. Virtual Machines used to deploy management nodes and management cluster components.

- Install <u>SUSE Linux Enterprise Micro</u> on the virtual machines. Version 5.4 of the product was used as the base OS on all virtual machines. This lightweight operating system is purposely built for containerized and virtual workloads. It leverages the enterprise hardened security and compliance of SUSE Linux Enterprise and merges them with a modern, immutable, developer-friendly OS platform.
- For more information on installing and managing SUSE Linux Enterprise Micro, please review:
 - o <u>SUSE Linux Enterprise Micro Deployment Guide</u>
 - o SUSE Linux Enterprise Micro Administration Guide
- Install Rancher Management on virtual machines. There are several deployment options for Rancher Kubernetes Management platform. For the purposes of this evolution, we chose <u>Helm CLI Quick Start</u>.
- Install the *helm* package.

- Note: if the PackagHub repo is not activated, enable it with: SUSEConnect -p PackageHub/15.4/x86_64
- Helm installation as transactional update on SUSE Linux Enterprise Micro:
 # transactional-update pkg install helm-3.8.0-bp154.2.27
- Install K3s on Linux (SUSE Linux Enterprise Micro):
 - # curl -sfL <u>https://get.k3s.io</u> | INSTALL_K3S_VERSION="v1.24.14+k3s1"
 INSTALL_K3S_SKIP_SELINUX_RPM=true INSTALL_K3S_EXEC='server --cluster-init
 --write-kubeconfig-mode=644' sh -s -
 - # k3s kubectl get nodes

test1:~	# k3s ku	bectl get nodes		
name	STATUS	ROLES	AGE	VERSION
test1	Ready	control-plane,etcd,master	15s	v1.24.14+k3s1

- # kubectl apply --validate=false -f <u>https://github.com/cert-manager/cert-manager/releases/download/v1.11.0/cert-manager.crds.yaml</u>
- o # helm repo add jetstack https://charts.jetstack.io
- *# helm repo update*
- # export KUBECONFIG=/etc/rancher/k3s/k3s.yaml
- # helm install cert-manager jetstack/cert-manager --namespace certmanager --create-namespace --version v1.11.0
- # kubectl get pods --namespace cert-manager

test1:~ # kubectl get podsnamespace cer	t-manage	r		
NAME	READY	STATUS	RESTARTS	AGE
cert-manager-85945b75d4-g5qgq	1/1	Running	0	59s
cert-manager-cainjector-7f694c4c58-11s88	1/1	Running	0	59s
cert-manager-webhook-7cd8c769bb-6px5t	1/1	Running	0	59s

- Install Rancher over K3s installation:
 - # helm repo add rancher-stable <u>https://releases.rancher.com/server-</u> <u>charts/stable</u>
 - # export HOSTNAME="test1.eth.cluster"
 - # export RANCHER_VERSION="2.7.3"
 - # kubectl create namespace cattle-system
 - # helm install rancher rancher-stable/rancher --namespace cattle-system --set hostname=test1.eth.cluster --set version=2.7.4 --set replicas=1
- Go to Rancher server URL and login:

	Howdy!	
t looks like this is v	your first time visiting Rancher; if you pre-set your own	
pootstrap passwor generated for you.	d, enter it here. Otherwise a random one has been To find it:	
For a "docker run"	installation:	
• Find your co	ontainer ID with docker ps , then run:	
• Password	ogs <u>container-id</u> 2>&1 grep "Bootstrap :"	
For a Helm installa	tion, run:	
<pre>kubectl get s bootstrap-sec template='{{ {{"\n"}}'</pre>	secretnamespace cattle-system cret -o go- .data.bootstrapPassword base64decode}}	
	k	
	Password Show	Y

Figure 2. Rancher login page and URL after installation.

- Run the command from the "For a Helm installation run:" to show generated to enter and create a new one.
- Since Harvester was used as the virtual machine platform, you can integrate Rancher with Harvester as described in: Virtualization Management.

Virtualization Management	ent		
Harvester Clusters	Harvester Cluster:	Create	
	Cluster Name * harv-test-cluster	Cluster Description Any text you want that better describ	
	Member Roles Agent Environment Vars Labels & Annotations	Member Roles User Default Admin (admin) Local	Role Cluster Owner

Figure 3. Harvester and Rancher integration.

Virtualization Man	agement				:	2
Harvester Clusters	Harvester Cluste	rs			Import Exis	ting
	🛓 Download KubeConfig	t Download YAML	🛍 Delete			
	□ State ≎ Name \$	Version 0	Machines	Age 🗘		
	Active harv-te	st-cluster v1.24.11+rke	2r1 1	3.6 mins	Manage	:

Figure 4. Harvester and Rancher integration.

At the end of this section, we have a fully working Rancher Kubernetes cluster.

