31-1-2022

Installation manual

Kubernetes with GPU integration

Maenhoudt Tom
STUDENT AT HOWEST

Content

Introduction

This document shows you how to create a RKE cluster with Rancher and how to deploy Nvidia GPU-operator. Nvidia GPU-operator makes it possible to use GPUs in user applications, for example training of neural networks. GPUs are also used for machine learning and deep learning.

I assume you have a basic understanding of Kubernetes and how containerization works. I will be explaining some concepts of Kubernetes, but I won't be able to cover it all. If you want to learn more about containers and Kubernetes, I highly recommend watching following Pluralsight courses:
Pluralsight | Docker and Kubernetes: The big picture
Pluralsight | Getting started with Kubernetes

In this demo I use a custom DNS server that is running on a Raspberry Pi. For every machine I created a DNS record. This makes things a lot easier because you don't need to remember IP addresses. If you don't have a custom DNS server or are unable to create DNS records use the IP address of the machine instead.

For operating system on the computers, I use Ubuntu Server LTS 20.04. It is Debian based and I am familiar with Debian based distributions. You can use a different operating system as long as docker can run on it.

This whole project is built and tested with Nvidia GPUs. AMD GPUs are another story, and the Nvidia GPU-operator will not work with AMD GPUs.

One last thing make sure that your master node has a drive that is big enough and has great performance. If your drive is not good enough, some pods will be evicted from the master node because of disk pressure. It is crucial for the stability of your cluster that this does not happen. Especially in an environment where nodes will be coming and going.

What do you need

- Full access to the computers that you want to use
- Computers with GPUs to function as worker nodes
- Internet connection
- WinSCP or FileZilla to transfer files over SSH (not necessary but handy to have)
- Empty USB drive (at least 8GB) or a different method to deploy an OS to the machine (like PXE boot)
- Rufus or another tool to create bootable drive
- Ubuntu Server ISO file (option 2 manual server installation)
- Way to access computers like SSH or physical access
- Virtualization tool like VMware Workstation or VirtualBox if you want to virtualize the master node (I use VMware Workstation Pro 16)
- If you want to use SSH, you can use the built-in SSH tool in CMD in Windows or Putty. I personally really like Terminal from the Microsoft store

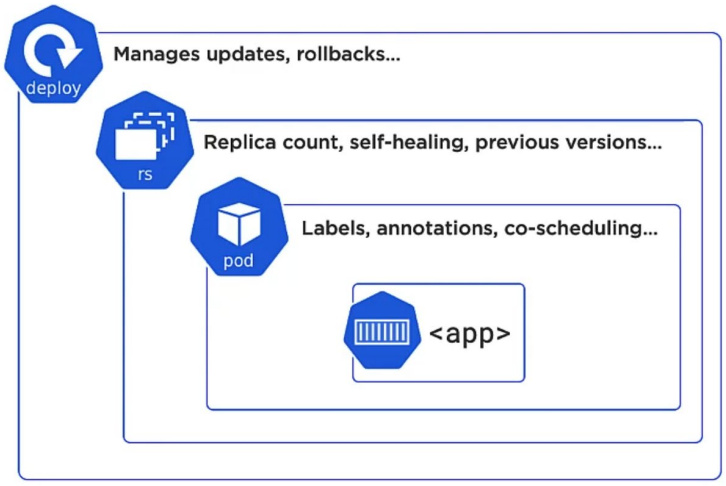
What is Kubernetes mainly used for? What are the main components?

Kubernetes Architecture

Kubernetes is a big orchestrator to organize containers. You say to Kubernetes I want this, and Kubernetes will make it happen. Kubernetes is mainly used for scaling containerization across multiple systems. These systems are called nodes.

Main components

- Cluster
    - Contains master and worker nodes
- Master node
    - In charge, cluster control plane, monitors cluster, makes changes and responds to events
    - Kube-apiserver: front-end to control plane, consumes JSON/YAML files
    - Cluster store: persists cluster state and config, based on etcd, performance is critical
    - Kube-controller-manager: Controller of controllers, watch loops, reconciles observed state with desired state
    - Kube-scheduler: Watches for new work tasks, assigns work to worker nodes
- Worker node
    - Does the work, listen to the master(s)
    - Kubelet: main Kubernetes agent (Linux of Windows), registers node with cluster, executes Pods, Reports back to Masters
    - Container runtime: can be docker or containerd, pluggable: Container Runtime Interface (CRI)
    - Kube-proxy: Networking component, Pod IP-addresses, basic load-balancing
- Manifest
    - Declares desired state
- Pod
    - containers of Kubernetes, where the app lives
    - Shared execution environment, networking, memory, filesystem
    - Pods contain containers
    - App container: contains the ap
    - Mesh container: handles traffic going in/out the pod
    - Pods don't scale but do behind the scenes stuff like annotations, labels, policies, resources, co-scheduling containers, …
- Services Object
    - keeps track of IP-addresses of pods
    - Stable name and IP for clients that connect to the service (DNS)
    - Updates list of IP-addresses when new pods arrive or when pods die
    - Provides basic load-balancing
    - ClusterIP: IP assigned by Kubernetes for the cluster
    - DNS name registered with cluster DNS based on CoreDNS technology
    - NodePort: exposes a port to the outside so you can connect
- High level controllers
    - deployments (stateless apps), cron jobs,
    - Advanced features,
    - Implemented by controllers
    - Deployment Controller: Deploys changes to the cluster
    - Reconciliation loop: Checks if Desired State is the same as observed state
    - Can do self-healing
- Scaling
    - create/remove pods



The Declarative Model and Desired State

Declarative model: Describe what you want (desired state) in a manifest file.
Manifest is posted to the API-server.
Observed state can be different from Desired state, for example node fails -> Kubernetes does everything to reach Desired state.

Concept: master nodes and worker nodes

An important concept in Kubernetes that I will touch on briefly is master nodes and worker nodes. A Kubernetes cluster contains master nodes and worker nodes. The master nodes schedule tasks and tell the worker nodes what to do. The worker nodes listen to the master nodes. To have a robust cluster it is good practice to set up a cluster with multiple master nodes. Make sure to avoid 2 masters because this could cause a split-brain situation. I will not show how to do this but there is a lot of information in the official Kubernetes documentation.

What makes this project function: Nvidia GPU-operator

The core of this whole project is Nvidia GPU-operator. To use Nvidia GPUs in clusters there is a lot of configuration work to do. You need a driver for each GPU, an Nvidia runtime, Nvidia toolkit, Nvidia device plugin and an Nvidia DCGM-exporter. You can manually configure all of this, but the Nvidia GPU-operator is designed to do all the work for you.

Once the GPU-operator is installed on a cluster it will automatically deploy a node feature discovery pod on each node to check the available hardware. When this pod finds an Nvidia GPU it installs the driver and makes all the changes so the GPU becomes available to the cluster. GPU-operator also updates the driver. When nodes come and go the GPU-operator validates everything each time a node comes online.

Some explanation of why I made certain decisions

Before proceeding with the installation of Kubernetes I want to explain some decisions I made. First of all is the choice of what type of cluster. There are many “flavors” and I looked at 5 of them. I compared K3s, K8s, MicroK8s, RKE and RKE2. My main focus is the support for GPUs.

K3s

- Quick deployments
- Limited in terms of databases (depends on what types of deployments run on the cluster)

- Created with support for IoT Edge devices such as a Raspberry Pi
- Lightweight
- Created by Rancher Labs

K8s

- Sturdy
- Needs more resources than K3s (this is still fairly limited compared to the computing power of desktops with GPUs)
- External tools required for GPU support
- Created by Google (successor to Bork and Omega)

MicroK8s

- Built-in GPU support in the form of Nvidia GPU-operator
- Lightweight
- Support for Edge devices
- Has no database limit like K3s
- Production Ready?
- Created by Canonical

RKE

- Rancher uses RKE(2) under the hood
- External tools required for GPU support
- Runs entirely in docker
- Created by Rancher Labs
- Uses declarative way (single yaml file to create and tweak cluster)

RKE2

- Built-in containerd needs to be tweaked for GPUs
- External tools required for GPU support
- Created by Rancher Labs
- Uses declarative way (single yaml file to create and tweak cluster)

Conclusion

K3s is built for IoT Edge devices and because of this I did not use it. K8s is sturdy and good for almost every task but it is not simple to get a running cluster if it's the first time experimenting with Kubernetes. MicroK8s looks very promising with built-in GPU support. I did not work with MicroK8s because of the debate online if MicroK8s is considered production ready.

Because this is my first experience with Kubernetes, I want to be able to setup a cluster fairly quickly so I can start testing GPU support. That is why I use Rancher. A cluster is created from scratch in about 5 - 10 minutes. Rancher uses RKE and RKE2 to provision clusters.

I first used RKE2 because it comes with a built-in container runtime, containerd. It is supposed to have better support for GPU-operator than RKE and it's more future proof. The only catch is that the built-in containerd does not support GPU-operator and you have to install a separate container runtime and point RKE2 to this runtime. I got this working, and the GPUs were usable. You can find the steps in the know issues section. While I was testing the cluster something updated which broke the setup with RKE2 and I had to find another solution. This leaves RKE, still great and used in a lot of clusters but eventually RKE2 will replace RKE. That is why I wanted to make this project with RKE2.

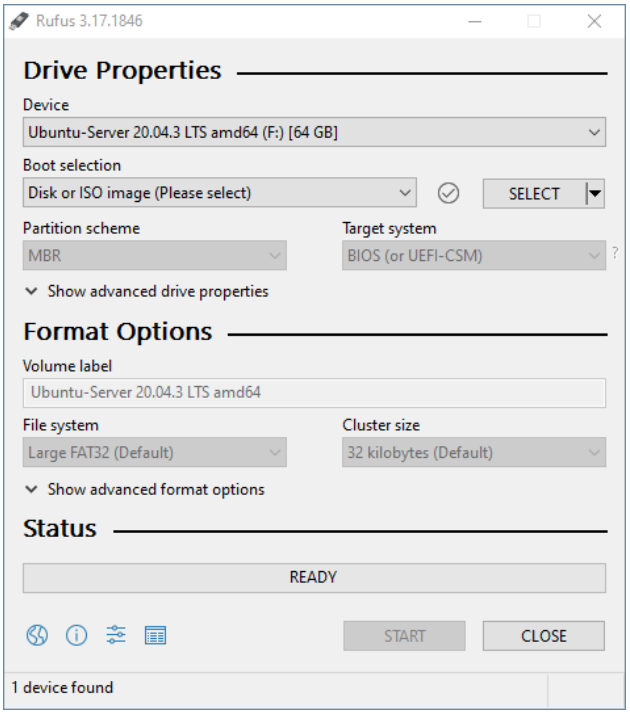
Some basic commands used in this manual

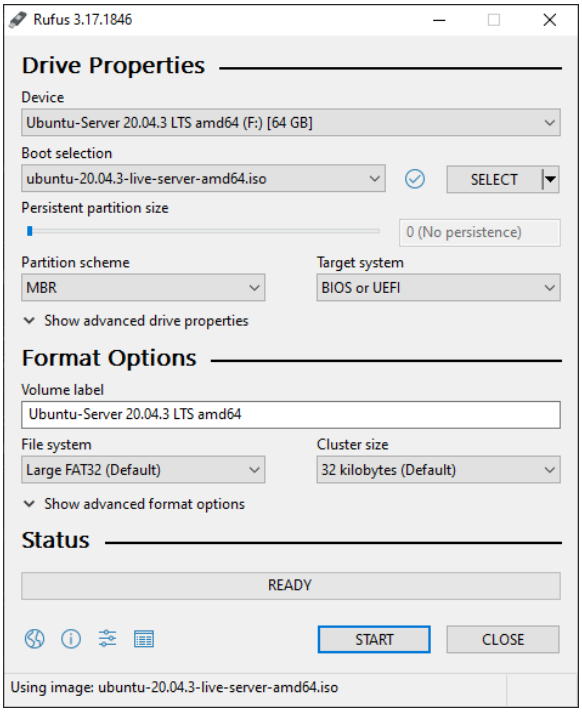
- Sudo, used to run commands with root privileges.
- Ls, with this command you can list files and folders in a path
- Nano/vim are text editors. Vim is the more advanced text editor for experienced users

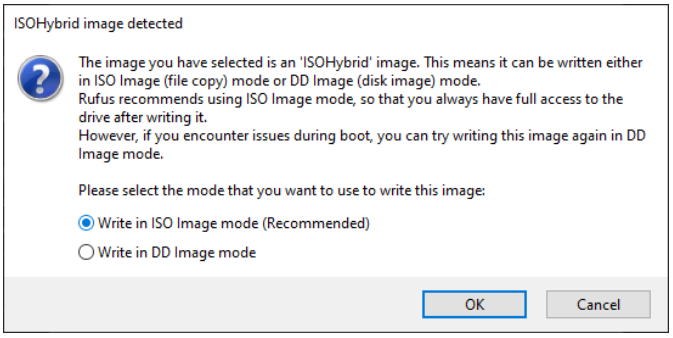
Creation of bootable drive

This section shows how to create a bootable USB drive with the help of Rufus.

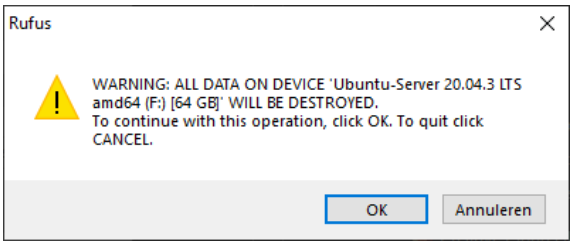
When you open the program, you are greeted with the following screen. Rufus automatically tries to select the correct drive. Make sure the drive is correct. After that select the ISO file you downloaded and change the name of the volume if needed. Leave everything else default. Then click on start.







After that you will get a warning that all the data on the drive will be erased. Make sure all the important data is backed up. Then click ok.



Rufus will now create the bootable drive and will prompt you when it's done.

Installation of Ubuntu Server

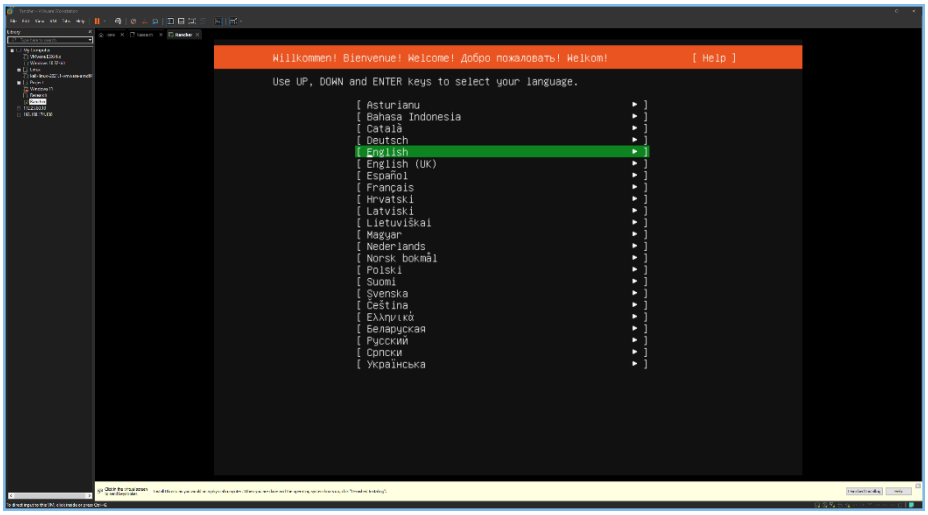
Now we will install Ubuntu Server. All the screenshots are taken from VMware workstation Pro but if you install Ubuntu Server on a physical machine the steps should be the same. It is a default installation so if you are familiar with this you can just skip to Prepare master node

I configured my virtual machine with 8 CPU cores and 8GB of memory. You can go with less CPU cores, but you need 8GB of memory or more.

