01-NOV-2023

SUSE Rancher Stack

NVIDIA DGX Testing and Deployment Guide

Table of Contents

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 website, "*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 Harvester 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 on-prem setups). The virtual machine's OS was SUSE Linux Enterprise Micro.

Worker Nodes:

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

Two NVIDIA DGX (physical nodes) added to the Kubernetes cluster for deployment of GPU-based 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: https://ranchermanager.docs.rancher.com/

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 https://docs.harvesterhic.io/ 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.

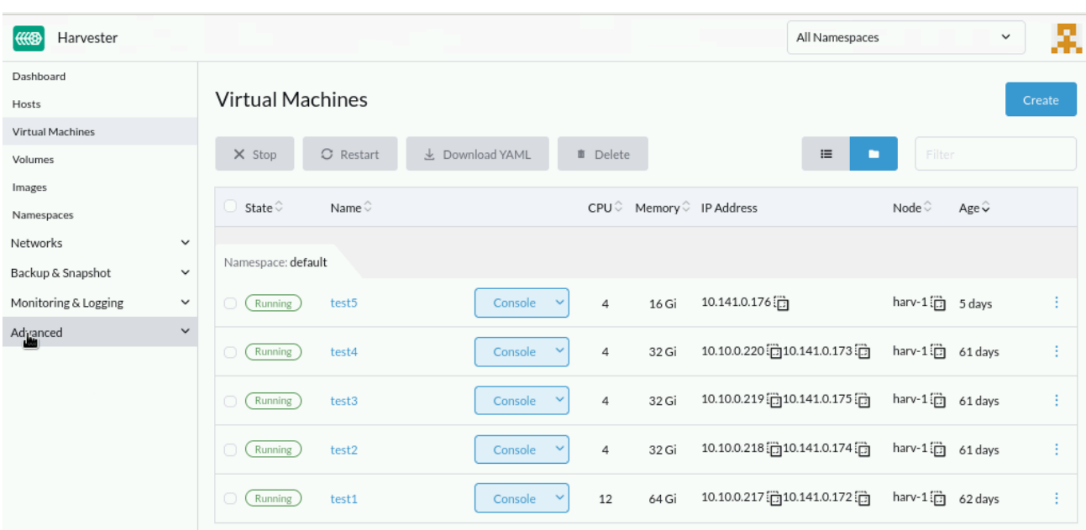

Figure 1. Virtual Machines used to deploy management nodes and management cluster components.

- Install SUSE Linux Enterprise Micro 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:
    - SUSE Linux Enterprise Micro Deployment Guide
    - 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 Helm CLI Quick Start.
- 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 https://get.k3s.io | 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 kubectl get nodes
NAME     STATUS   ROLES                       AGE   VERSION
test1    Ready    control-plane,etcd,master   15s   v1.24.14+k3s1
```

  - *# kubectl apply --validate=false -f https://github.com/cert-manager/cert-manager/releases/download/v1.11.0/cert-manager.crds.yaml*
  - *# 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 cert-manager --create-namespace --version v1.11.0*
  - *# kubectl get pods --namespace cert-manager*

```
test1:~ # kubectl get pods --namespace cert-manager
NAME                                       READY   STATUS    RESTARTS   AGE
cert-manager-85945b75d4-g5qgq              1/1     Running   0          59s
cert-manager-cainjector-7f694c4c58-lls88   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 https://releases.rancher.com/server-charts/stable*
  - *# 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:

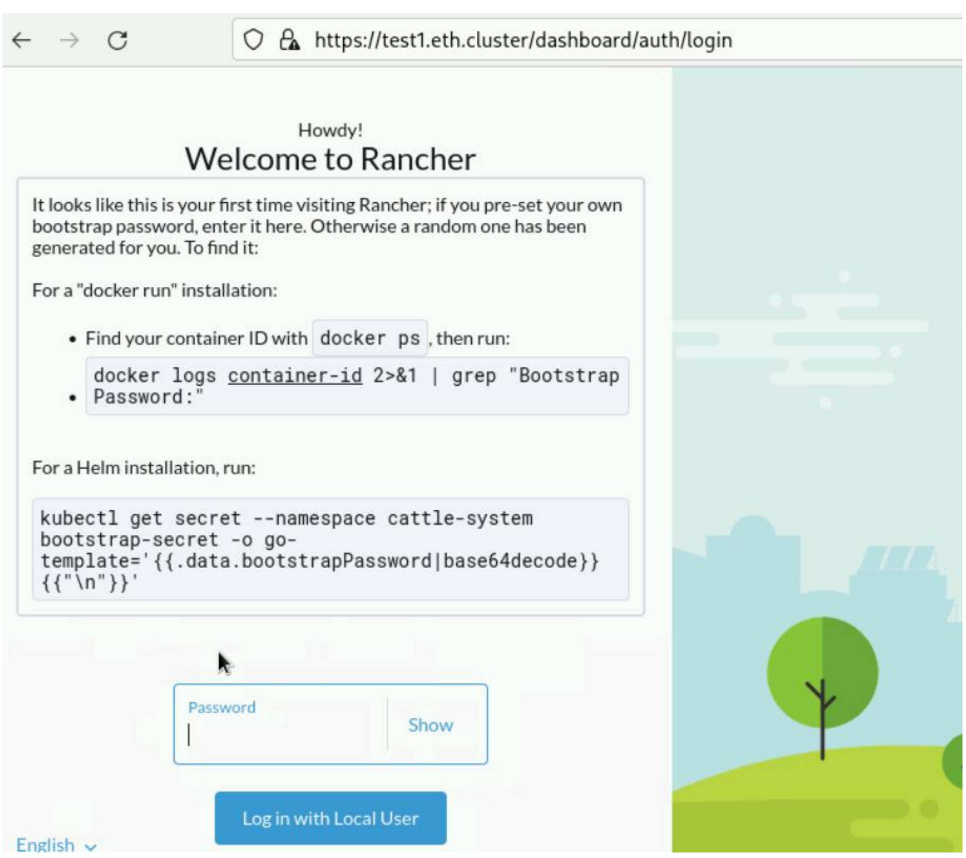

Figure 2. Rancher login page and URL after installation.