Test/Task	Complete Y/N
Build Rancher Cluster (pre-requisite)	Υ
Install and deploy RKE2 cluster as worker nodes	Ν
Deploy NVIDIA GPU operator and enable Multi-Instance GPU	Ν
Install and deploy Longhorn (persistent storage)	Ν
Deploy accelerated workload on NVIDIA DGX worker nodes	Ν

Install and deploy RKE2 cluster as worker nodes.

Once a Rancher Kubernetes Management installation is provisioned, we use it to create an RKE2 cluster.

• From the Rancher console, go to Cluster Management, select RKE2 and click "Custom".



Cluster Manage	ement			÷ 😃
Clusters	2	Provision new nodes and create a cl	uster using RKE2/K3s	RKE1 RKE2/K3s
Cloud Credentials				
Drivers				
RKE1 Configuration	~	Amazon EC2	Azure	DigitalOcean
Advanced	~			
		Harvester	Linode	VMware vSphere
		Use existing nodes and create a clus	ster using RKE2/K3s	
*		Custom		

Figure 5. Rancher Cluster Management View.

• Select a proper/certified Kubernetes version and a cloud provider. In our test setup, RKE2 embedded was selected.

 Clusters Cloud Credentials 	2	Cluster Name* rke2-testcluster	Cluster Description Any text you want that better describes this o	
Orivers RKE1 Configuration	~	Cluster Configuratio	on	
Advanced ~	Basics	Basics		
		Add-On Config Agent Environment Vars etcd Labels & Annotations Networking	Kubernetes Version v1.24.13+rke2r1 Show deprecated Kubernetes patch versions Container Network multus,calico	Cloud Provider Default - RKE2 Embedded
	k	Registries Upgrade Strategy Advanced	Security Default Pod Security Policy unrestricted	CIS Profile (None)
			Pod Security Admission Configuration Template	
074			Cance	Edit as YAML Create

Figure 6. Cluster configuration sample – RKE2.



As additional nodes need to be registered, select <CREATE> (see figure 6), select Node • Role and copy the registration command (see figure 7).

Cluster Manag	gement	:
Clusters	3	
Cloud Credentials		Step 1
RKE1 Configuration	~	Node Role
Advanced	~	Choose what roles the node will have in the cluster. The cluster needs to have at least one node with each role.
		🗹 etcd 🗹 Control Plane 🗹 Worker
		Show Advanced Step 2
	k	Registration Command
		Run this command on each of the existing Linux machines you want to register.
		curl -fL https://test1.eth.cluster/system-agent-install.sh sudo sh -s - server https://test1.eth.clusterlabel 'cattle.io/os=linux'token 7d5jrln15cjrd8tm8nj4g8zv8ps91k5v9c5rhjdf74x8w2cvzcwrzvca-checksum 112bf557d4d5aab04fe99784af0455ac39cbeeb84242ab12a5cef2b2ebacb7b5etcd controlplaneworker
		Incomuna Calent this to also TI Curaifonation if your comunations a solf signed costificante

Figure 7. Registration command to be copied.

- Paste to the target node.
- When complete, ensure that you have an odd number of nodes in the RKE2 cluster. In our sample, we have a total of three nodes with all three roles assigned to them.

Cluster Manageme	ent								:
Cloud Credentials Cloud Credentials Drivers RKE1 Configuration Advanced	3 Na • De Pr	Cluster: rke2-t amespace: fleet-defaul escription: Nvidia DG> rovisioner: RKE2	t Age: 24 secs			Exp	plore	Detail	Config
*		Machines Provis	ioning Log Registrat ML Delete Name	ion Snap	shots Condi ernal/Internal IP	tions Recen	os≎ R	Related I Filter Roles \\$ Age \\$	Resources
		Running	custom-3dda79f89681	test3 10.	10.0.219 📋/Sa	me as External	Linux A	All 5 min	s :
		Running	custom-91a2e7cd6229	test4 10.	10.0.220 📴 / Sa	me as External	Linux A	All 2.8 m	ins :
		Running	custom-dbd86386dbfe	test2 10.	10.0.218 📴 / Sa	me as External	Linux A	All 12 mi	ns :

Figure 8. Three-node RKE2 cluster view.

- Adding NVIDIA DGX nodes to the RKE2 cluster: From the Rancher deployment, select the role to be applied to the NVIDIA DGX nodes. In this instance: 'worker'. Copy the command shown in Figure 7 (for RKE2 registration) and paste/use on the DGX nodes.
- When complete, verify that all nodes were deployed to the cluster. A successful completion can be verified via the Rancher console or from the command line.