First of if you are installing Ubuntu Server on a bare metal computer make sure the computer boots to the USB drive. If the computer does this not automatically go into the BIOS and manually select the drive or set it first in boot order. To access and change the boot order of your computer check the manual.

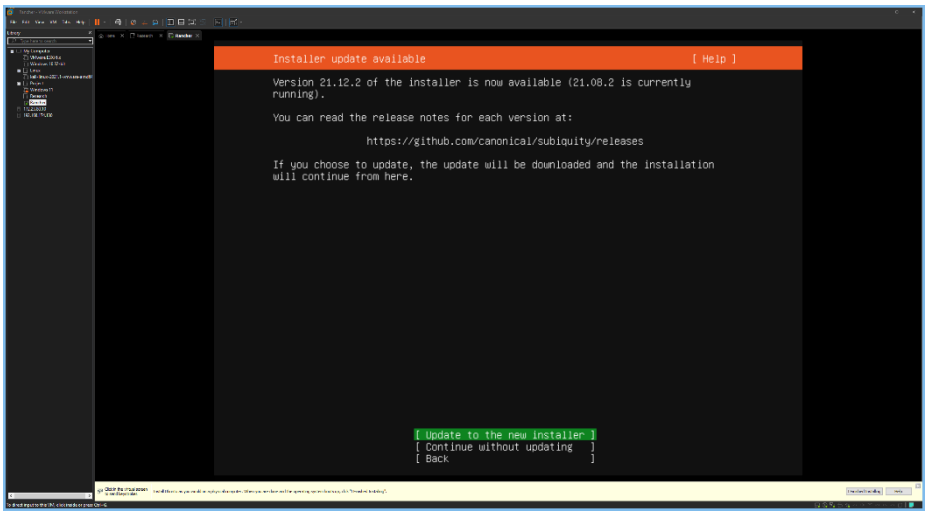
To navigate in the installer, you can use the arrow keys to go up and down. Enter to continue and spacebar to select something.

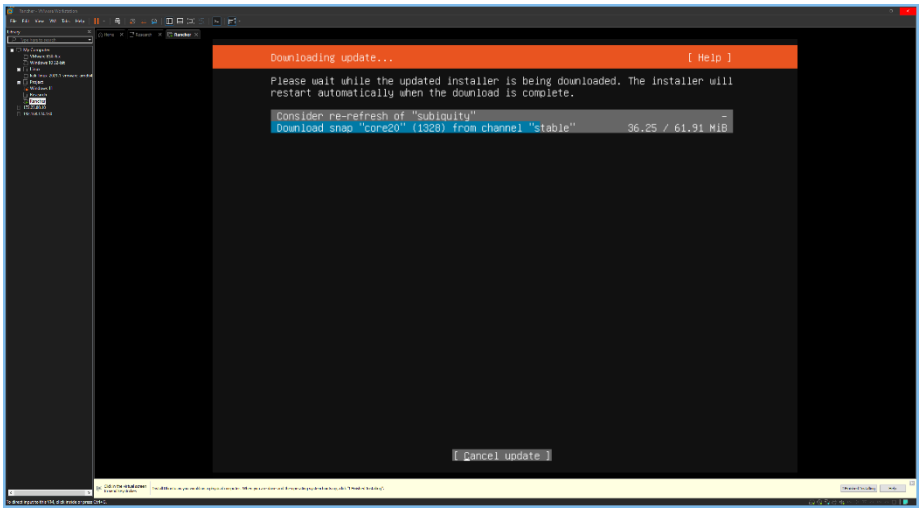
When the computer boots to the drive you will see a lot of text flying by. Eventually you will see the following screen



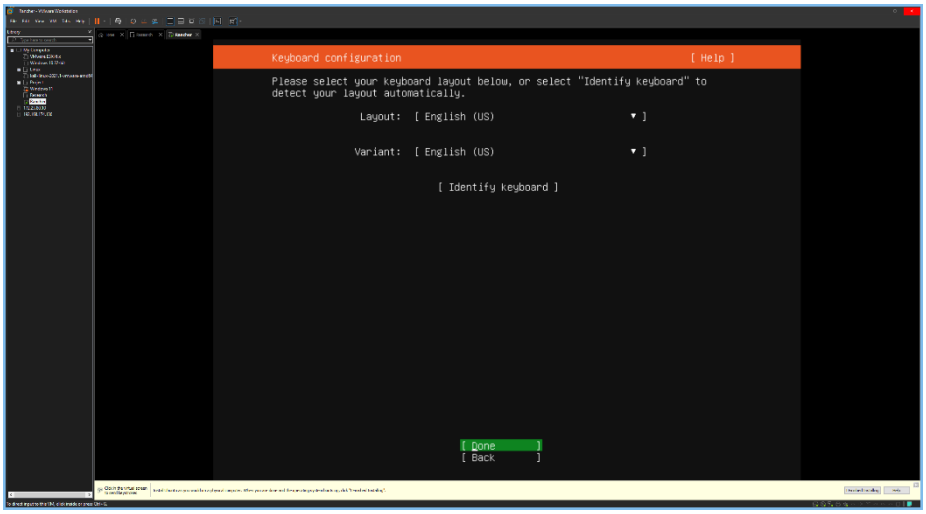
Select the preferred language with the arrow keys and then press Enter.

Next screen will ask you if you want to update to the new installer or continue without updating. I recommend updating to the new installer. To update make sure you are connected with the Internet.

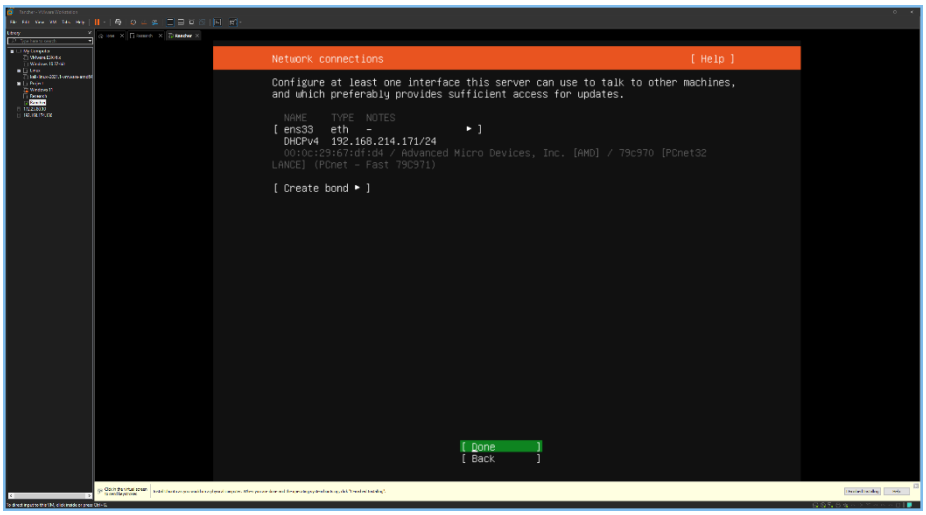




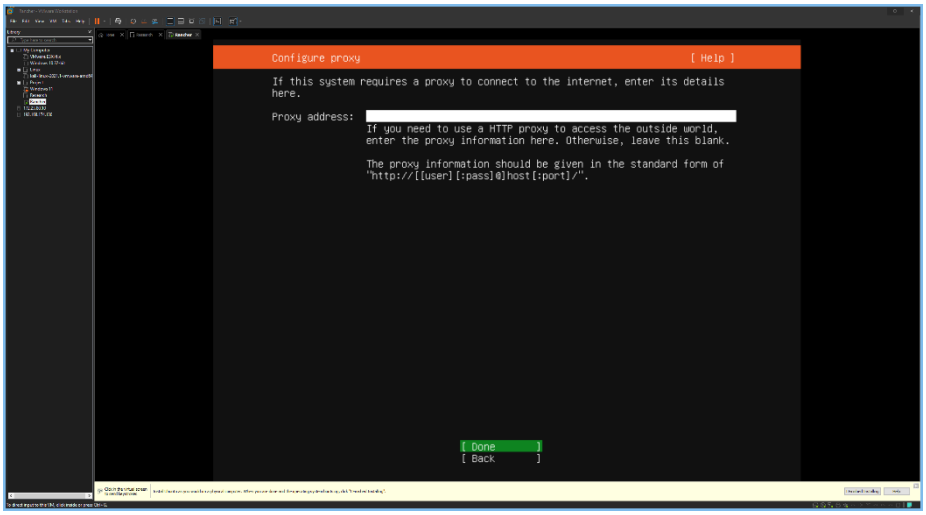
Next up is the configuration of the keyboard. I have a QWERTY keyboard, so I selected English (US) and variant English (US). Select the appropriate layout for your keyboard then continue.



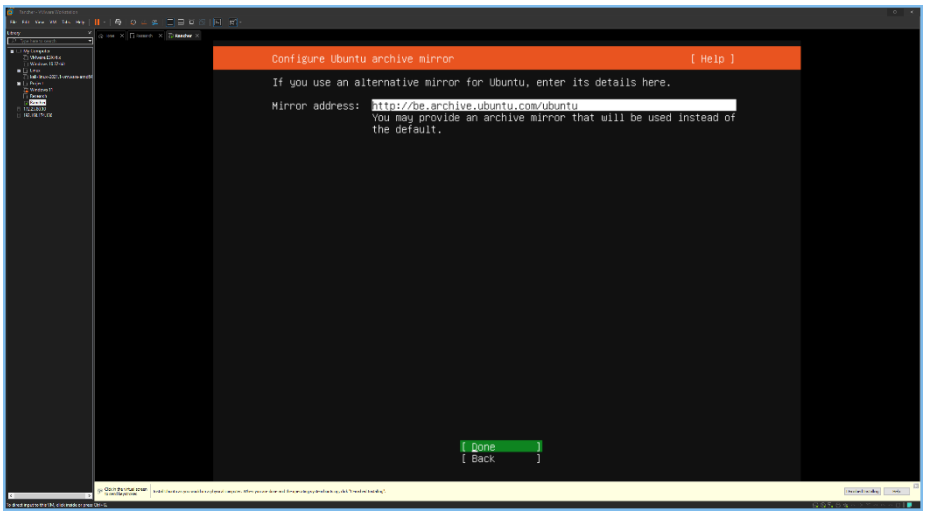
Next up is the configuration of the network interface. If you are connected with an Ethernet cable you should see an IP address appear after a second or 2. You can also continue without an Internet connection.



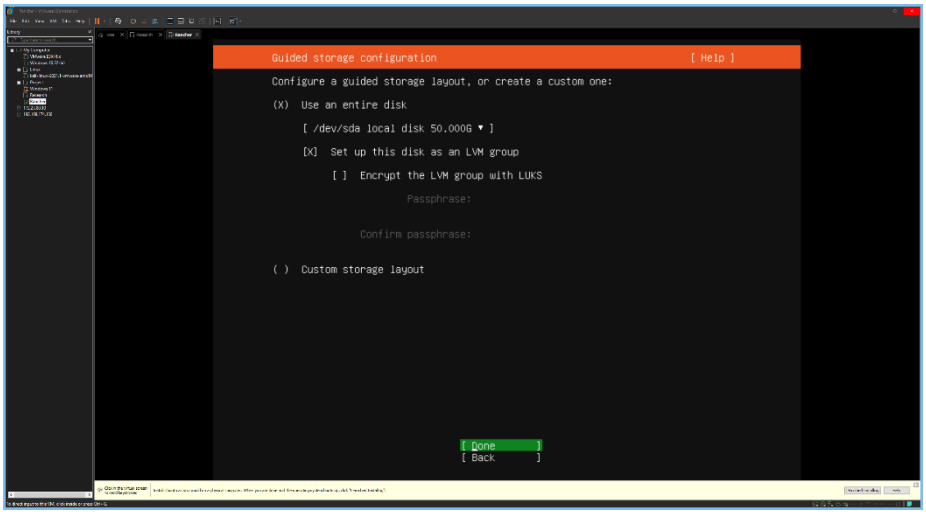
After that you can configure a proxy address. Most people don't need this but if you are not sure check with your network administrator. If you are trying this project out at home, you can probably skip this step.

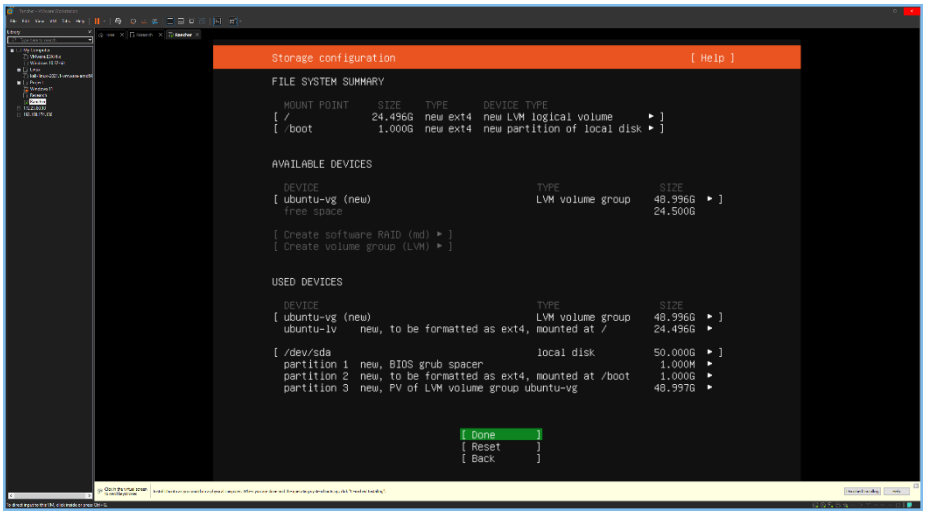


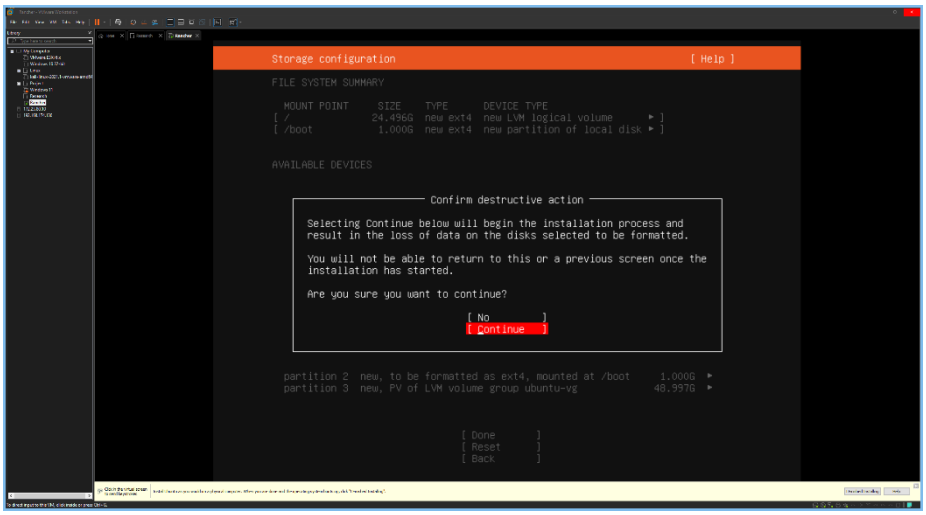
The following that you need to configure is the Ubuntu archive mirror. This is where ubuntu gets its updates. You can leave this default and go to the next step.