- 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.

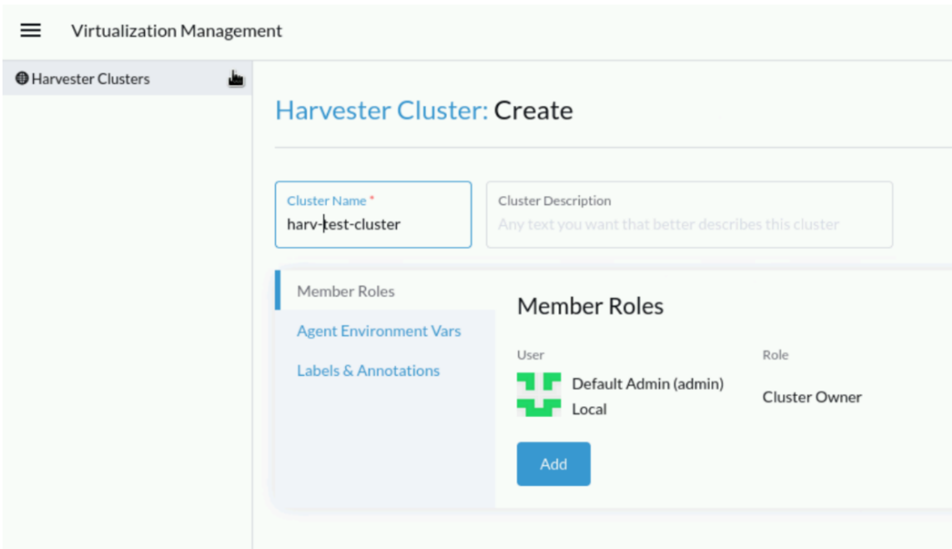

Figure 3. Harvester and Rancher integration.

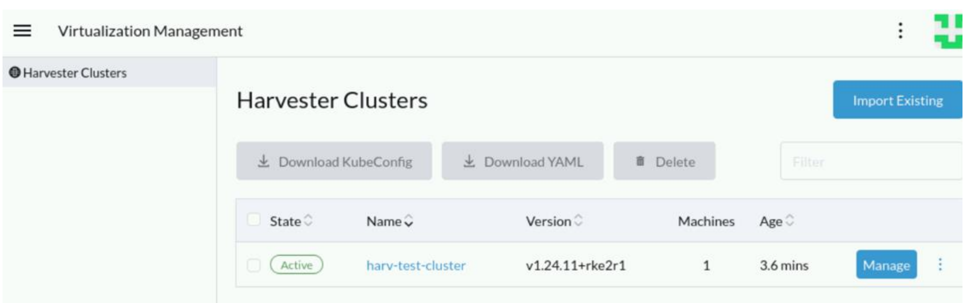

Figure 4. Harvester and Rancher integration.

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

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

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"*.

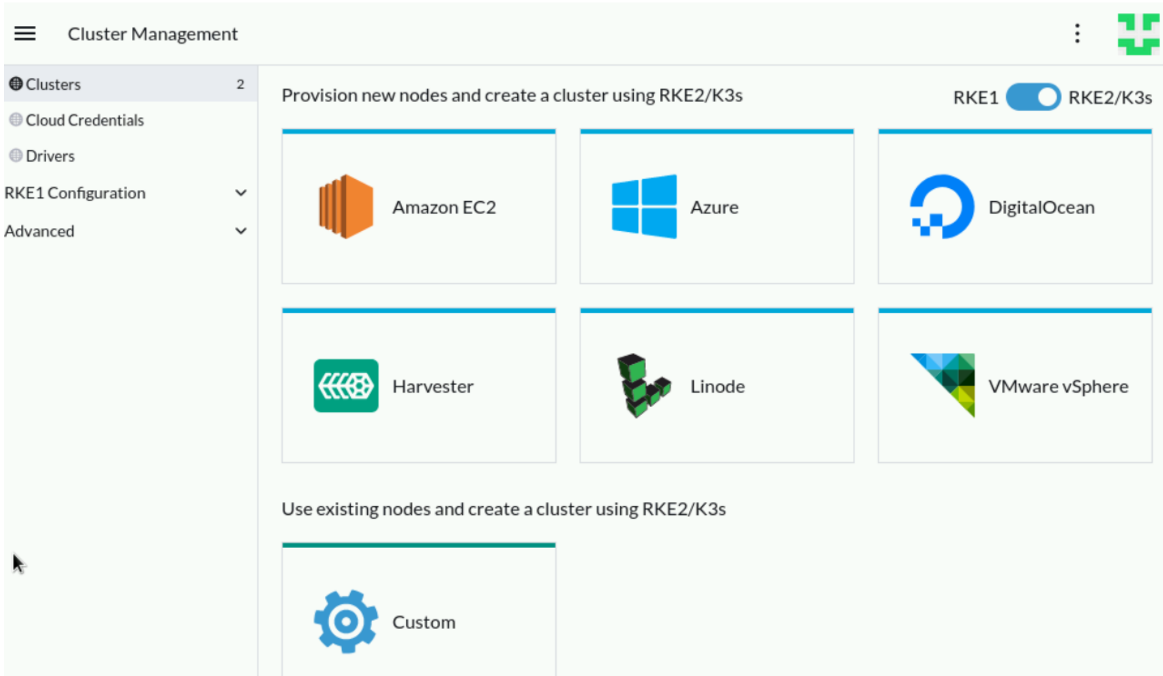

Figure 5. Rancher Cluster Management View.