← → G	U Ca	nttps://test	i.etn.clu	uster/dashb	ooard/	/c/c-m·	-ttw2nk8r	r/explo	rer/n	ode		67	% Y	3		⊚	≚	=
= 💦 rke2-test								opni X	8			~	Ť	٤	• 0	Q	:	ž
Cluster ^	Nodes 🕸																	
O Projects/Namespaces	11 Cordon	O Drain	± Dow	nload YAML	# De	lete												
Nodes 5																		
© Cluster and Project Members to Events 1	◯ State≎	Name Q	Roles 0	Version 0		External/li	nternal IP 🗘		oso	CPUO		RAMO		Pods		Age 🗘		
Workloads ~	C (Active)	basepod-dgx03	Worker	v1.24.13+rke2r	1	-/10.10.0	213 📋		Linux	n/a	10	n/a		•	219	22 days		÷
Apps 🗸		Labels: feature.	ode.kubernete	sio/cpu-cpuid.ADX+t	true feat	ture.node.kul	bernetesio/cpu-cp	uid AESNI-tr	ue feat	ture.node.kub	ernetes.ia/c	pu-cpuic	LAVX-tru	8				
Service Discovery		feature.node.kub	ernetes.io/cpu-	cpuidAVX2-true fo	leature.nod	ie kubernete	s.io/cpu-cpuid.CLZ	ZERO-true	feature.n	iode.kubernet	es.io/cpu-cp	uid.CP9	00ST-tru	e				
Storage Y	(Active)	basenod-dex04	Worker	v1 24 13+rke2r	1	10.10.0.21	4 [r] / Same as	s External	Linux	n/a		n/a			239	55 days		
Monitoring Y		Labele Dev en	and the second												20/	and and a		25
Longhorn		feature.node.kub	emetes.io/cpu-	cpuid.AVX2-true fi	eature.nod	ie kubernete	s io/cpu-cpuid.CLZ	ZERO-true	featuren	ode.kubernet	esture.node es.io/cpu-cp	uid.CPB	005T=tru	e Show	more			
More Resources ~	(Active	test2	All	v1.24.13+rke2r	1	10.10.0.21	18 [] / Same as	s External	Linux	n/a		n/a		-	359	55 days		÷
		Labels: features	ode.kubernete	sio/cpu-cpuid.ADX+t	true feat	ture.node.kul	bernetes.io/cpu-cp	uid.AESNI-tr	ue feat	ture.node.kub	ernetes.io/c	pu-cpuid	AVX-tru					
		feature.node.kub feature.node.kub	ernetes.ia/cpu-i ernetes.ia/cpu-i	cpuid.AVX2-true fr cpuid.HYPERVISOR-	true Sho	ie kubernete ow more	s.io/cpu-cpuid.CLZ	ZERO-true	featuren	ode.kubernet	es.io/cpu-cp	uid.FM/	3-true					
k	Active	test3	All	v1.24.13+rke2r	1	10.10.0.21	19 🗇 / Same as	s External	Linux	n/a		n/a			30%	55 days		÷
		Labels: feature.	ode.kubernete ernetes.io/cpu-i	sio/cpu-cpuid.ADX=t cpuid.AVX2=true fo	true feat	ture.node.kub še.kubernete	bernetes.io/cpu-cp s.io/cpu-cpuid.CLZ	uid.AESNI-tri ZERO-true	ue feat	ture.node.kub iode.kubernet	ernetes.ia/c es.io/cpu-cp	pu-cpuic uid.FMA	LAVX-true	8				
		feature.node.kub	ernetes.io/cpu-	cpuid.HYPERVISOR-	true Sho	ow more												
	C (Active)	test4	All	v1.24.13+rke2r	1	10.10.0.22	20 🕞 / Same as	is External	Linux	n/a		n/a			299	55 days		1
		Labels: feature.	ode.kubernete ernetes.io/cm-	s.io/cpu-cpuid.ADX-t	true feat	ture.node.kut še kubernete	bernetes.io/cpu-cp	wid AESNI-tr	ue feat	ture.node.kub	ernetes.io/c	pu-cpuid	LAVX-true	8				
Cluster Tools		feature.node.kub	emetes.ia/cpu-	cpuid.HYPERVISOR+	true Sho	ow more												
test2:~	# kube	ctl get	nod	es														
NAME	STAT	TUS R	OLES								AG	E	V	ERS:	ION			
basepod-dgx0	3 Read	dy w	orke	r							22	d	v	1.24	4.13	⊦rke	e2r	1
basepod-dgx0	4 Read	dy w	orke	r							55	d	v	1.24	4.13	⊦rke	e2r	1
test2	Read	dy c	ontr	ol-pla	ne,	etcd	,mast	er,w	ork	ker	55	d	v	1.24	4.13	⊦rke	e2r	1
test3	Read	dy c	ontr	ol-pla	ne,e	etcd	,mast	er,w	ork	ker	55	d	V	1.24	4.13	⊦rke	e2r	1
test4	Read	dy c	ontr	ol-pla	ne,	etcd	,mast	er,w	ork	ker	55	d	V	1.24	4.13	⊦rke	≘2r	1

Figure 9. Verification of successful role assignment to DGX nodes via Rancher Management console and command line (kubectl).

At the end of this section, we now have an RKE2 cluster deployed with NVIDIA DGX available as worker nodes.