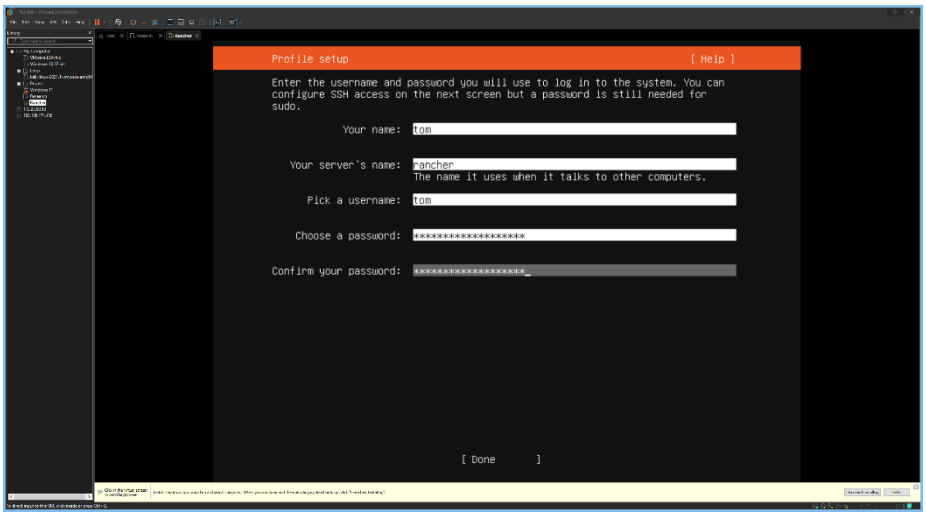
The next few steps are about storage and configuring the boot drive. For the sake of simplicity, I leave everything default. If you are familiar with Ubuntu or other Linux distributions, you can configure this so it best suits your needs.



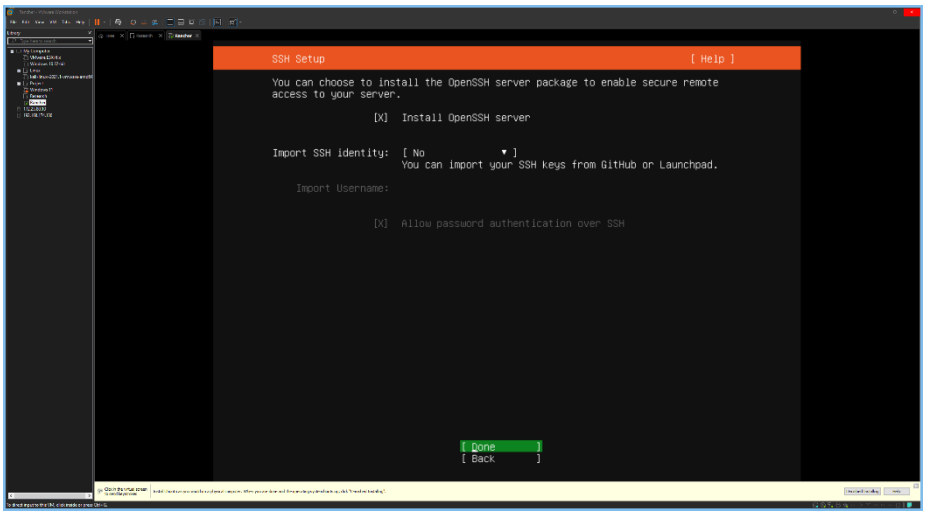




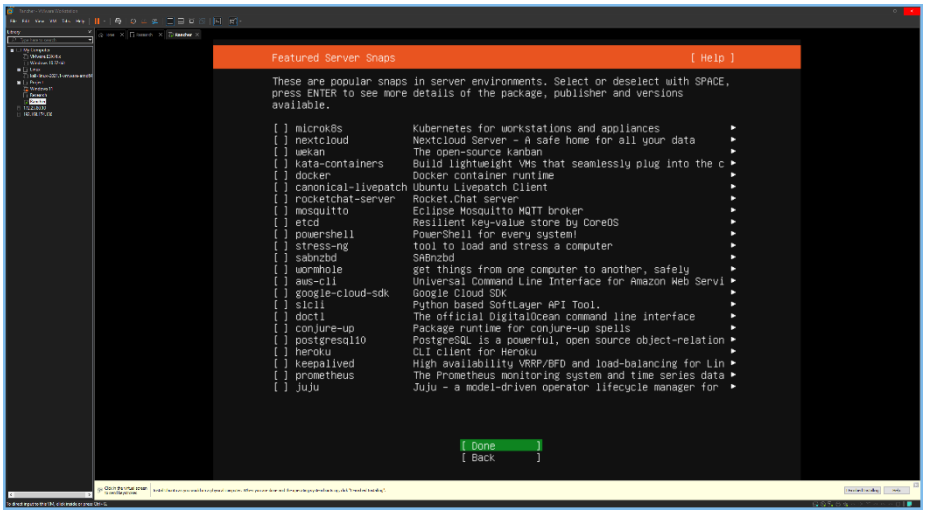
Next step is configuring a user profile. The server's name can be whatever you want but make sure it is unique on your network.

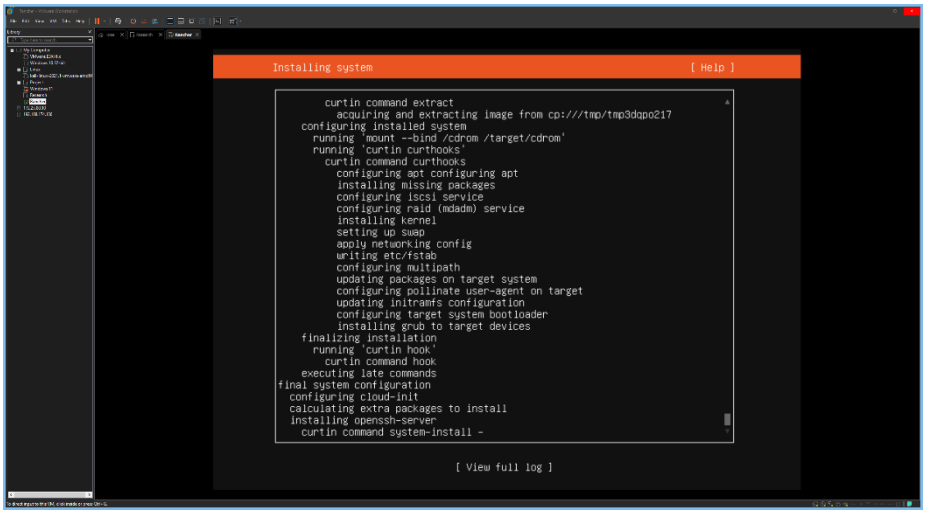


On the next screen you get the option to install OpenSSH server. This allows you to use SSH to connect to the computer and manage it remotely. If you are not planning on SSH you can skip this. To install it just tick the box with space.

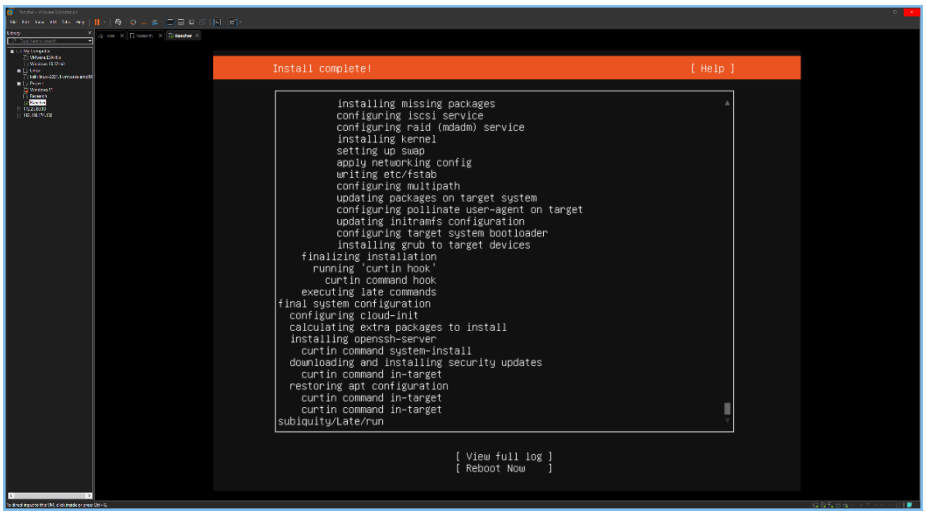


As a last step before installing the operating system you can select additional software to install. I stay with the default installation. Docker will be installed with a custom script. If you don't select software and you need it later, you can always install it with apt (package manager for Ubuntu). Click done and Ubuntu will be installed.



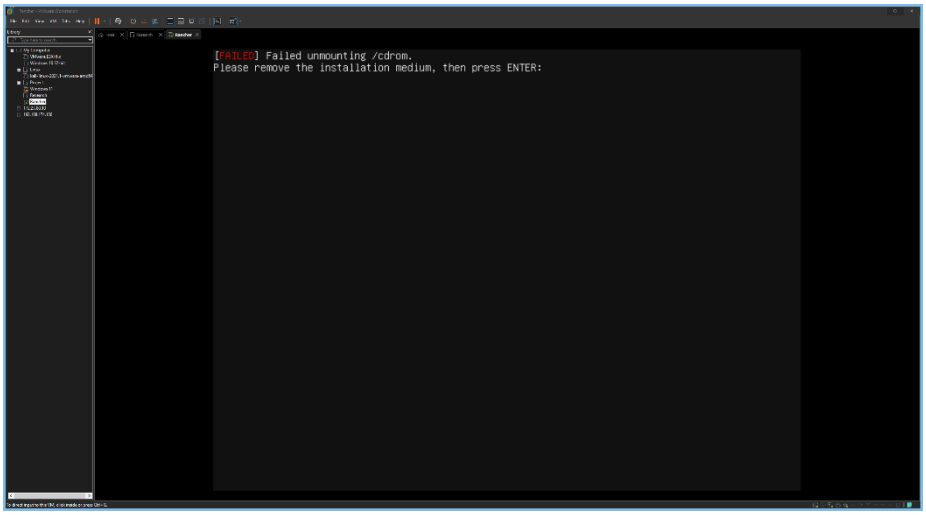


When Ubuntu is installed, you will be able to click reboot on the bottom of the screen.

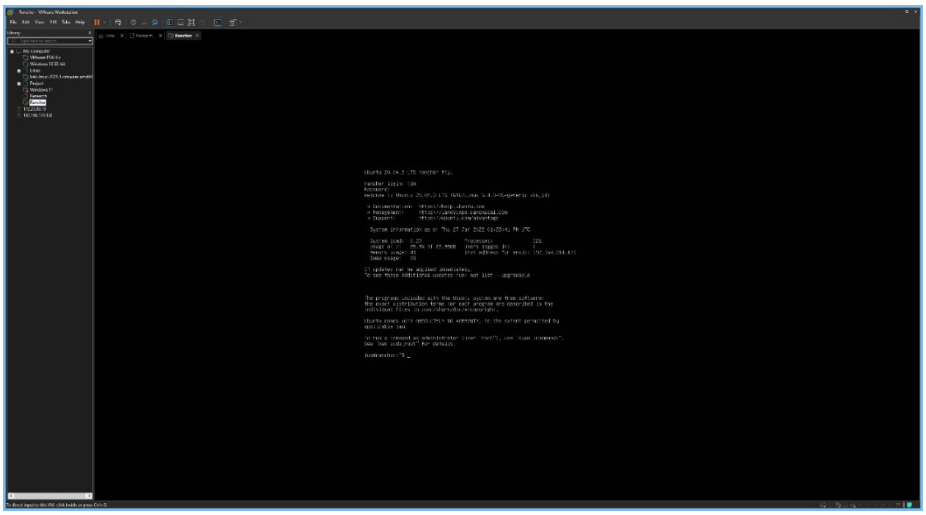


The computer/virtual machine will reboot, and you will be able to log in.

If you encounter the following screen on a physical computer, unplug the USB drive and hit Enter. If you get the same screen on a virtual machine just press Enter.



If everything went according to plan, you will be able to log in with the user you created during the installation.

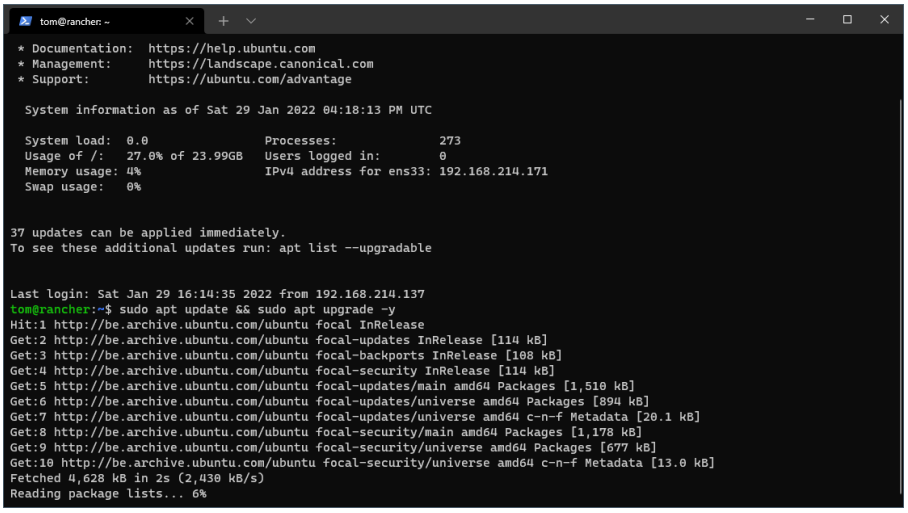


It is always a good idea to update the whole system after a clean installation. You can do this by running following command:

```
sudo apt update && sudo apt upgrade -y
```

Sudo is used to run commands with all the rights. You can compare this to administrator rights from Windows but keep in mind sudo is way more powerful and will not protect you from harming your system.

You may be prompted for your sudo password, this is the same as you user password. I changed this so I don't have to fill in my password every time I use sudo. This is considered bad practice so I will not show how to do this.