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

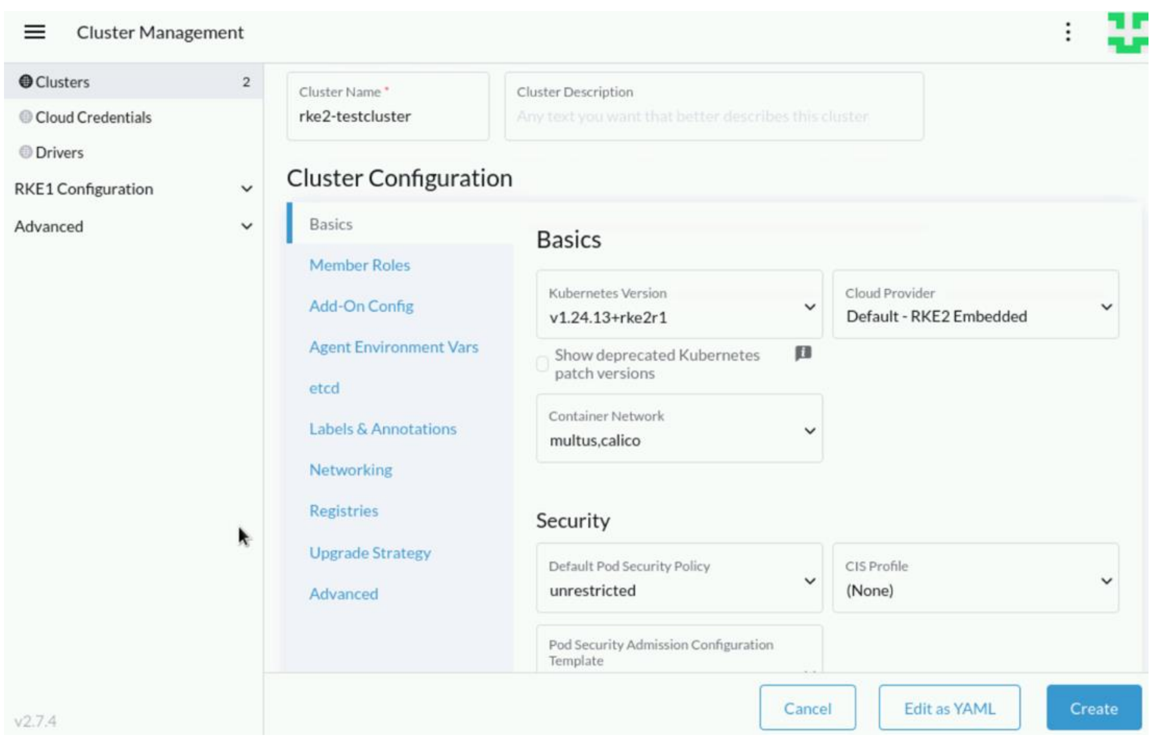

Figure 6. Cluster configuration sample – RKE2.

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

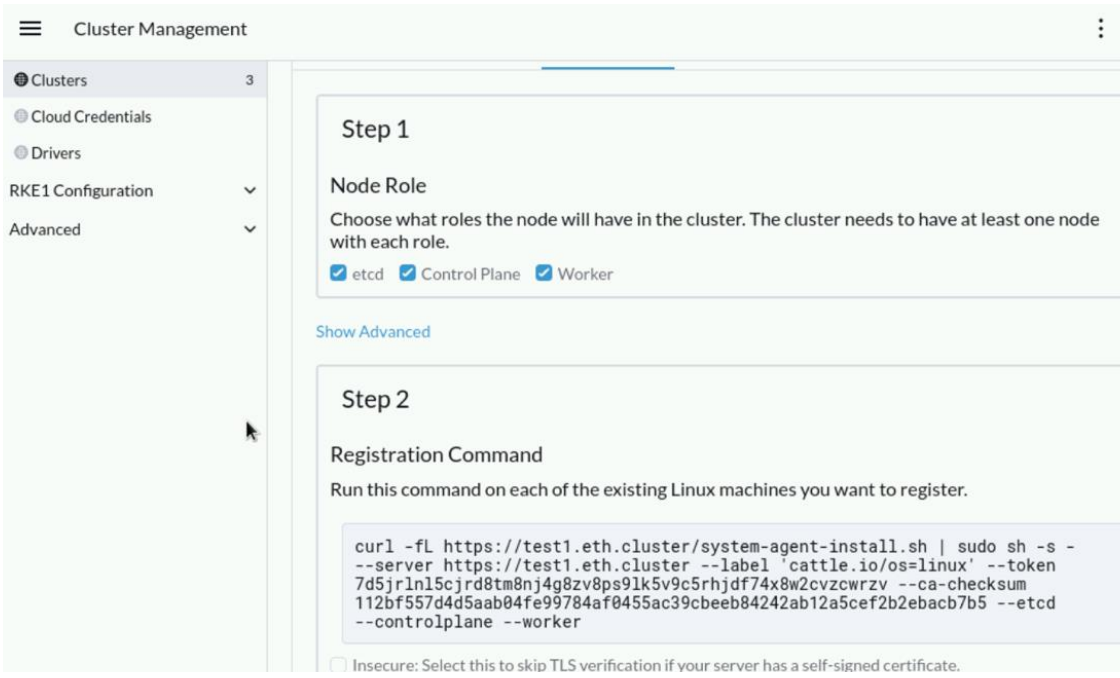

Figure 7. Registration command to be copied.

- 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.