Test/Task	Complete Y/N
Build Rancher Cluster (pre-requisite)	Y
Install and deploy RKE2 cluster as worker nodes	Υ
Deploy NVIDIA GPU operator and enable Multi-Instance GPU	Ν
Install and deploy Longhorn (persistent storage)	Ν
Deploy accelerated workload on NVIDIA DGX worker nodes	Ν

Deploy NVIDIA GPU operator and enable Multi-Instance GPU (MIG).

"The NVIDIA GPU Operator manages NVIDIA GPU resources in a Kubernetes cluster and automates tasks related to bootstrapping GPU nodes. Since the GPU is a special resource in the cluster, it requires a few components to be installed before application workloads can be deployed onto the GPU. These components include the NVIDIA drivers (to enable CUDA), Kubernetes device plugin, container runtime and others such as automatic node labelling, monitoring and more". (Operator available at: <u>https://github.com/NVIDIA/gpu-operator</u>)

In this section, we will deploy the NVIDIA GPU Operator and demonstrate MIG partitioning on the NVIDIA DGX node.

First, we deploy the NVIDIA GPU Operator to the cluster. From the Rancher Management system, add the NVIDIA gpu-operator helm chart.

🚴 rke2-test						default \mathbf{X}			~	Ť	٤			Q	:	н
r oads	* *	Charts														
irts	^	PSP Removal	: Before upgrad	ing a cluster to Kub	ernetes 1.25-	⊦, please ensure y	/ou revi	ew your Heln	n applicat	ions for Po	od Security F	olicies	and up	date the	em	
alled Appr	1	accordingly														
aneu Apps	2	All charts hav	ve at least one ve	ersion that is install:	able on cluste	ers with Linux and	d Winde	ows nodes un	less other	wise indic	ated.					×
ent Operations	0															
en operations		nvidia					~	All Catego	ries							~
e Discovery	~							, ar earcego								
;e	~						0									
	~						_									
A Sources			gpu-operator NVIDIA GPU O creates/configu GPUs atop Kub	perator ires/manages ernetes Linux only	I											
= 🚴 rke2-test								gpu-operator X		v	± ۲		0	a c	:	æ
Cluster	~	Pods 🕸													Cr	reate
Workloads	î														h	
III CronJobs	7	± Download ₩	AML # De	lete							=					
III Deployments	2	-	and the second						2000							
📾 Jobs	0	State	Name 🖓		Imag	eĢ			Ready 0	Restarts	IP 0	Not	e o	Age C		
StatefulSets	0	Namespace: gpu-c	operator													
Pods	23	Comina)				in the old in these distances	diama	0.0.0.0.0.0			10 42 221 2	, bas	epod-	7.4		
Apps Service Discovery	Ŭ	C County	Sho-reardi e-disc	user k-surchs	inet.	ovniviu agpuneature	r-uiscover	y.vu.7.0-uulo	1/1		10.42.221.5	" dgx	04	7 days		
Storage		C Running	gpu-feature-disc	overy-5gwhl	nvcr.	io/nvidia/gpu-feature	e-discover	ry:v0.7.0-ubi8	1/1	3 (7d18h ago)	10.42.147.2	45 dgx	Jpod- 03	22 day	s	1
Policy	~	(Running)	gpu-operator-95	b545dóf-bsq4k	nvcr.	io/nvidia/gpu-operati	or:v22.9.1	1	1/1	7 (8h ago)	10.42.215.1	ló test	2	7 days		
Monitoring	~	(Running)	gpu-operator-no 84c7c7c6cf-pffj4	de-feature-discovery-mi l	ester- k8s.g	cr.io/nfd/node-featu	re-discov	ery:v0.10.1	1/1	6 (8h ago)	10.42.153.1	173 test	4	7 days		1
Longhorn More Resources		Running	gpu-operator-no worker-28htr	de-feature-discovery-	k8s.g	cr.io/nfd/node-featu	re-discov	ery:v0.10.1	1/1	75 (8h ago)	10.42.221.4	9 basi	apod- 04	22 day	s	E
		(Running)	gpu-operator-no worker-75sqq	de-feature-discovery-	k8s.g	cr.io/nfd/node-featu	re-discov	ery:v0.10.1	1/1	80 (8h ago)	10.42.147.2	21 base	epod- 03	22 day	5	Ĩ.
		Running	gpu-operator-no d5rgz	de-feature-discovery-wo	orker- k8s.g	cr.io/nfd/node-featu	re-discov	ery:v0.10.1	1/1	29 (8h ago)	10.42.153.1	130 test	4	22 day	s	1
		(Running)	gpu-operator-no kvófb	de-feature-discovery-wo	orker- k8s.g	cr.io/nfd/node-featu	re-discov	ery:v0.10.1	1/1	32 (8h ago)	10.42.215.1	test	2	22 day	5	ł
ĸ		Running	gpu-operator-no	de-feature-discovery-wo	orker- k8s.g	cr.io/nfd/node-featu	re-discov	ery:v0.10.1	1/1	33 (8h ago)	10.42.231	12 test	3	22 day	s	
		(Running)	nvidia-container-	-toolkit-daemonset-öxrp	2 nvcr.	io/nvidia/k8s/contain tu20.04	ner-toolki	tv1.11.0-	1/1	2 (7d18h ago)	10.42.221.5	7 bas	epod- 04	22 day	5	1
Cluster Tools		(Running)	nvidia-container-	-toolkit-daemonset-pl5x	b nvcr.	io/nvidia/k8s/contain	er-toolki	t:v1.11.0-	1/1	3 (7d18h	10.42.147.2	15 bas	epod-	22 day	s	1