The next step is installing docker and configuring Rancher.

Prepare master node

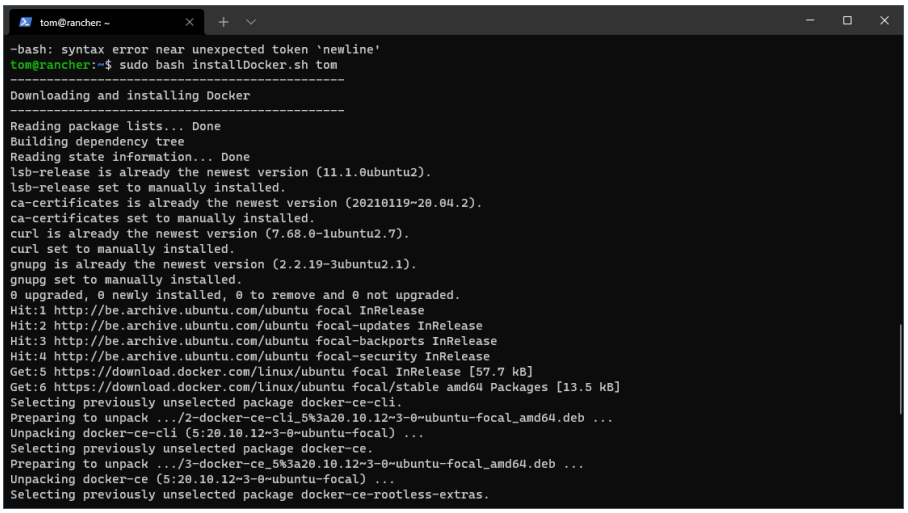
Now that the installation of Ubuntu server is finished, we can setup the master node and Rancher. Log in either with SSH or directly onto the computer. On my GitHub repository you can find custom

scripts to install Docker, Kubectl and Nvidia GPU-operator. In this manual I will be using these scripts. All the commands come from the official documentation of the respective tools.

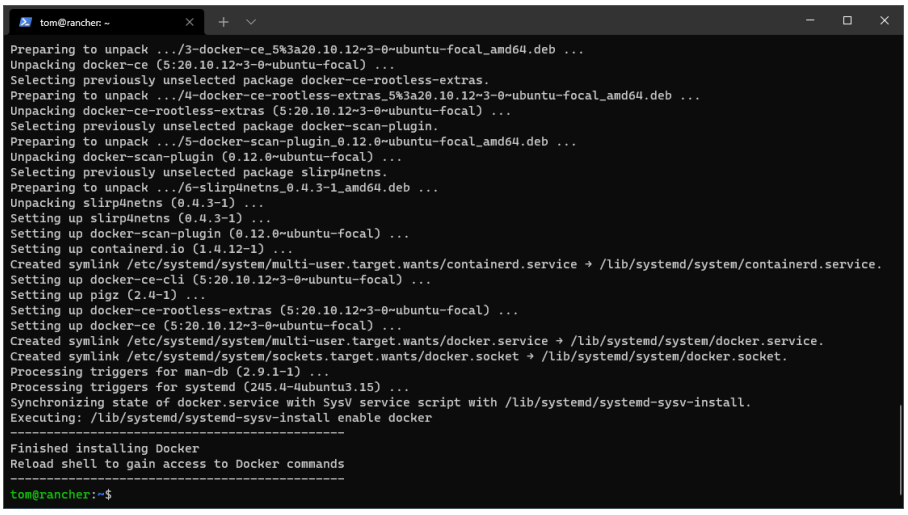
The scripts can be downloaded as a zip-file, or you can clone the repository. If you download the scripts as a zip-file on your main computer, you can transfer the necessary files with FileZilla or WinSCP or using an USB drive.

First install docker on the master node using the custom script. The other nodes will be configured later. The script uses a username as parameter. This user will then be added to the docker group so that user can execute docker commands. The script can be run with this command:

```
sudo bash installDocker.sh <username>
```



When the script finishes you are prompted with the message to reload the shell. Just log out and log back in.



Test the docker installation by running the following command:

```
docker ps --all
```

The output should look like this.

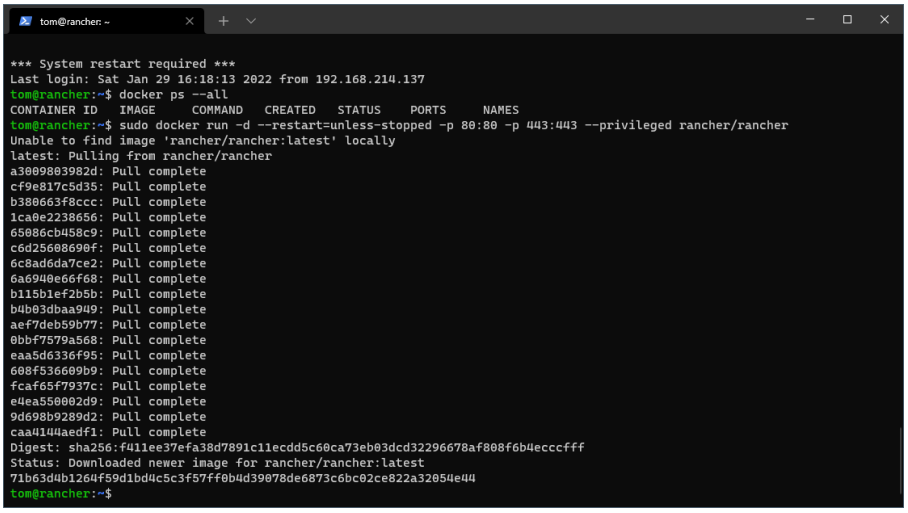
```
tom@rancher:~$ docker ps --all
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
tom@rancher:~$
```

Installation and configuration Rancher

Rancher is installed in docker with 1 command:

```
sudo docker run -d --restart=unless-stopped -p 80:80 -p 443:443 --privileged rancher/rancher
```

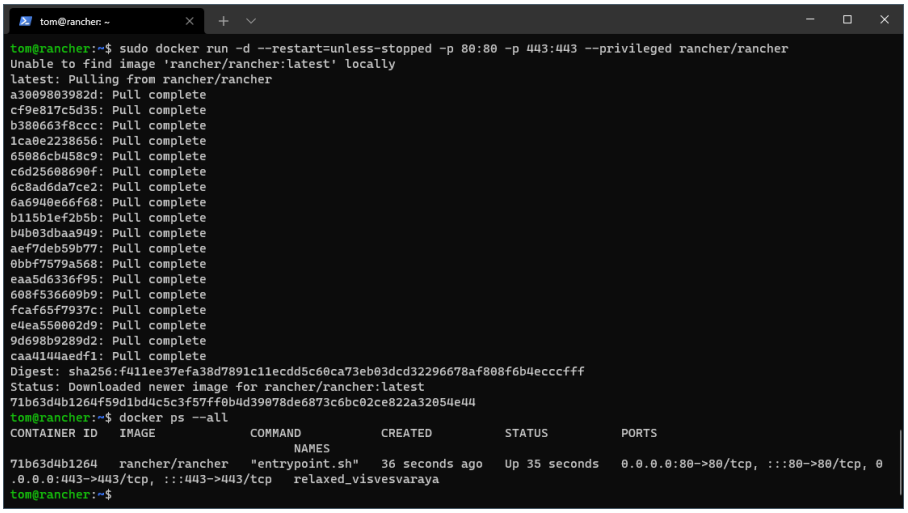
The command means the following: docker run container with rancher/rancher image. Always restart the container unless stopped manually. Use ports 80 and 443 with privileged rights.



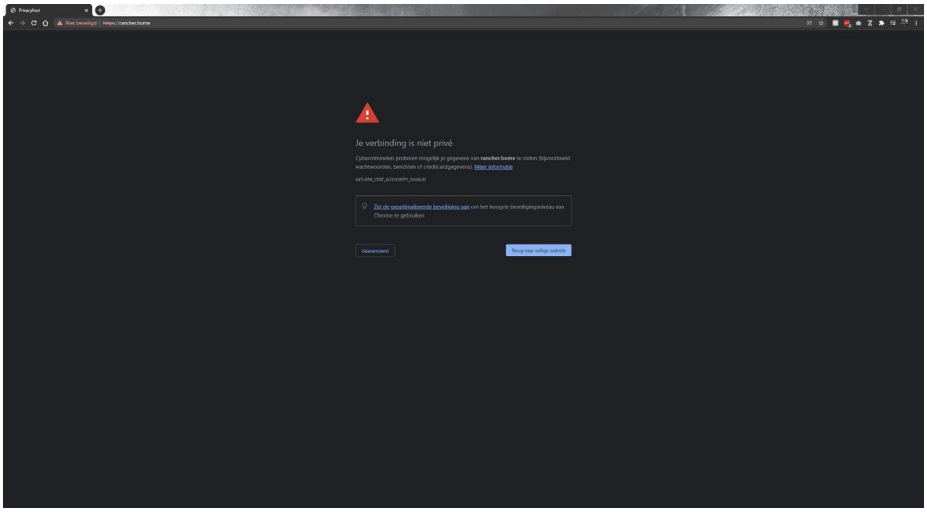
Docker will download the necessary files and then run the container.

To get the status of all the containers present on the system run the following command:

```
docker ps –all
```



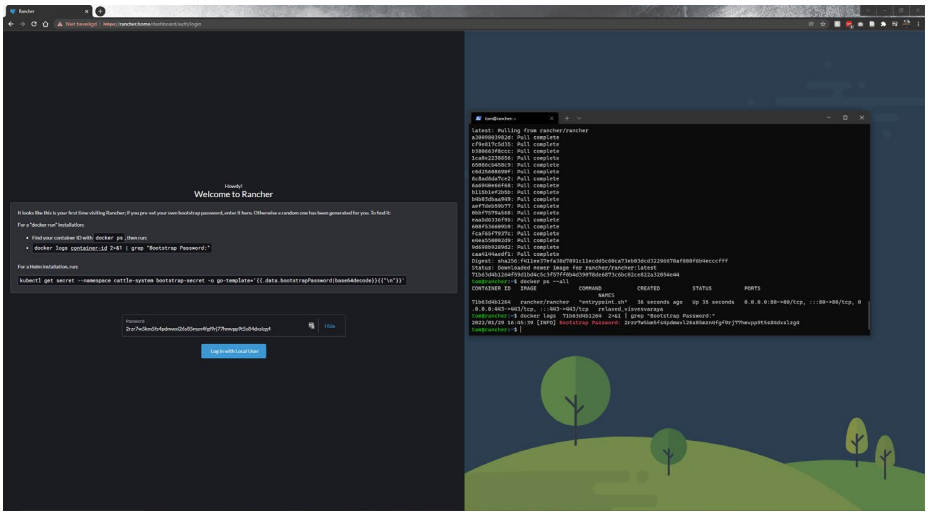
After about a minute the Rancher UI should be reachable through your browser. In the search bar type your IP-address. You should see the following screen:



This warning means that your browser cannot verify the certificate of Rancher. Rancher uses self-signed certificates. These certificates don't come from a known CA authority, so the browser doesn't trust them. You can click on advanced and proceed to the page (unsecure).

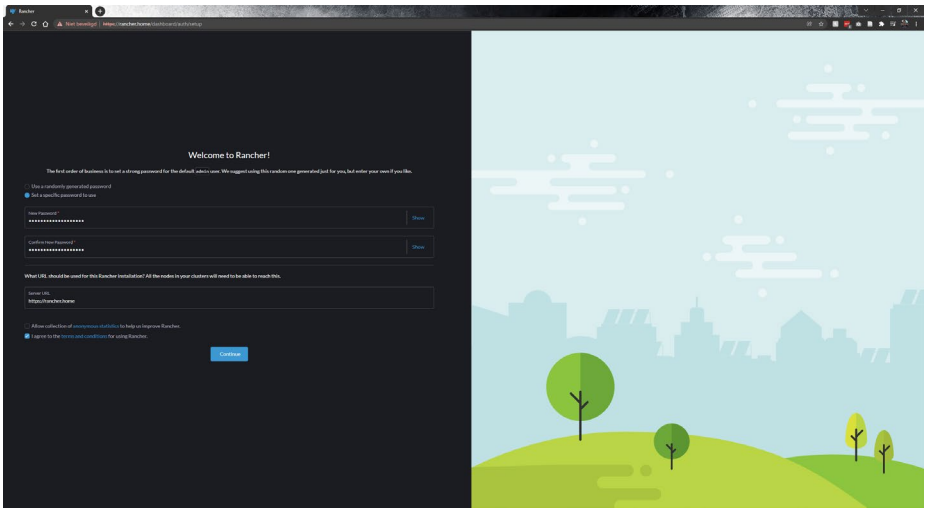
First setup

Now that Rancher is running you are greeted with a one-time setup. Follow the instructions for a docker installation and fill in the password you get. Then click on Log in with Local User.



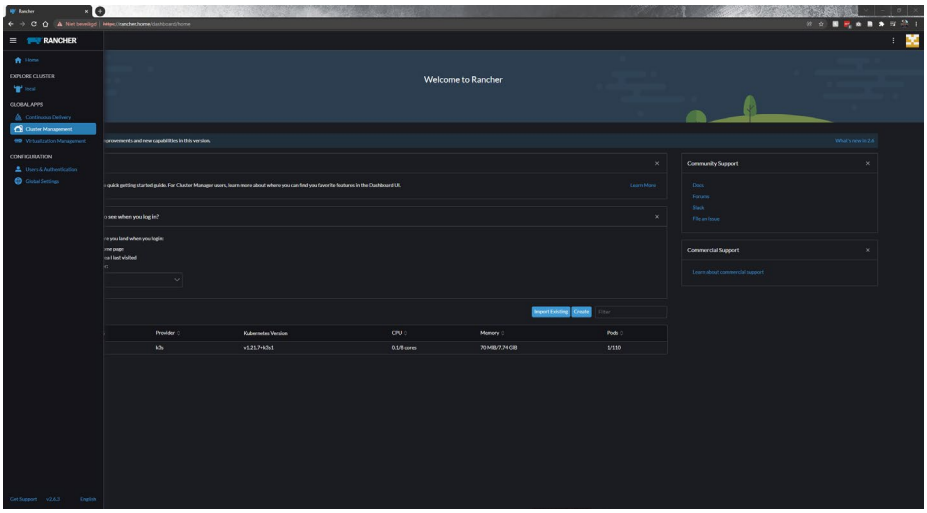
The next step is choosing if you want to use your own password or to keep using the generated password. I will change the password to something that is a bit easier to remember.

The URL that should be used is the URL that you accessed Rancher with. It is important that all the computers need to be able to reach this URL. In my example I use https://rancher.home but that is because I use a custom DNS server that maps this URL to the IP-address of the computer that runs Rancher. The best thing is to use an URL with the IP-address of the server. Accept the terms and continue.