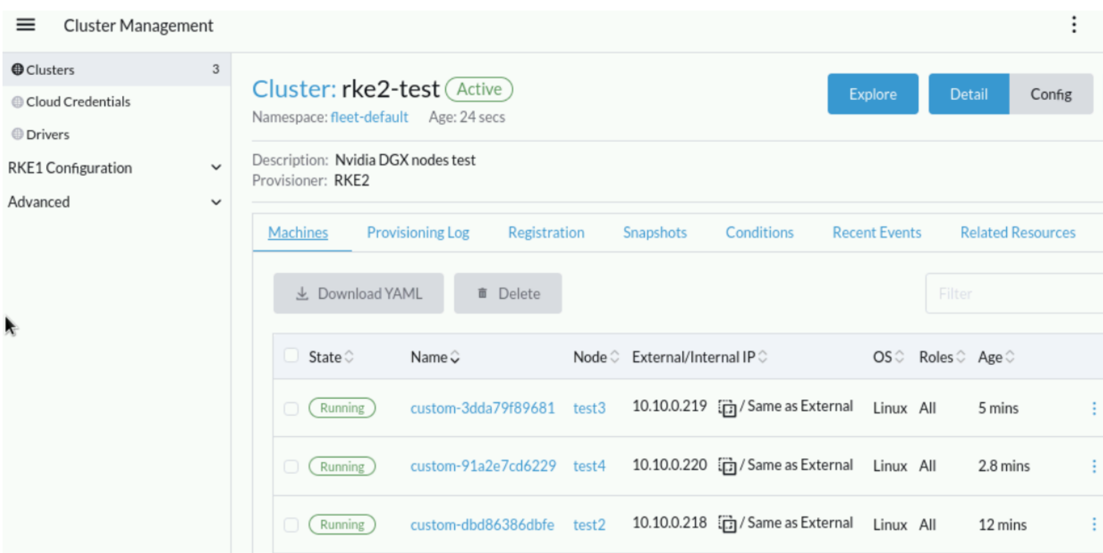

Figure 8. Three-node RKE2 cluster view.

- 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.

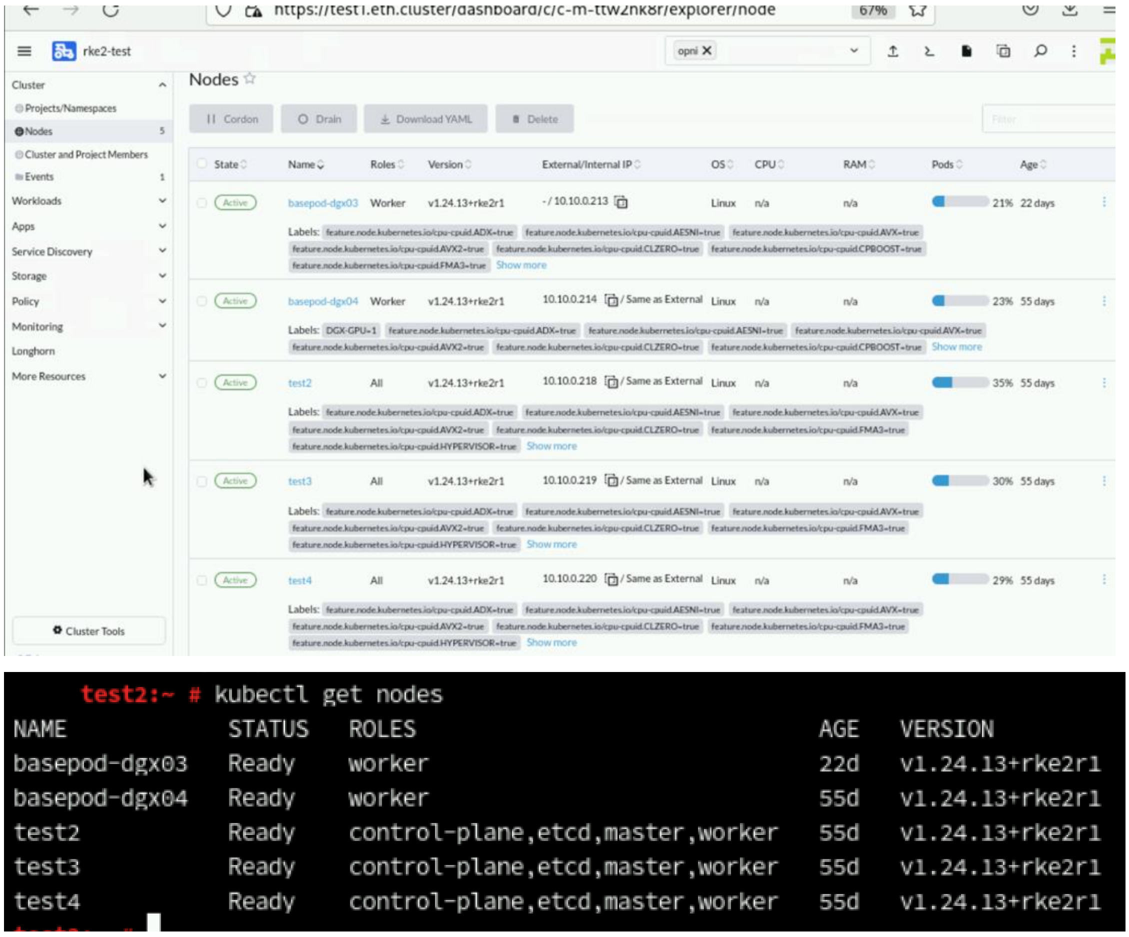

Figure 9. Verification of successful role assignment to DGX nodes via Rancher Management console and command line (kubectl).

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	Y
Deploy NVIDIA GPU operator and enable Multi-Instance GPU	N
Install and deploy Longhorn (persistent storage)	N
Deploy accelerated workload on NVIDIA DGX worker nodes	N

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: https://github.com/NVIDIA/gpu-operator)

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.