Figure 10. NVIDIA gpu-operator deployment using helm charts (via Rancher).

Some NVIDIA GPUs such as H100, A100, and A30 support Multi-Instance GPU (MIG) technology. "MIG can partition the GPU into as many as seven instances, each fully isolated with its own bandwidth memory, cache, and compute cores."

To enable MIG, change the node(s) *nvidia.com/mig.config* to the correct value that you want to achieve. Per NVIDIA GPU Operator with MIG documentation "The mig-manager uses a ConfigMap called mig-parted-config in the GPU Operator namespace in the daemonset to include supported MIG profiles".

The default value for the migManager variable is 'all-disabled'. To apply a profile:

- Check the correct profile for the correct GPU type on your DGX nodes. NVIDIA A100-40GB and A100-80GB have different profiles. Our systems had the A100-80GB. As such, we opted for the 'all-1q.10qb profile'.
- From the Rancher console select Storage -> ConfigMaps -> default-mig-parted-config.



Cluster Workloads	* *	ConfigMaps 🛱				Create
Apps Service Discovery	* *	🛃 Download YAML 🗴 Delete				
© PersistentVolumes	^	□ Name≎	Namespace 🗘	Data	Age 🗘	
StorageClasses	0	default-gpu-clients	default	clients.yaml	19 hours	÷
 ConfigMaps PersistentVolumeClaims 	8	O default-mig-parted-config	default	config.yaml	19 hours	:
Secrets	2	gpu-operator-node-feature-discovery-topology-updater-conf	default	nfd-topology-updater.con	f 19 hours	÷
Policy More Resources	č	gpu-operator-node-feature-discovery-worker-conf	default	nfd-worker.conf	19 hours	÷
More Resources		🔘 kube-root-ca.crt	default	ca.crt	21 days	
		pvidia-container-toolkit-entrypoint	default	entrypoint sh	19 hours	

Figure 11. Rancher ConfigMaps view with default-mig-parted-config option shown.

- To apply the 'all-1g.10gb' profile via the Rancher interface, go to Cluster -> Nodes -> Select the nodes where MIG is to be enabled -> Config -> Labels & Annotation.
- Search for the *nvidia.com/mig.config* label and verify your current profile. To make changes, click on the 3 dots (...) and select '*Edit Config*' as shown in Figure 15 below. Click on the labels and change *mig.config* to your desired profile and then click 'Save'.

	default X	· ± Σ 🗎 Φ Ο	:
nvidia.com/gpu.pro	duct	NVIDIA-A100-SXM4-80GB	Remov
nvidia.com/gpu.rep	licas	1	Remov
nvidia.com/mig.cap	able	true	Remov
nvidia.com/ <mark>mig.com</mark>	ng	all-1g.10gb	Remov
nvidia.com/mig.com	fig.state	success	Remov
nvidia.com/mig.stra	itegy	mixed	Remov
plan.upgrade.cattle	.io/system-agent-upgrader	7dbfe3bc7aa1f9e217597840afda8c191bdef89403968e d04de99eec	Remov
rke.cattle.io/machi	ne	715fae06-e3eb-4823-94c4-42cb92d41fd5	Remov
		Cancel Edit as YAML	Sav

Figure 12. Having selected an NVIDIA DGX node, we apply the desired profile to the nvidia.com/mig.config variable.

- Once properly saved, the 'nvidia.com/mig.config.state' will show 'success'.
- Optional: mig-config can be verified by connecting directly to the DGX node(s) and running the 'sudo nvidia-smi mig -lgi' and 'sudo nvidia-smi -L' and check the output for the appropriate labels and partitions.

At the end of this section, we have deployed the NVIDIA GPU Operator and enabled MIG.

Test/Task	Complete Y/N
Build Rancher Cluster (pre-requisite)	Y
Install and deploy RKE2 cluster as worker nodes	Y
Deploy NVIDIA GPU operator and enable Multi-Instance GPU	Y
Install and deploy Longhorn (persistent storage)	Ν
Deploy accelerated workload on NVIDIA DGX worker nodes	Ν



Install and deploy Longhorn (persistent storage).