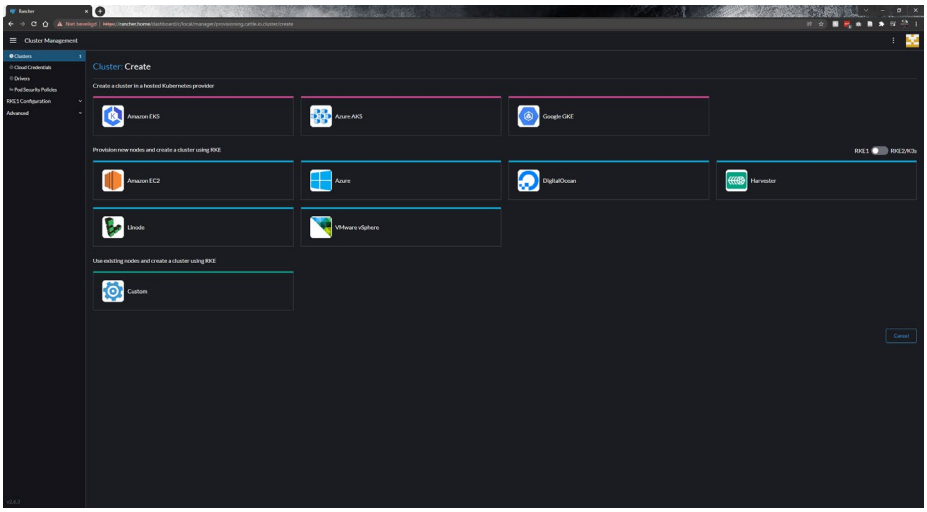


Deploy fist cluster

Now that Rancher is set up you can deploy your first cluster. Click on the 3 lines in the top left of your screen. Click on Cluster Management.

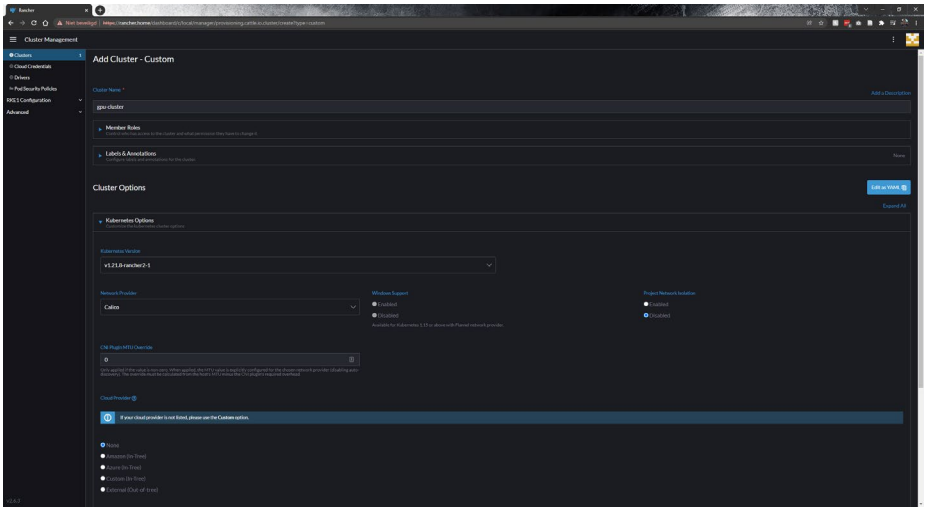


Here you will see that there is already a cluster present called local. This cluster contains one machine, this is the computer running Rancher. This cluster is always present and only used by Rancher. Click on create to create your own cluster.

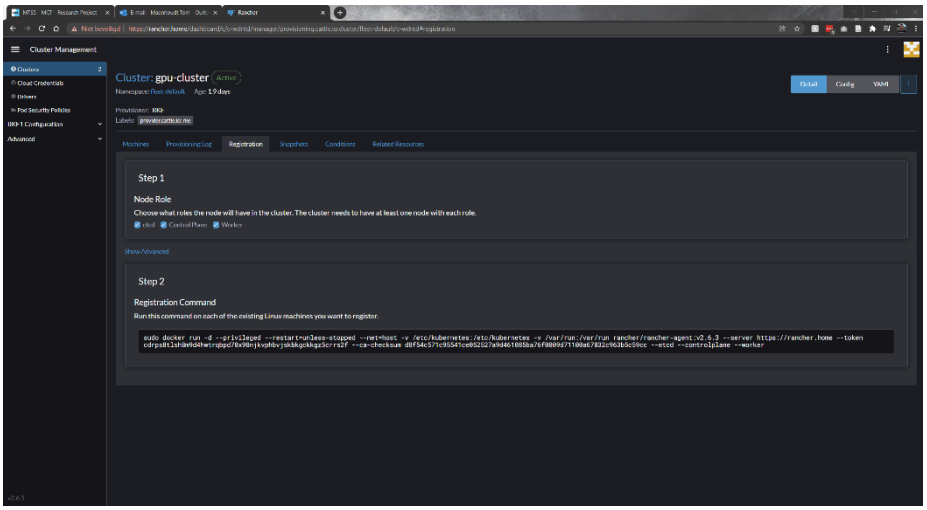


Here we will choose Custom because the entire cluster will run on premise. Here you also have the option to choose RKE1 vs RKE2/K3s. I use RKE1 because of the reasons I explained in the Some explanation of why I made certain decisions section.

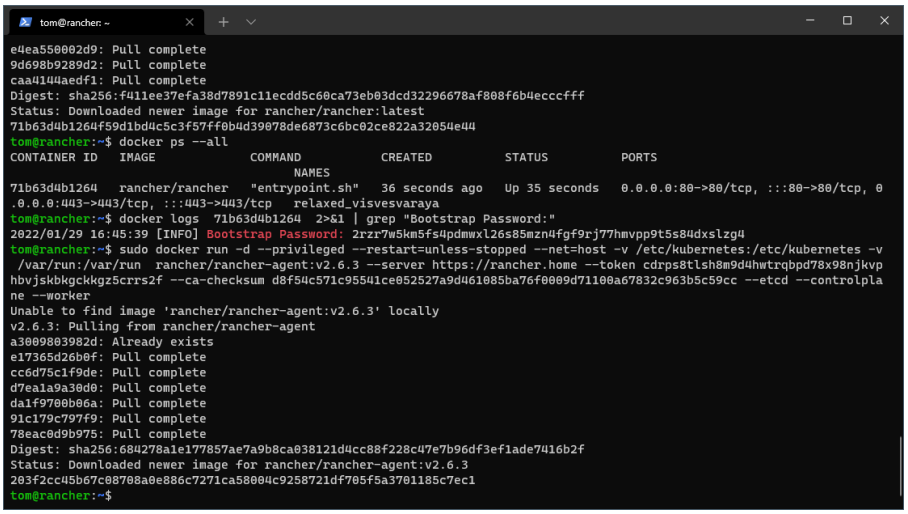
I named my cluster gpu-cluster because this cluster is focused on GPUs. Make sure you change the Network Provider to Calico because I experienced a lot of problems with Canal in my testing. You can find more explanation in the known issues section. Leave everything else default.



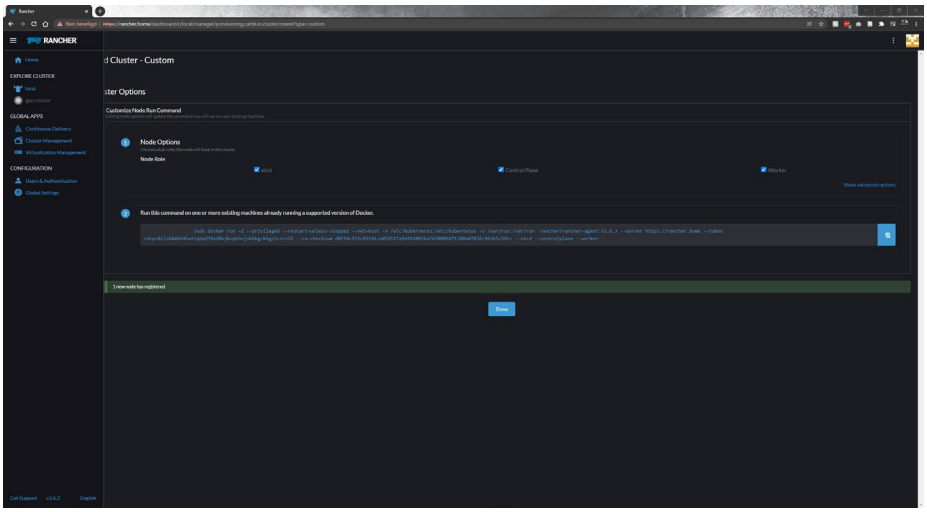
On the next screen you will find instructions to create the cluster. Every cluster needs an etcd host, control plane and worker. It is import that the master node is also a worker because the Nvidia GPU-operator will keep running on the master. This way the master and the GPU-operator stays stable if worker nodes come and go. The master is normally the etcd host and also has a control plane.

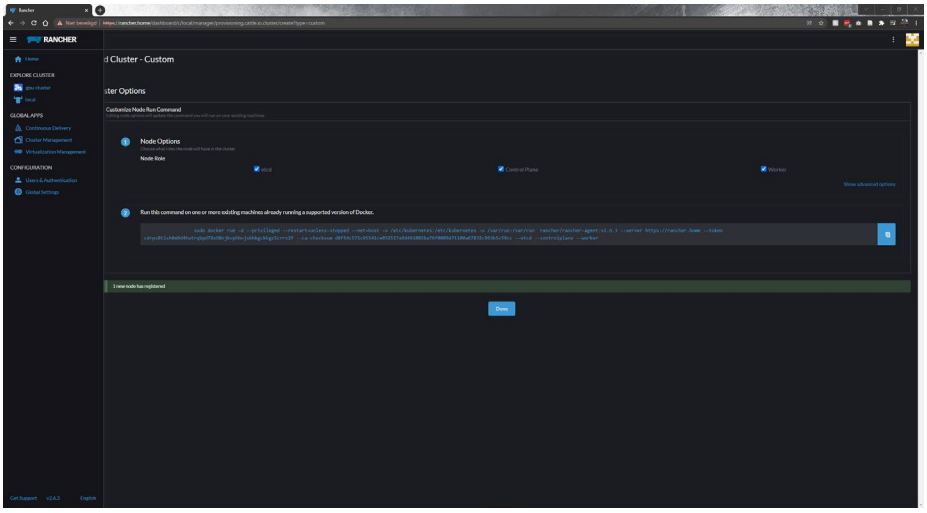


Copy the command and run it on the master node.

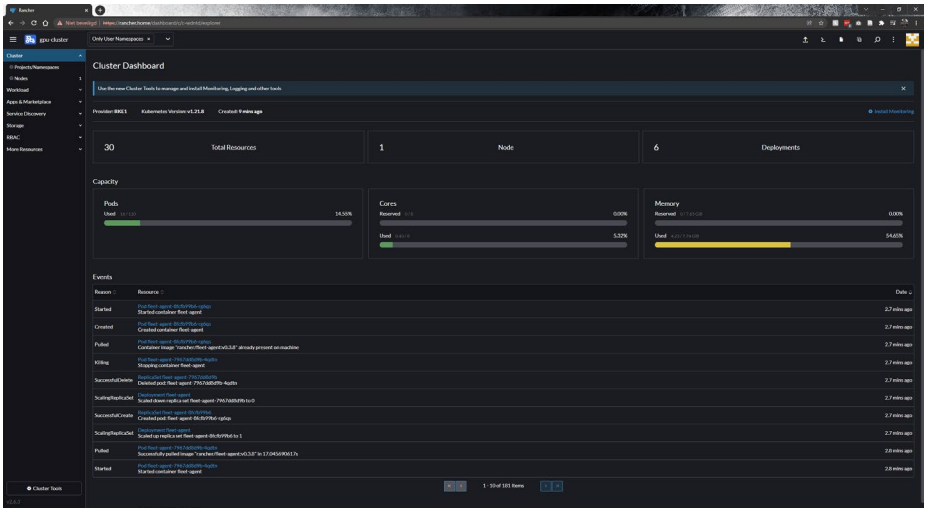


Docker will download the needed files and start creating the cluster. This will take about 5 – 10 minutes. You can check the status with docker ps --all. Not all containers will be created immediately but they will appear over time. You can also check the status of the cluster in Rancher. Click on the 3 lines and you will see your new cluster in the list. If the name of the cluster is greyed out like in the image, then your cluster is still provisioning. Once its name is blue your cluster is available.





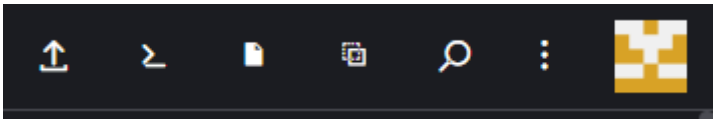
When you click on the name of your cluster you will be directed to the overview page of the cluster.



The next step is installing Helm and Nvidia GPU-operator.

Install Kubectl, Helm and Nvidia GPU-operator

To install something with Helm into your cluster you will need a kubeconfig file. This file contains all the info for Kubernetes tools to communicate with the cluster. You can download this file from the Rancher UI. In the top right corner of your screen next to your profile picture, you will see different icons.

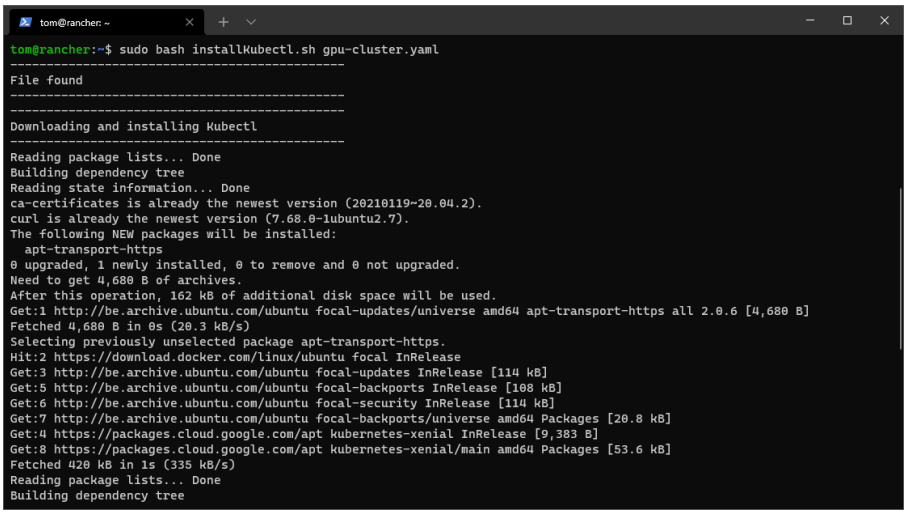


With the 3rd icon you can download the kubeconfig file. If you prefer to copy the content and create your own kubeconfig file you can click on the 4th icon. This file needs to be present on the computer that you will use to run Kubectl and Helm commands. I will use the master node for this. Transfer the file with FileZilla or WinSCP to the machine.

The script installKubectl.sh contains the installation steps for installing kubectl. This script also creates a config so the default cluster kubectl talks to is your newly created cluster. Another thing the script does is create an autocompletion script for bash. Kubectl is used to query a cluster through the CLI. It can also be used to deploy pods and deployments.