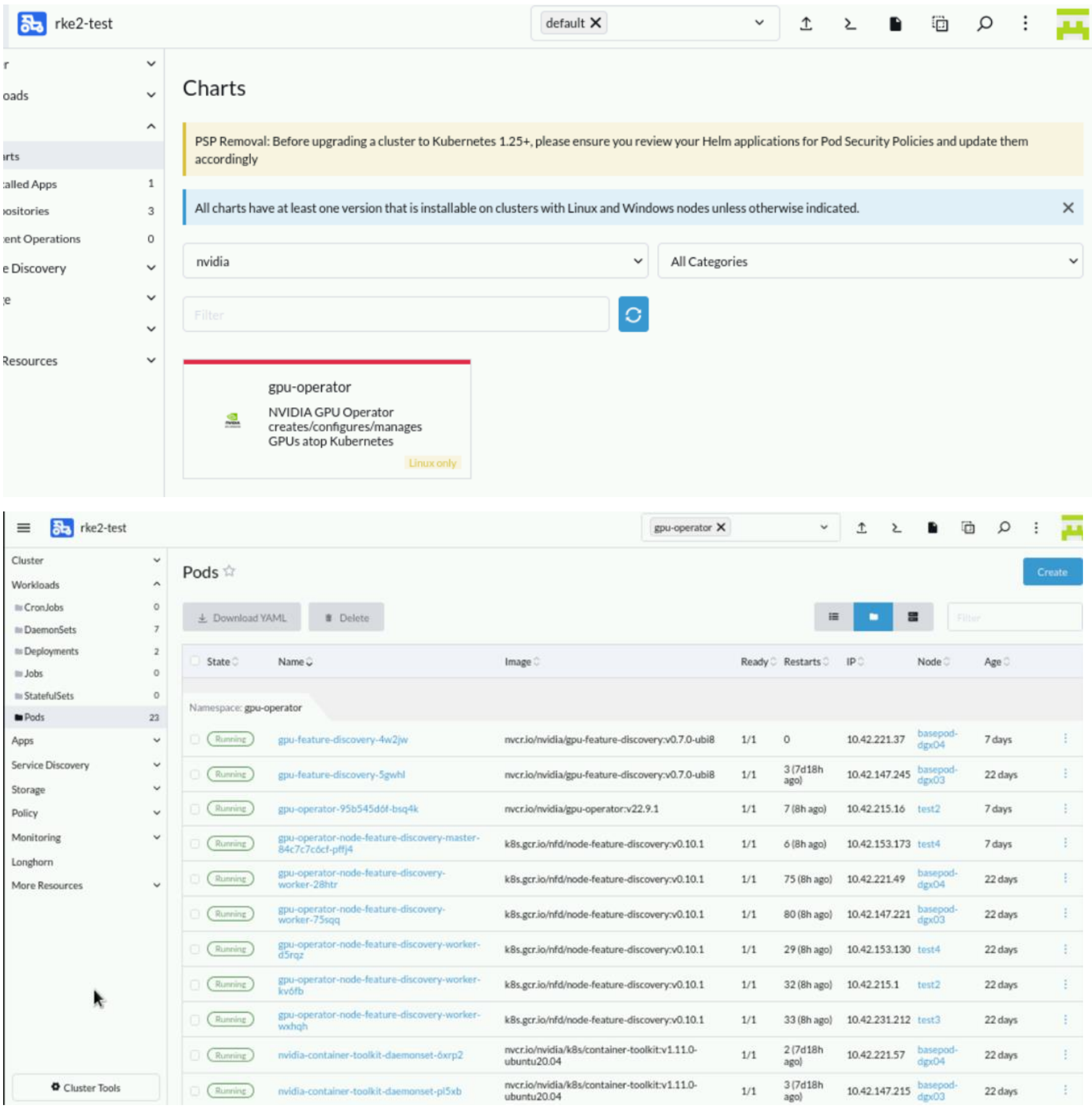

Figure 10. NVIDIA gpu-operator deployment using helm charts (via Rancher).

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-1g.10gb profile'.*
- From the Rancher console select Storage -> ConfigMaps -> default-mig-parted-config.

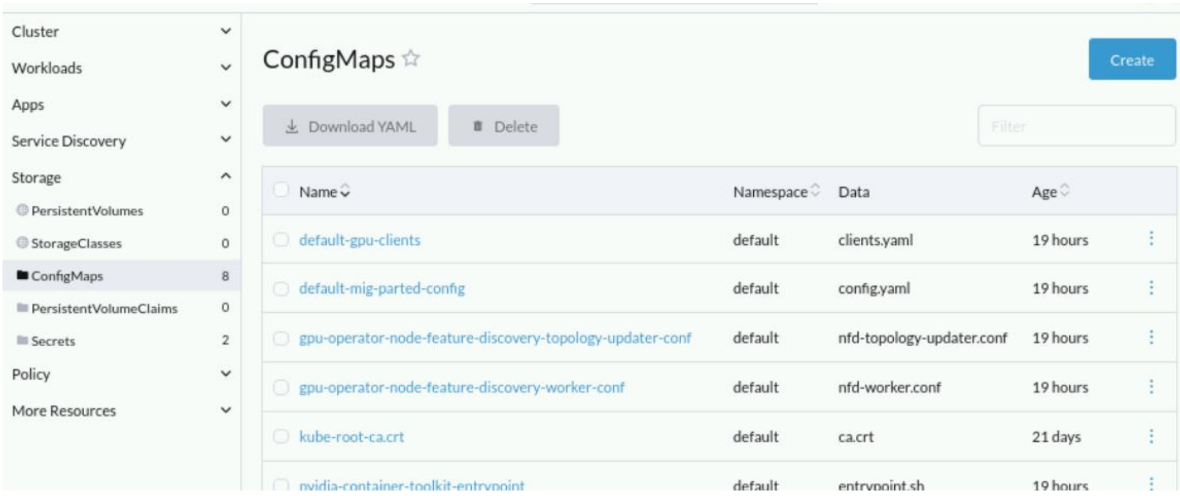

Figure 11. Rancher ConfigMaps view with default-mig-parted-config option shown.

- 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'.