Longhorn is an official CNCF project that delivers a powerful cloud-native distributed storage platform for Kubernetes that can run anywhere. Combined with Rancher, Longhorn makes the deployment of highly available, persistent block storage in your Kubernetes environment easy, fast, and reliable.

Installation of Longhorn is straightforward. Please refer to <u>Longhorn Installation</u> <u>Documentation</u> for deployment via Rancher Apps, Kubectl, or Helm as desired.

• Note: Before installing Longhorn, make sure that each node on the cluster has open-iscsi installed. The NVIDIA DGX nodes that we utilized had their iscsi service masked. If needed, run *sudo systemctl unmask iscsid.service* to unmask.

LONGHO)RN Le Dashbo	ard 🔜 Node	E Volume C Reci	urring Job 🕻 Back	up 🕸 Setting			
ashboard								
	1			0 00 T			-	
	Volume			Z.3Z II Storage Schedulable			J Nodes	
								4
	*							
 Healthy 		1	Schedulabl	e	2.32 Ti	Schedulable		
Degraded		0	e Reserved		1.07 Ti	Unschedulable		
In Progress	i i	0	Used		169 Gi	🔵 Down		
Fault		0	Disabled		0 Bi	Disabled		
Detached		0	Total		3.55 Ti	Total		
Total		1						
		shhoard 🗆 No	de 🗉 Volume	C Recurring Joh	C Backup	爺 Settina∨		
Lowen				G Recoming 500	G Duckup	e octang •		
Node								
Expand All	Delete Edit Node	e				Name	v	
-	Status 韋	Readiness	Name ≑	Replicas 韋	Allocated	¢ Used ¢	Size 🌲	
	Schedulable	Ready	basepod-dgx03	1	50 / 2462 4	Gi 44 35 / 1758 85 Gi	1.2 Ti +528 Gi Reserved	
+						101 102 117 JUL 101 101		
+								
+	Schedulable	Ready	basepod-dgx04 10.42.221.61	0	0 / 2462.4	Gi 55.24 / 1758.85 Gi	1.2 Ti +528 Gi Reserved	
+ -	Schedulable	Ready	basepod-dgx04 10.42.221.61 test2	0	0 / 2462.4	Gi 55.24 / 1758.85 Gi	1.2 Ti +528 Gi Reserved 28 Gi	
+ +	Schedulable	Ready Ready	basepod-dgx04 10.42.221.61 test2 10.42.215.54	0	0 / 2462.4	Gi 55.24 / 1758.85 Gi Gi 24.37 / 39.99 Gi	1.2 Ti +528 Gi Reserved 28 Gi +12 Gi Reserved	
* · ·	Schedulable Schedulable Schedulable	Ready Ready Ready	basepod-dgx04 10.42.221.61 test2 10.42.215.54 test3 10.42.231.224	0 1 1	0 / 2462.4 50 / 55.99 50 / 55.99	Gi 55.24 / 1758.85 Gi Gi 24.37 / 39.99 Gi Gi 23.88 / 39.99 Gi	1.2 Ti +528 Gi Reserved 28 Gi +12 Gi Reserved 28 Gi +12 Gi Reserved	
* U * D * D	Schedulable Schedulable Schedulable	Ready Ready Ready	basepod-dgx04 10.42.221.61 test2 10.42.215.54 test3 10.42.231.224 test4	0 1 1	0 / 2462.4 50 / 55.99 50 / 55.99	Gi 55.24 / 175a.e5 Gi Gi 24.37 / 39.99 Gi Gi 23.aa / 39.99 Gi	1.2 Ti +528 Gi Reserved 28 Gi +12 Gi Reserved 28 Gi +12 Gi Reserved 28 Gi	

Figure 13. Longhorn view of storage across all nodes that make up the cluster (including NVIDIA DGX).

At the end of this section, we installed and deployed Longhorn as a persistent storage solution.

Test/Task	Complete Y/N
Build Rancher Cluster (pre-requisite)	Y
Install and deploy RKE2 cluster as worker nodes	Y
Deploy NVIDIA GPU operator and enable Multi-Instance GPU	Y
Install and deploy Longhorn (persistent storage)	Y
Deploy accelerated workload on NVIDIA DGX worker nodes	Ν

Deploy accelerated workload on NVIDIA DGX worker nodes.

The testing/certification effort should demonstrate the execution of a workload leveraging NVIDIA GPUs on the NVIDIA DGX nodes. SUSE opted to showcase <u>Opni.</u> Opni is open-source software designed for multi-cluster and multi-tenant observability. It's built on Kubernetes and simplifies the process of creating and managing backends, agents, and data related to logging, monitoring, and tracing. With its built-in AlOps, Opni allows users to swiftly detect anomalous activities in their data.

Complete installation instructions are available at: https://opni.io/installation/opni.

Please note: Opni requires the *cert-manager* certificate controller. There are two ways to install: using *kubectl apply* with static manifests or via *helm*. Given the availability of the Rancher Management platform, *helm* was used.