Install kubectl with the following command:

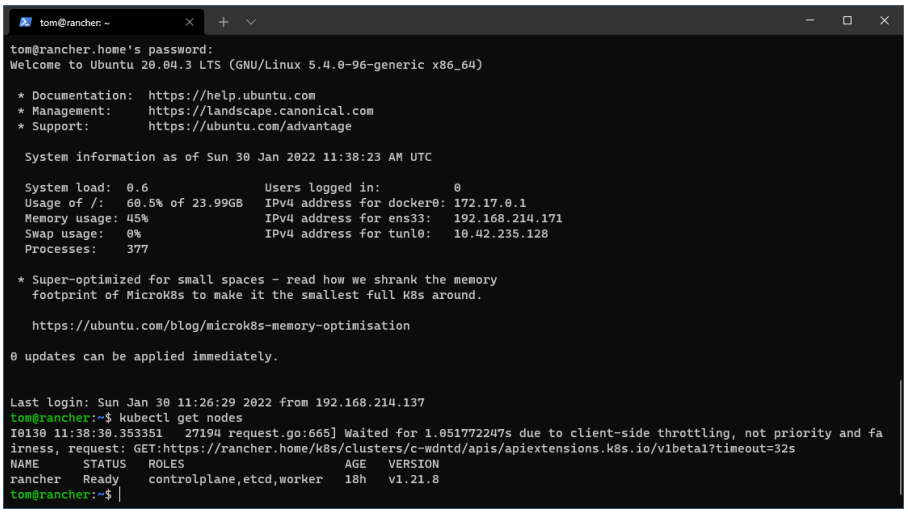
```
sudo bash installKubectl.sh <name-of-kubeconfig-file>
```



Once the script finishes you are prompted to reload the shell. Just log out and log back in. Once you did this you can test the kubectl config with the following command:

```
kubectl get nodes
```

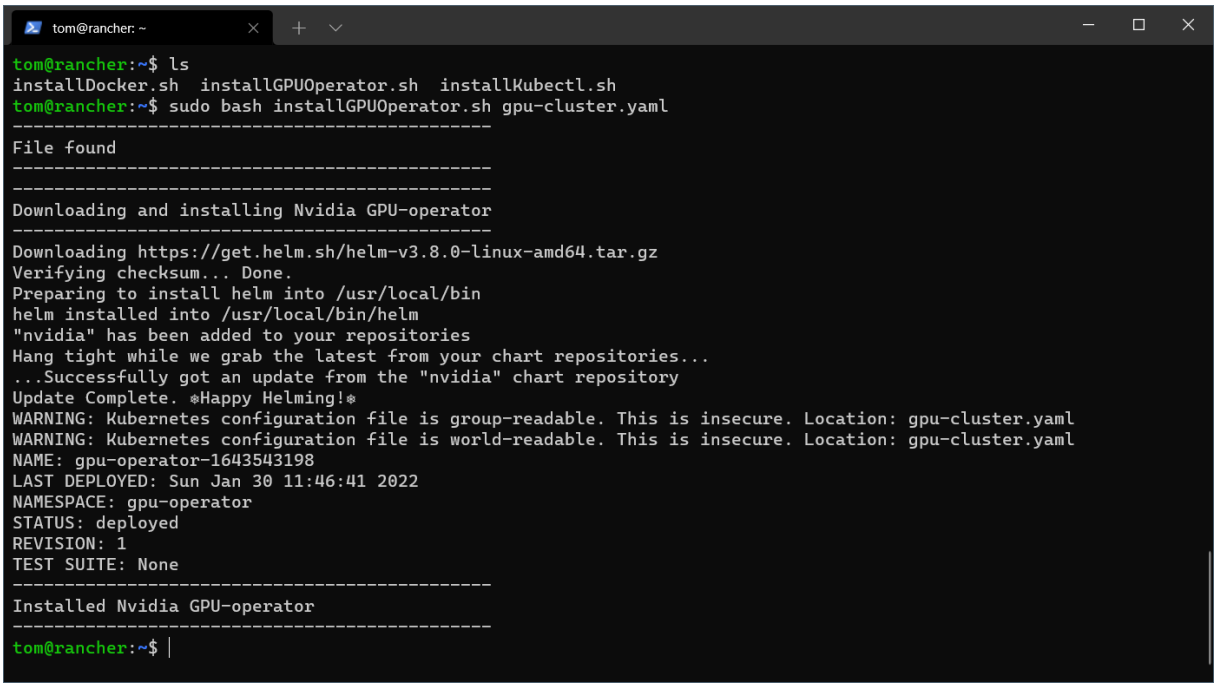
This command queries the cluster for all the nodes in the cluster. The output should look like this:



Now that kubectl is installed Helm and Nvidia GPU-operator are next. The Helm installation is included in this script Just run the installGPUOperator.sh script with the following command:

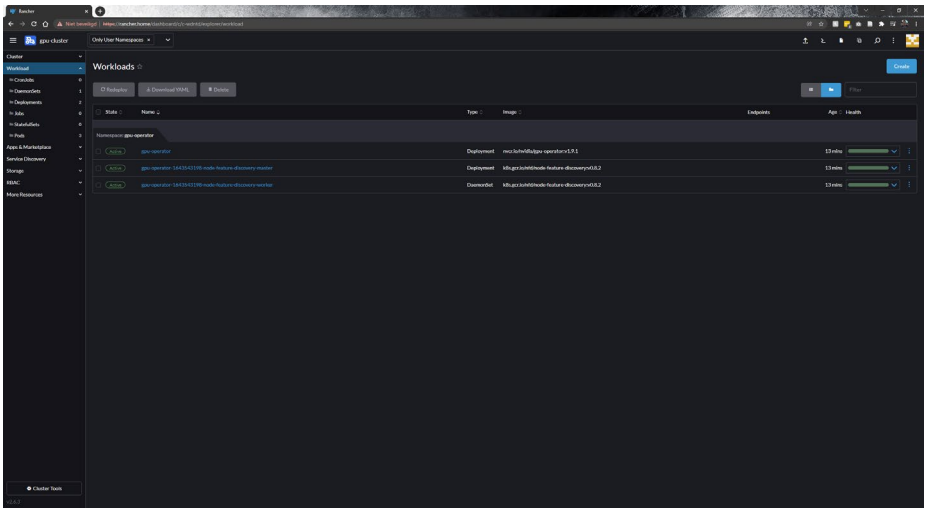
```
sudo bash installGPUOperator.sh <name-of-kubeconfig-file>
```

This can take a few minutes to complete.

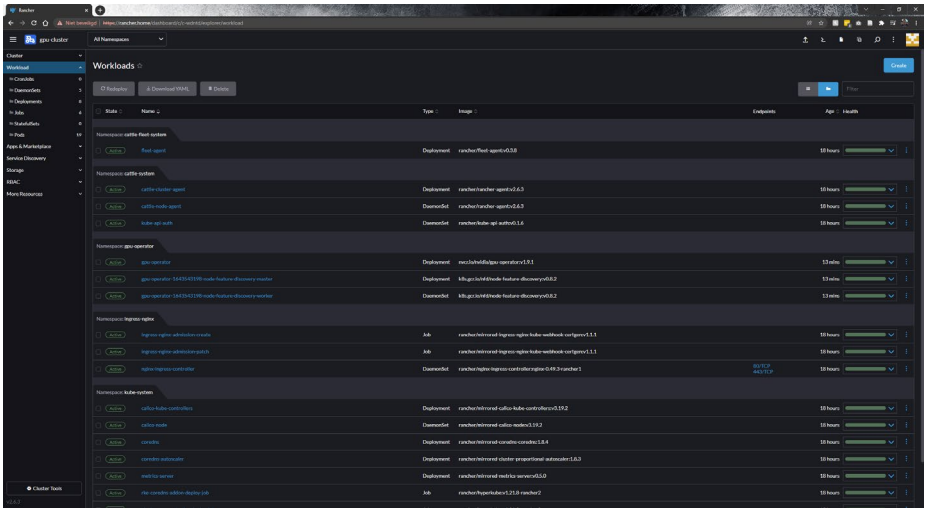


When the script completed you can also see the Nvidia GPU-operator in Rancher. Go to your cluster overview page and click on Apps & marketplace. Under Installed Apps you will find the GPU-operator. The GPU-operator deploys a few pods as part of a deployment on the master node. This is why the master node also needs the function worker. You can find these pods under Workload. The first time you visit this page there is a filter enabled so only pods in user namespaces are shown. If you want to see the other pods like the ones from Calico and other backend services, you can remove the filter in the top left.

This is with the filter enabled:



And this is without the filter:



Removing the filter can be useful if something doesn't work like it should or if a pod fails that belongs to a backend service. You will not see this with the filter. The disadvantage of removing the filter is that you may see a lot of information that you don't need.

You can also see that the script created a new namespace called gpu-operator. This is to keep things organized when the cluster expands to more nodes. By default, everything is installed in the default namespace if nothing is specified.

Now that the master node is fully setup, the worker nodes can join the cluster.

Prepare worker node

Before joining the cluster there are 2 things that need to be done. This is disabling swap and installing docker. Disabling swap is recommended by Rancher labs. You can learn more about swap here.

Log in to a computer that will function as a worker node in the cluster. I will be using SSH to connect to the pc remotely. If this is the first time logging in to the computer, it is a good idea to update the operating system first.

```
sudo apt update && sudo apt upgrade -y
```

Disable swap

Disable swap with the following command:

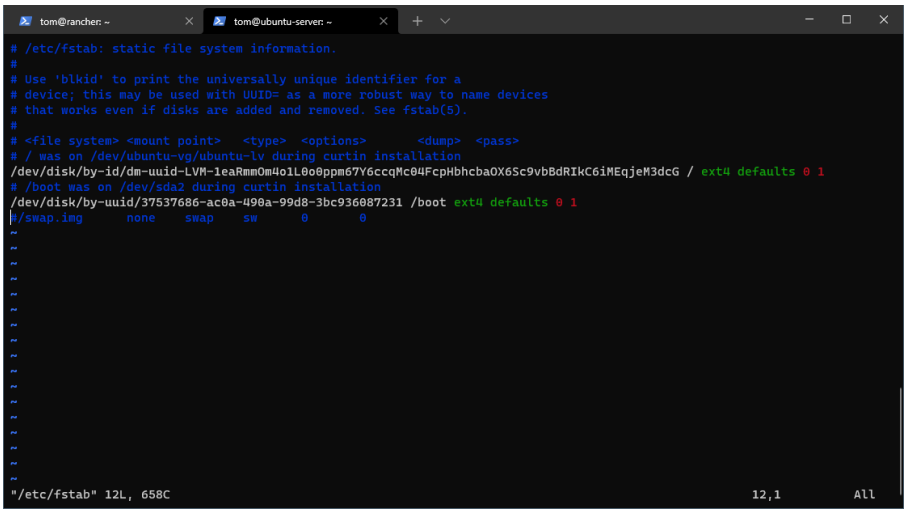
```
sudo swapoff -a
```

This disables swap but the change is reverted on reboot. To make this persistent edit the file /etc/fstab. I use vim as my text editor. If you are more of a beginner, I recommend nano. Both are preinstalled.

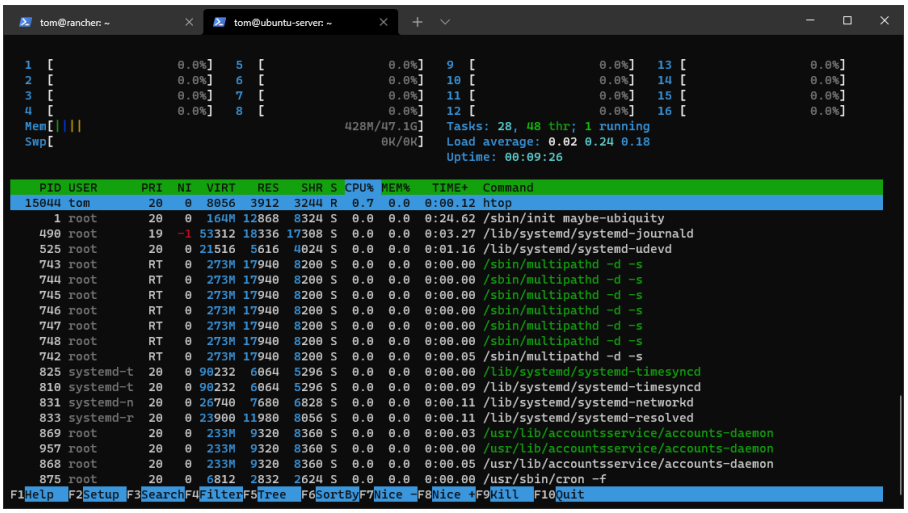
```
sudo nano /etc/fstab
```

The last line that mentions swap should be commented out. Just put a # in front.

If you are using nano save and exit the file by pressing crtl + x and then y, confirm with enter.



You can check if swap is disabled by running the command htop. This is kind of similar like task manager from Windows but for Linux. In the top bar you will see swap usage. This should say 0K/0K. This means that swap is disabled.



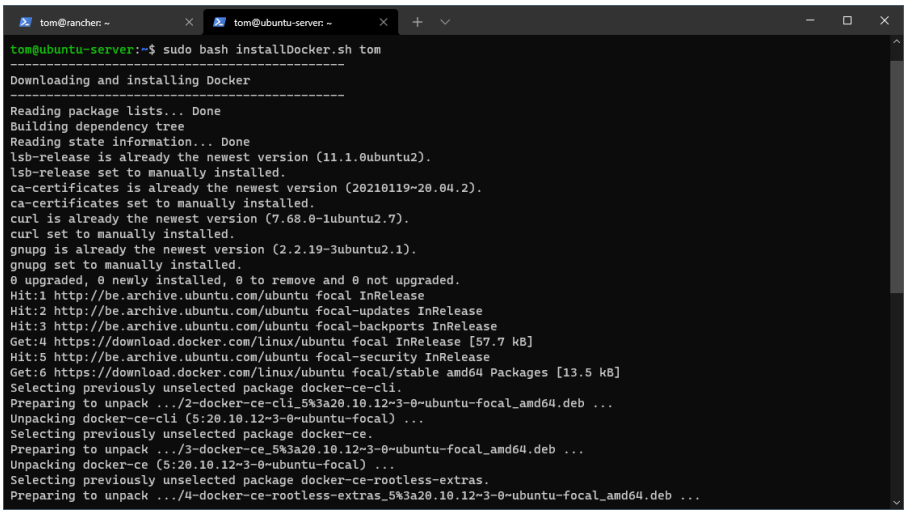
Exit htop by pressing F10.