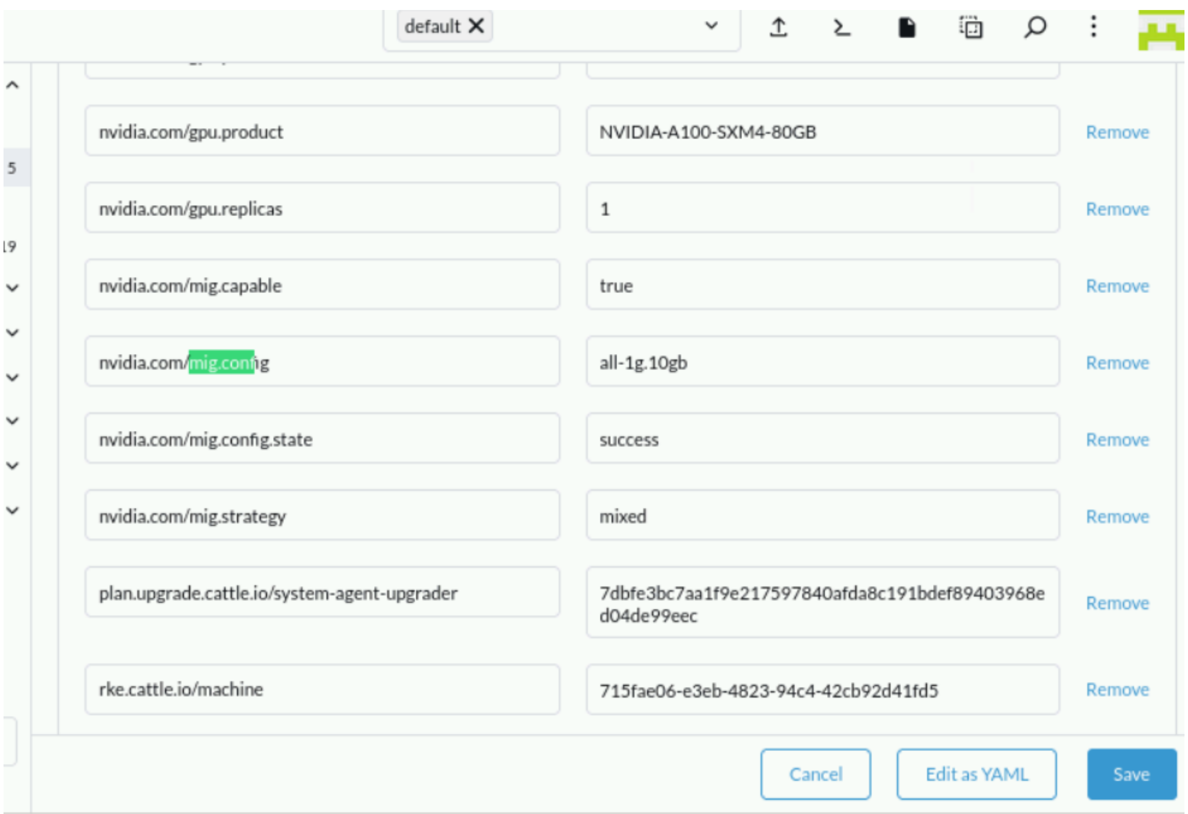

Figure 12. Having selected an NVIDIA DGX node, we apply the desired profile to the nvidia.com/mig.config variable.

- 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)	N
Deploy accelerated workload on NVIDIA DGX worker nodes	N

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 Longhorn Installation Documentation 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.