Installation steps.

Agent installation:

• From the Rancher User Interface, select the Opni helm chart, followed by Install.

Cluster	~	Charte	
Workloads	~		
Apps	^	PSP Removal: Refore ungrading a cluster to Kubernetes 1.25+, please ensure you review your Helm applications for Pod Security Policies and undate them	
€ Charts		accordingly	
Installed Apps	16		
Repositories	4	All charts have at least one version that is installable on clusters with Linux and Windows nodes unless otherwise indicated.	×
Recent Operations	0		
Service Discovery	~	opni 🗸 V All Categories	×
Storage	~	Filter	
Policy	~		
Monitoring	~		_
Longhorn		Opni Opni Agent opni-prometheus-crd	
More Resources	~	Multi Cluster Observability with AlOps Opni Agent Installs the Prometheus CRDs for Opni.	
		Linux only Linux only Linux only	/

Figure 14. Opni installation via Rancher User Interface (UI).

- Follow the prompts in the user interface. Specifically:
 - New namespace selection.
 - Gateway hostname. This is the hostname that agents will used to connect to the Opni gateway.
 - Service type. LoadBalancer was selected as service type.
- Once the prompts are complete, click Next and Install.

Enable Opni backend:

An observability backend is where observability data is sent for storage and querying. These backends are configured in the Opni Management UI. There are three backends currently available via Opni (see https://opni.io/installation/opni/backends) for complete details.

For the purposes of our test, we will enable Opni Logging. Steps followed:

- Enable port forwarding from the local machine to access the dashboard. For example: kubectl -n opni port-forward svc/opni-admin-dashboard web:web
- Note: To access a Kubernetes cluster from the local machine, you will need to install *kubectl* and copy the cluster's *kubeconfig* file to your local ~/.*kube/config* directory.
- Note: For best user experience, consider using Google Chrome.

$\leftarrow \ \rightarrow \ {\bf G}$	00	http://localhost:1	2080/agents		*	${igsidential}$	$ \pm $	≡
OPNI								
Agents								
Logging		Agents					Add	
Monitoring	~							
Alerting	~	🛍 Delete						
AlOps	~							
		Status 🗘	Name 🗘	Cluster ID \Diamond		Local	\diamond	
		Ready	local	2cb1fc91-94e6-4732-9ef3-6066dc5a7fde		\checkmark	:	
k								

Figure 15. Accessing Opni via the dashboard.

Enable logging from the Opni dashboard and select 3 replicas from the Controplane Pods • option.

$\leftarrow \ \rightarrow \ G$	00	http://localhost:12080/logo	jing-config#controlplane-	pods	ជ	⊚ ⊻ ≡
OPNI						
Agents						
Logging		Logging				
Monitoring	~					
Alerting	~	Data Retention *				
AlOps	~	7d				
		Primary Pods Ingest Pods Controlplane Pods Dashboard	Controlplane P Replicas 3 Storage Persistent Show Advanced	Yods D Storage Class longhorn	V Disable p	od separation
	A -				Car	ncel Save

Figure 16. Opni logging and ControlPlane Pods configuration.

Opni usage and assignment to MIG GPU partition:

As previously mentioned, Opni is an AIOps observability platform. Log aggregation capabilities are performed using <u>OpenSearch</u>.

- Once Opni logging is configured, we proceed to access the OpenSearch dashboards. •
- Enable port forwarding of the "opni-opensearch-svc-dashboard" by executing: kubectl -n • opni port-forward svc/opni-opensearch-svc-dashboards 5601:5601

Access the OpenSearch dashboard:



Figure 17. OpenSearch dashboard access.

- Opni will create an admin user that must be used to log in to the dashboards. The username and password admin credentials can be obtained from the opni-adminpassword secret.
 - For username: kubectl get secret -n opni opni-admin-password -o jsonpath='{.data.username}' | base64 -d
 - For password: kubectl get secret -n opni opni-admin-password -o 0 jsonpath='{.data.password}' | base64 -d
- Opni AlOps currently features *log anomaly detection*. It provides log insights by distinguishing normal and anomalous logs. There are two 'flavors' available today:
 - Pre-trained models trained by SUSE Rancher specialized on the Kubernetes control plane, Rancher, and Longhorn logs which do not require a GPU.
 - o Auto generated models for user selected workloads where the user selects 1 or more workload deployments important to them and Opni will self-train a model and provide insights for logs belonging to user selected workloads. An NVIDIA GPU is required for this option.
- Once enabled the *opni-svc-gpu-controller* pod will be deployed.
- From Opni AlOps, select Deployment Watchlist. Select a watchlist from the dashboard. Depending on the size of the watchlist, it may take a few hours to collect log information.