Install docker

Transfer the script installDocker.sh to the computer and run it with the following command:

```
sudo bash installDocker.sh <username>
```

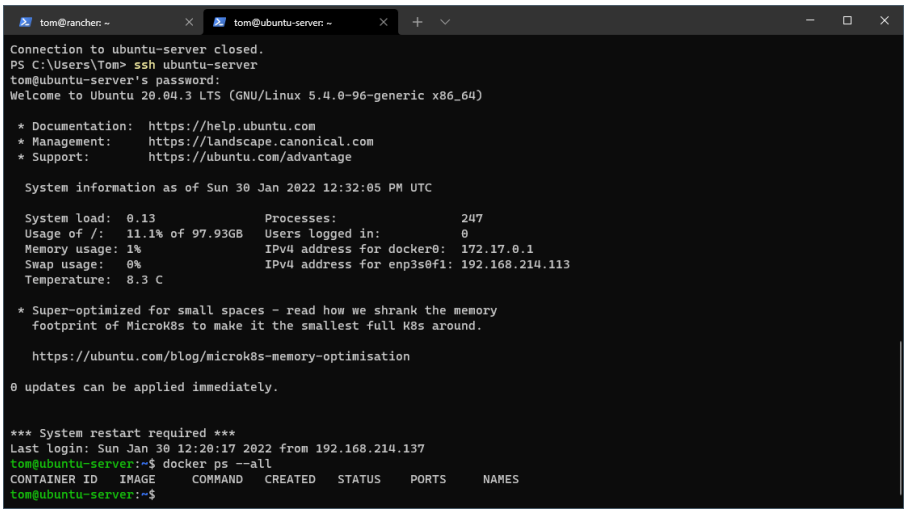
The username is the user that will be used for running docker commands.



When finished you are prompted once more to reload the shell. Log out and log back in. Test the docker installation by running

```
docker ps --all
```

The output should look like this:

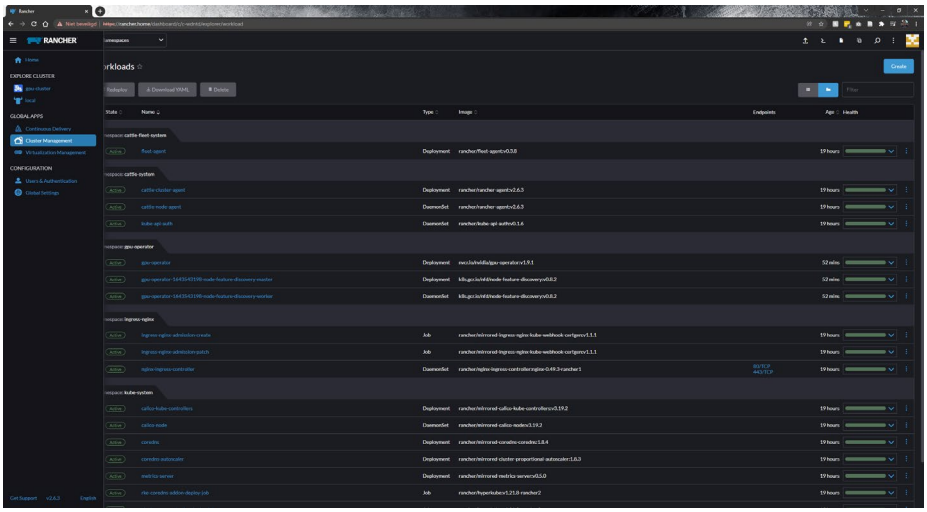


The computer is ready to join the cluster if swap is disabled and docker is installed.

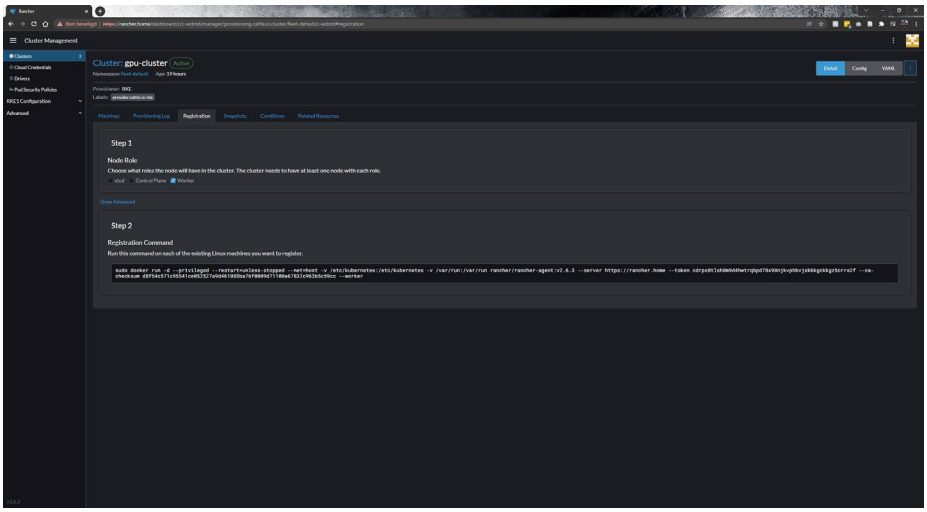
Join cluster

Joining a cluster is pretty easy. This is done by running one docker command on the computer that wants to join the cluster. This command is different for every cluster. You can find this command in the Rancher UI.

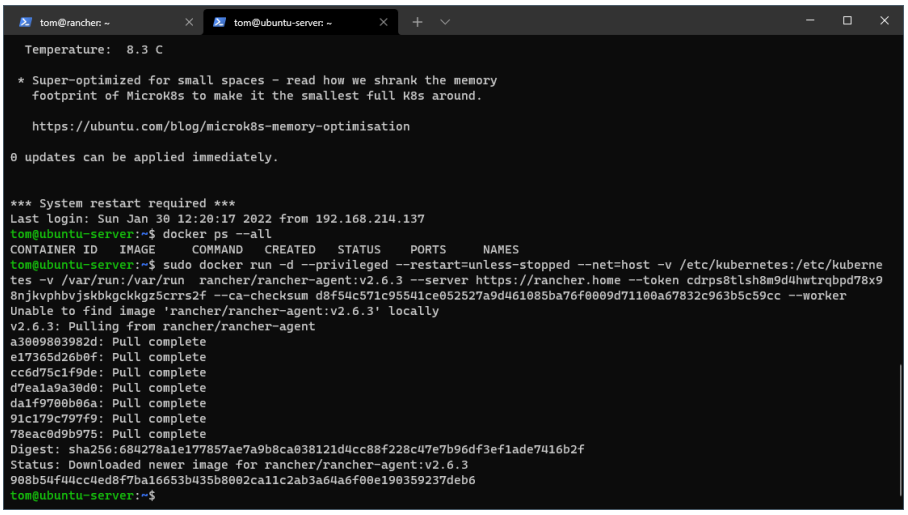
Go back to the Rancher UI and click on the 3 stripes in the top left corner. Choose Cluster management.



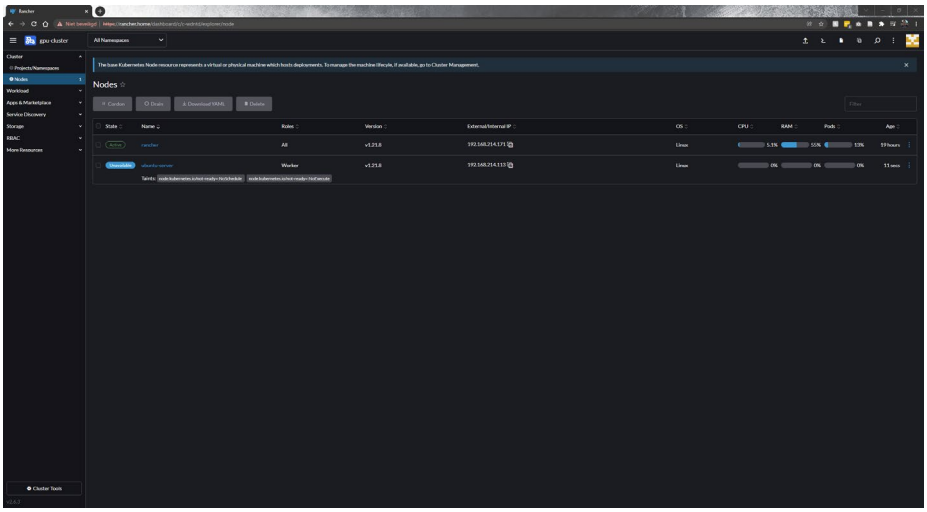
Then choose you cluster and there you will see the command you need. Just change the roles for the computer. Because this computer wants to join the cluster as a worker node the only function we need is Worker. Make sure this is the only selected option. The command will change accordingly.



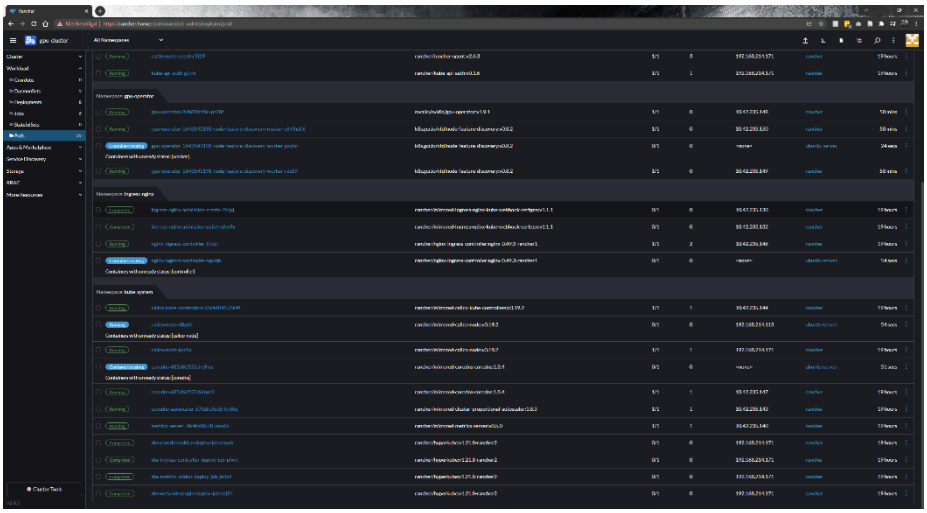
Copy the command and execute it on the computer that wants to join the cluster. Docker will download all the images needed and the computer will start joining the cluster. This will take a few minutes.



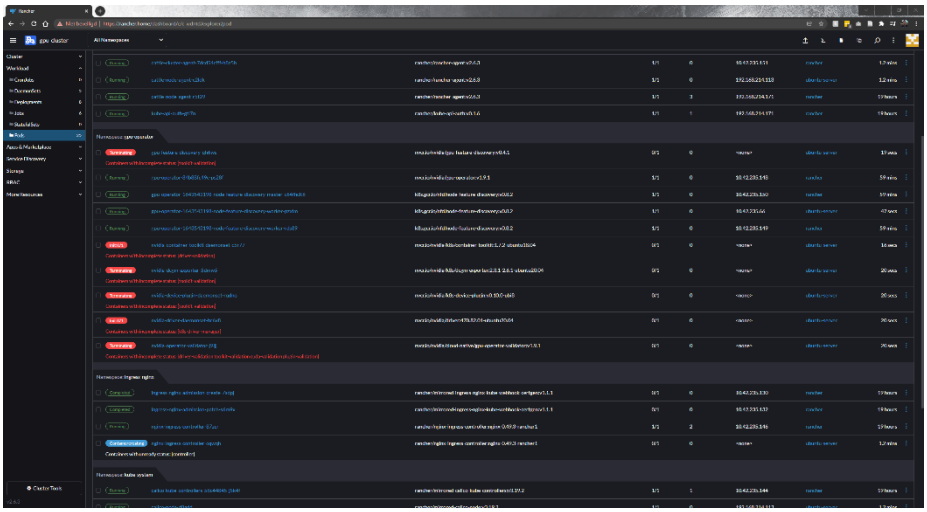
Go back to the Rancher UI and go to your cluster. Under the tab Nodes you will see the new computer appear after some time. If you don’t see it yet just have some patience. The process is running in the background and eventually the pc will appear.



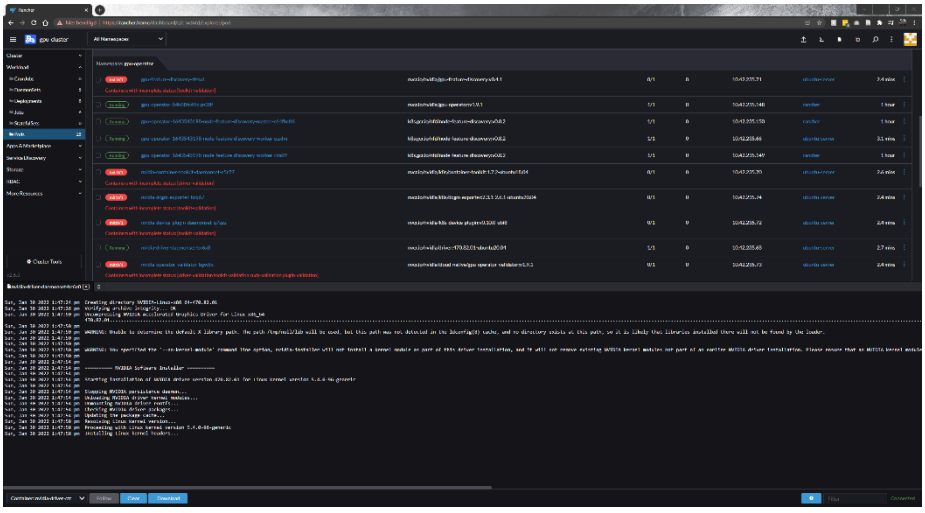
The new node will be configured automatically and become active. Once the node is active head over to Workload and then Pods. Remove the filter at the top of the page and you will see that pods are being created on the new node.



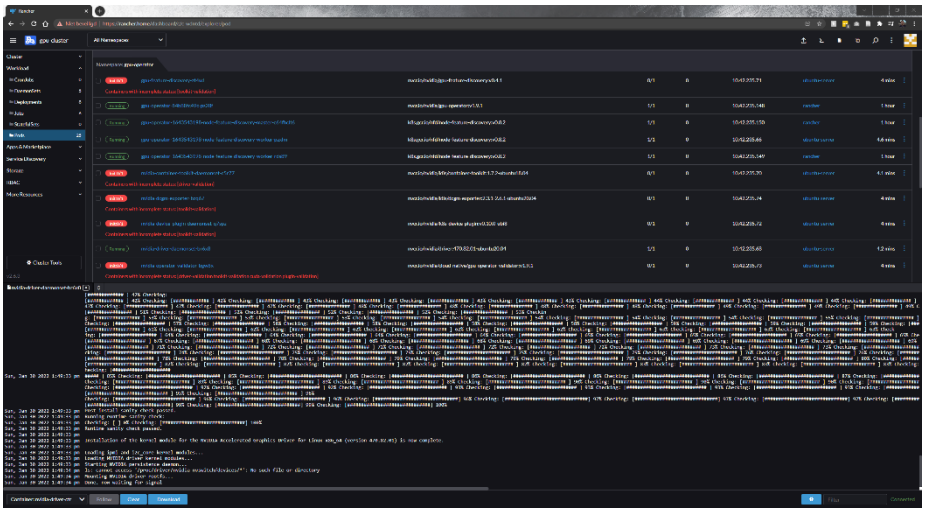
The first pods that are created are the backend pods and one pod from Nvidia GPU-operator. On all the nodes runs a pod like this. This pod called node-feature-discovery-worker checks the node for the available hardware. If an Nvidia GPU is found, the GPU-operator will start creating all the needed pods to use that GPU.