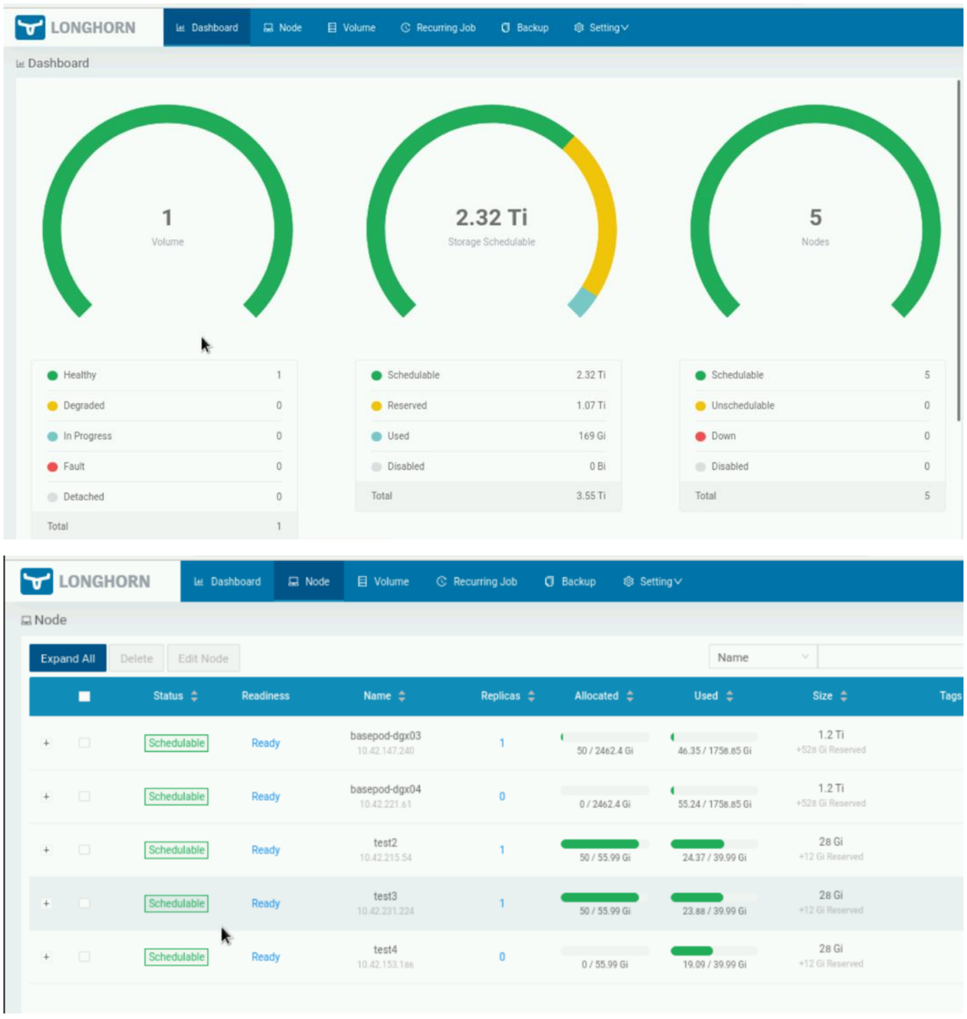

Figure 13. Longhorn view of storage across all nodes that make up the cluster (including NVIDIA DGX).

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	N

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 Opni. 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 AIOps, 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.*

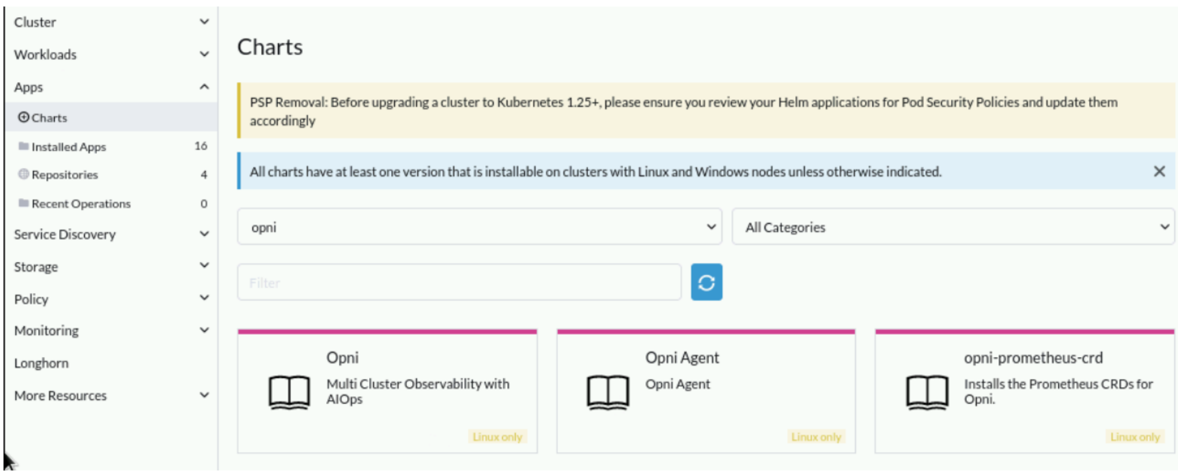

Figure 14. Opni installation via Rancher User Interface (UI).

- 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.

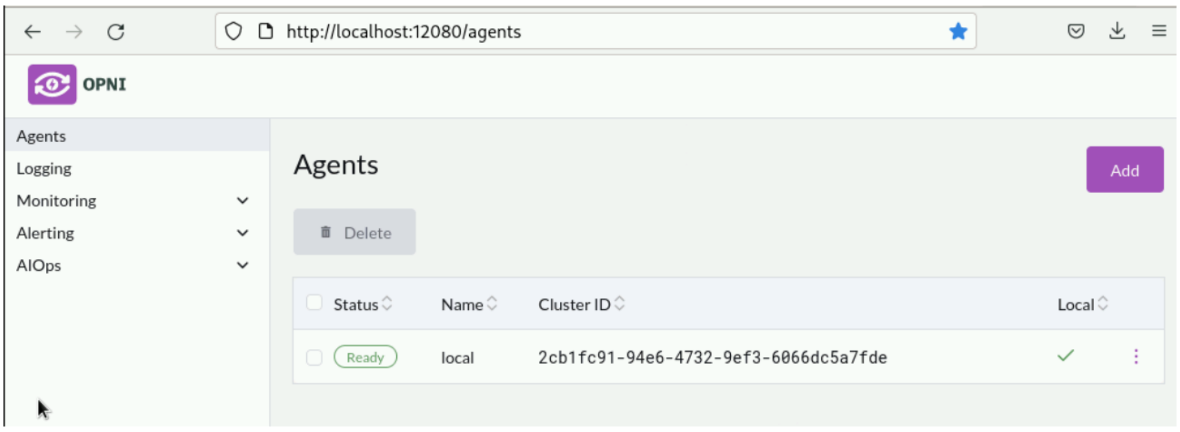

Figure 15. Accessing Opni via the dashboard.