v2.7.4	update-z22vw			dgx03		
opni-svc-gpu-controller-579969fd	rc-vs9tz 🛛					
Mon, Jul 17 2023 2:58:12 pm	2023-07-17T13:54:11.460062091-07 Taken: 3.97s ETC: 13789.03s 2023-07-17 20:58:12,214 - INFO -	7:00 2023-07-17 20:54:11,459 - Epoch Step: 1/1158 Loss: 8. - Epoch: 0 Total Progress: Totare per Sec: 252 19 1	INFO - Epoch: 0 6552 Tokens per Sec 2.91% Training Tim	Total Progress: 0.0 : 306.01 e Taken: 244.735	3% Training Time ETC: 8172.27s Epo	ch
	2023-07-17T14:01:56.875004578-07 Taken: 469.395 ETC: 7642.615	7:00 2023-07-17 21:01:56,874 - Epoch Step: 201/1158 Loss:	INFO - Epoch: 0 2.3963 Tokens per	Total Progress: 5.7 Sec: 270.73	9% Training Time	
Mon, Jul 17 2023 3:05:44 pm	2023-07-17 21:05:44,014 - INFO - Step: 301/1158 Loss: 1.7261	- Epoch: 0 Total Progress: Tokens per Sec: 269.33	8.66% Training Time	e Taken: 696.53s	ETC: 7341.47s Epoc	ch
	Z023-07-17114:09:18.984681073-07 Taken: 911.49s ETC: 6984.51s	Epoch Step: 401/1158 Loss:	1.2268 Tokens per	Sec: 278.73	54% Training Time	
	2023-07-17T14:12:42.582566196-07 Taken: 1115.09s ETC: 6616.91s	1:00 2023-07-17 21:12:42,582 - Epoch Step: 501/1158 Loss	INFO - Epoch: 0 ' : 0.9392 Tokens per	Total Progress: 14. Sec: 293.98	42% Training Time	
	2023-07-17T14:16:00.365408788-07 Taken: 1312 885 FTC: 6275 125	7:00 2023-07-17 21:16:00,365 -	INFO - Epoch: 0	Total Progress: 17.	30% Training Time	
	2023-07-17714:19:31.104825428-07	7:00 2023-07-17 21:19:31,104 -	INFO - Epoch: 0	Total Progress: 20.	18% Training Time	
	2023-07-17T14:23:00.571394890-07 Taken: 1733.08s ETC: 5782.92s	Epoch Step: 801/1158 Loss Epoch Step: 801/1158 Loss	INFO - Epoch: 0 : 0.7126 Tokens per	Total Progress: 23: Sec: 290.64	065 Training Time	

Figure 18. Log data being analyzed.



• We validate that the opni-svc-gpu-controller *is managing the workload in a GPU- powered node.*

d IP: 10.42.147	7.221 Wor	kload: opni-svc-gpu	u-controller-5799	969fd7c Node	e: basepod	l-dgx03			
bels: app.kuber	netes.io/name :	opni-svc-gpu-controlle	er app.kubernetes	s.io/part-of:opni	opni.io/ser	vice : gpu-controller	r		
od-template-hash	: 579969fd7c								
notations: Sho	w 6 annotatio	ons							
Containers	Metrics	Conditions	Recent Events	Related Re	sources				
Containers	Metrics	Conditions	Recent Events	Related Re	sources				
Containers	Metrics	Conditions	Recent Events	Related Re	sources	Init			
Containers State 🗘	Metrics Ready 💸	Conditions Name 🗘	Recent Events	Related Re	sources	Init Container [©]	Restarts	Started 💸	
Containers State 🗘	Metrics Ready \Diamond	Conditions Name 🗘	Recent Events	Related Re mage ()	sources	Init Container \Diamond	Restarts	Started 🗘	

Figure 19. Opni execution via opni-svc-gpu-controller.

• Note: To assign a workload to a specific MIG partition, a *nodeSelector* can be added to the configuration yaml file (for example: *nodeSelector.nvidia.com/mig-1g.10gb:1*)

At the end of this section, we successfully deployed an accelerated workload (Opni) and leveraged a GPU on an NVIDIA DGX node. Lastly, we showed how we could use a partition on the GPU instead of the entire GPU.

Test/Task	Complete Y/N
Build Rancher Cluster (pre-requisite)	Υ
Install and deploy RKE2 cluster as worker nodes	Υ
Deploy NVIDIA GPU operator and enable Multi-Instance GPU	Υ
Install and deploy Longhorn (persistent storage)	Υ
Deploy accelerated workload on NVIDIA DGX worker nodes	Υ

Wrap-Up

This document showed the successful integration of the NVIDIA DGX platform into a Kubernetes cluster (RKE2) managed via Rancher Management platform while also leveraging the persistent storage capabilities of Longhorn. Upon completion of the integration into the cluster, we deployed an accelerated workload via Opni and demonstrated its usage of an NVIDIA GPU within an NVIDIA DGX node.

SUSE and NVIDIA customers can rest assure that NVIDIA DGX can be integrated into a Rancher-managed Kubernetes infrastructure and its set of accelerators can be quickly used to address customer needs.