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

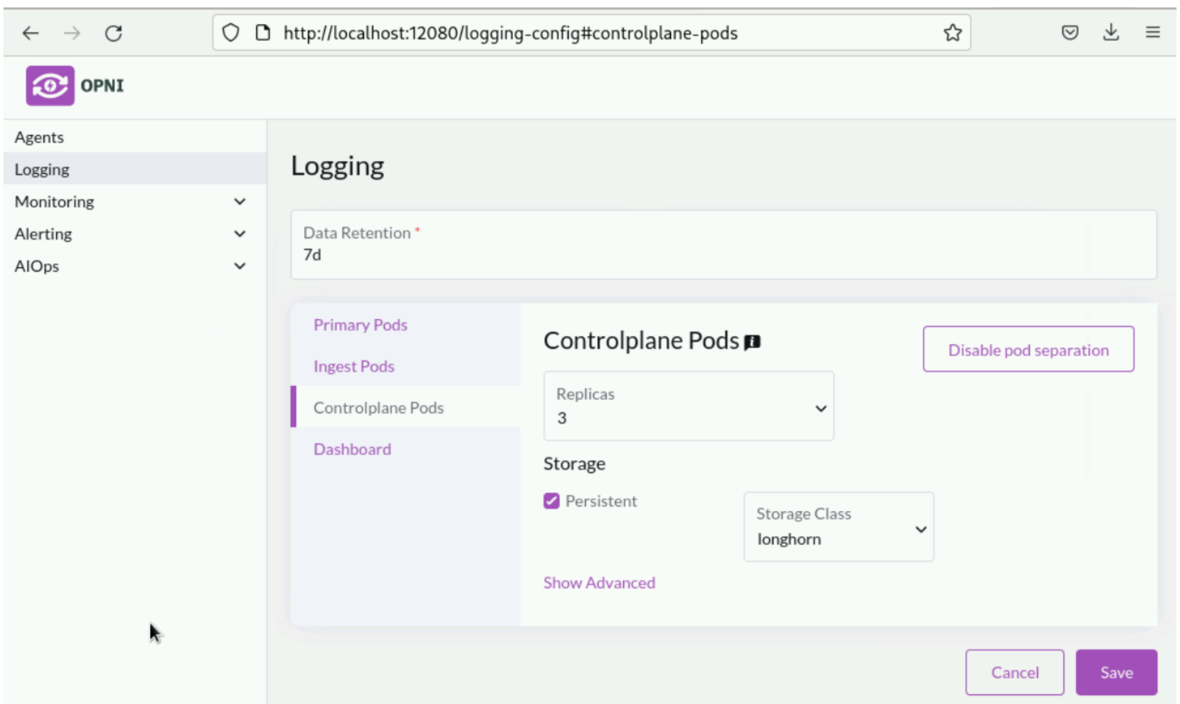

Figure 16. Opni logging and ControlPlane Pods configuration.

Opni usage and assignment to MIG GPU partition:

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

- 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-admin-password* 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 jsonpath='{.data.password}' | base64 –d*
- Opni AIOps 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.
    - 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 AIOps, 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.

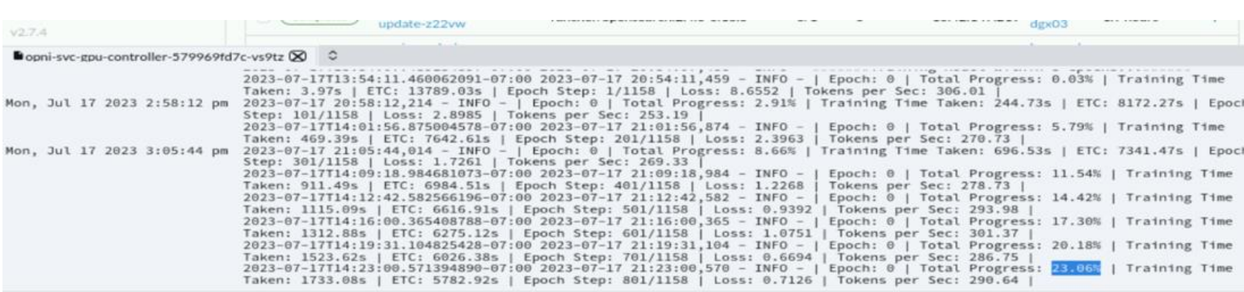

Figure 18. Log data being analyzed.

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

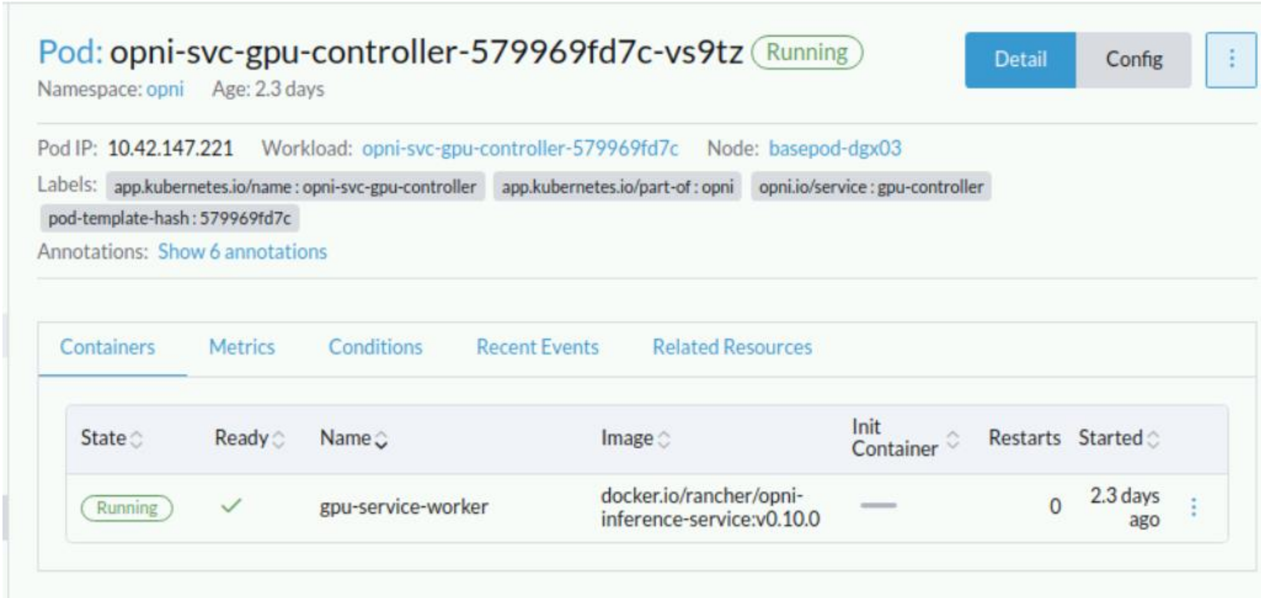

Figure 19. Opni execution via opni-svc-gpu-controller.

- 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)	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

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.