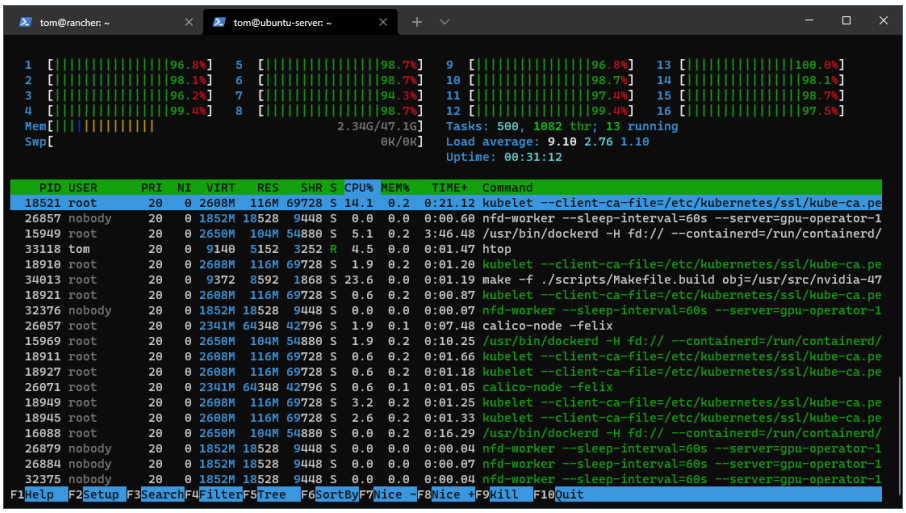
When the pods are created most of them will crash and be marked red. This is completely normal. All the pods are dependent on each other. After some time, all the pods will become green. This process can take up to 10 - 15 minutes depending on the hardware. The first pod that needs to be created is the nvidia-driver-daemonset. This pod installs a compatible driver for the GPU and checks regularly for new drivers and automatically installs them. When this pod is running you can view the logs to see its progress. Click on the 3 dots on the right side and click view logs.



Here you can see what is happening inside the pod. You will see warnings and errors, but you may ignore them. The driver is installed when you see Done, now waiting for signal.



You can also check your system load with htop in the terminal. When the driver is installing you will see that all the CPU-resources are in use. The driver installation is a pretty heavy task. This also takes the most time to complete.



When the driver is installed, the other pods will start changing state and more of them will get the status running or complete. Eventually all the pods should get a green status saying Completed or Running. The GPU can now be used in the cluster.

Repeat these steps starting from Prepare worker node for each node you want to add to the cluster.

Test Nvidia GPU-operator by deploying test pod

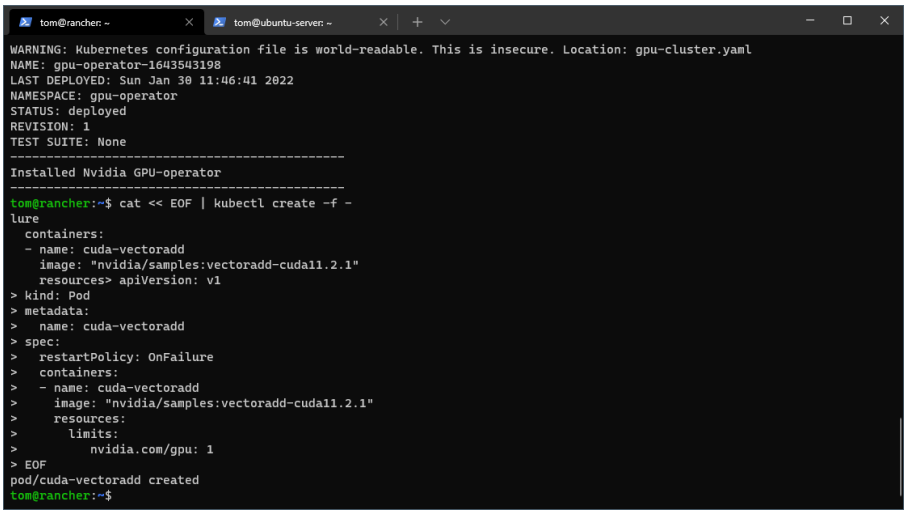
The very last step is deploying 2 test pods. The first one is a simple task that adds 2 vectors together. The 2nd pod generates a GPU load. Both these pods are deployed with kubectl.

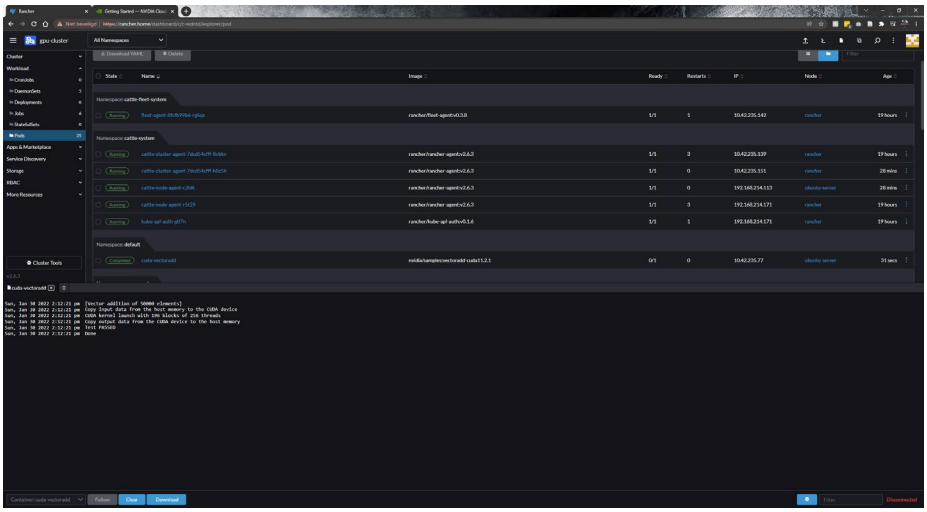
Execute the following command on the machine where you configured kubectl:

```
cat << EOF | kubectl create -f -
apiVersion: v1
kind: Pod
metadata:
```

```
  name: cuda-vectoradd
spec:
  restartPolicy: OnFailure
  containers:
  - name: cuda-vectoradd
    image: "nvidia/samples:vectoradd-cuda11.2.1"
    resources:
      limits:
         nvidia.com/gpu: 1
EOF
```

This command creates a manifest file and applies it to the cluster. This example should run fairly quickly. Once it is completed you can check the logs. The pod is created in the default namespace.



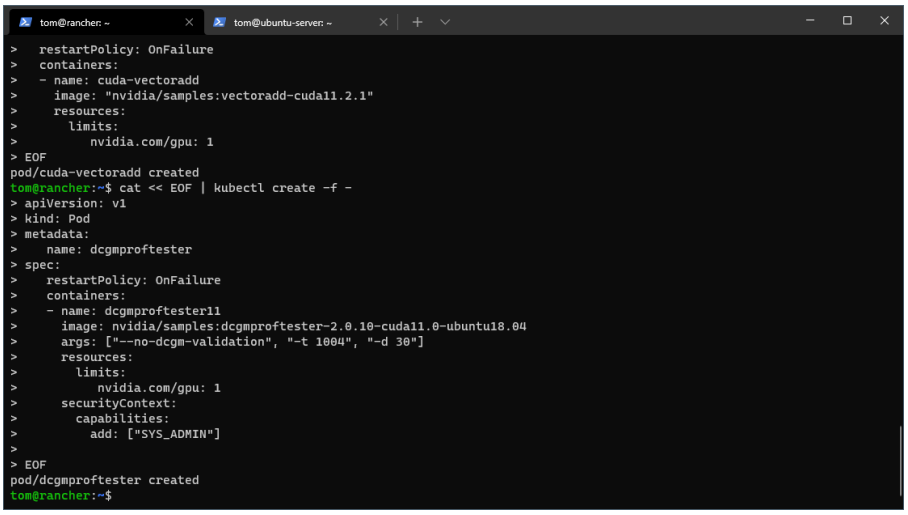


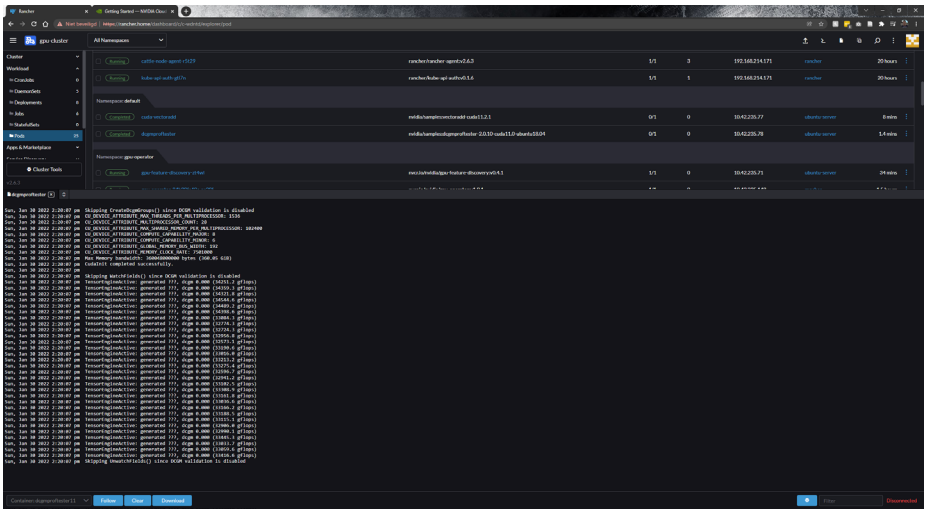
The second pod can be deployed by running the following command on the node configured with kubectl:

```
cat << EOF | kubectl create -f -
apiVersion: v1
kind: Pod
metadata:
  name: dcgmproftester
spec:
```

```
  restartPolicy: OnFailure
  containers:
  - name: dcgmproftester11
    image: nvidia/samples:dcgmproftester-2.0.10-cuda11.0-ubuntu18.04
    args: ["--no-dcgm-validation", "-t 1004", "-d 30"]
    resources:
      limits:
        nvidia.com/gpu: 1
    securityContext:
      capabilities:
        add: ["SYS_ADMIN"]
EOF
```

This will create a pod in the default namespace that generates a CUDA load. While the pod is running you can check the logs to see the progress.





Pods will automatically be scheduled on the nodes by the master. The master node contains a control plane that contains a schedular that automatically deploys the pods on the nodes.

You now have a cluster with usable GPUs. If you encounter any problems refer to the known problems and if you don't find an answer here, there are a lot of communities willing to help.

Known issues

RKE2

If you want to use RKE2 with GPUs, here is what I tried.

- Install containerd, this can be installed with the installDocker.sh script
- In the file /etc/containerd/config.toml, the following line needs to be commented out

```
Disabled_plugins = [“cri”]
```

- In the same file the following content must be added at the end of the file

```
[plugins.cri.cni]
    bin_dir = "/var/lib/rancher/rke2/bin"
    conf_dir = "/etc/cni/net.d"
```

- Restart containerd with the following command:

```
sudo systemctl restart containerd.service
```

- Create the directory /etc/rancher/rke2

```
sudo mkdir /etc/rancher/rke2
```

- Create following file:

```
sudo nano /etc/rancher/rke2/config.yaml
```

With the content:

```
container-runtime-endpoint: unix:///run/containerd/containerd.sock
```

- Join the cluster

This is how I got the Nvidia GPU-operator working for a few days. While I was testing the cluster something changed and broke this configuration. I hope that Rancher will integrate GPU-operator support for the built-in container runtime for RKE2.

Canal routing issue

I made a GitHub issue about this one, you can find it here.

Secure boot

Make sure secure boot is disabled in the BIOS/UEFI settings of the computer. When secure boot is enabled the GPU-operator will not be able to install the driver for the GPU. If you check the logs of the pod, you will see that it says that secure boot is enabled. To disable secure boot, consult the manual of your motherboard. The UEFI settings can be accessed with the following command in Ubuntu or by pressing the right key while the computer boots. (F2, DEL, ESC, this depends on manufacturer).

```
sudo systemctl reboot --firmware-setup
```

Sources

[1]

27 Mar 2020 David BothFeed 407up 5 comments, “An introduction to swap space on Linux systems,” *Opensource.com*. https://opensource.com/article/18/9/swap-space-linux-systems (accessed Jan. 30, 2022).

[2]

“Announcing containerd Support for the NVIDIA GPU Operator,” *NVIDIA Developer Blog*, Feb. 02, 2021. https://developer.nvidia.com/blog/announcing-containerd-support-for-the-nvidia-gpu-operator/ (accessed Jan. 17, 2022).

[3]

“Canal/Flannel CNI not creating necessary routes · Issue #36262 · rancher/rancher,” *GitHub*. https://github.com/rancher/rancher/issues/36262 (accessed Jan. 25, 2022).

[4]

“ClusterIP services not accessible when using flannel CNI from host machines in Kubernetes · Issue #1243 · flannel-io/flannel,” *GitHub*. https://github.com/flannel-io/flannel/issues/1243 (accessed Jan. 25, 2022).

[5]

“Docker and Kubernetes: The Big Picture | Pluralsight.” https://app.pluralsight.com/library/courses/docker-kubernetes-big-picture/table-of-contents (accessed Jan. 12, 2022).

[6]

“Error: Kubernetes cluster unreachable with helm 3.0 · Issue #1126 · k3s-io/k3s,” *GitHub*. https://github.com/k3s-io/k3s/issues/1126 (accessed Jan. 17, 2022).

[7]

M. Abrams, “Get Up and Running with NVIDIA GPUs in Rancher Kubernetes Clusters,” *SUSE Communities*. https://www.suse.com/c/rancher_blog/get-up-and-running-with-nvidia-gpus-in-rancher-kubernetes-clusters/ (accessed Jan. 14, 2022).

[8]

“Getting Started — NVIDIA Cloud Native Technologies documentation.” https://docs.nvidia.com/datacenter/cloud-native/gpu-operator/getting-started.html (accessed Jan. 13, 2022).

[9]

“Getting Started with Kubernetes | Pluralsight.” https://app.pluralsight.com/library/courses/kubernetes-getting-started/table-of-contents (accessed Nov. 10, 2021).

[10]

J. Mutai, “How To Install Kubernetes Cluster with Rancher RKE | ComputingForGeeks,” Jan. 28, 2020. https://computingforgeeks.com/install-kubernetes-production-cluster-using-rancher-rke/ (accessed Jan. 13, 2022).

[11]

“Install a Nvidia GPU Operator on RKE2 Kubernetes Cluster,” *The New Stack*, Nov. 09, 2021. https://thenewstack.io/install-a-nvidia-gpu-operator-on-rke2-kubernetes-cluster/ (accessed Jan. 13, 2022).

[12]

“Install Calico networking and network policy for on-premises deployments.” https://docs.projectcalico.org/getting-started/kubernetes/self-managed-onprem/onpremises (accessed Jan. 14, 2022).

[13]

“Install Docker Engine on Ubuntu,” *Docker Documentation*, Jan. 12, 2022. https://docs.docker.com/engine/install/ubuntu/ (accessed Jan. 13, 2022).

[14]

“Installing kubeadm,” *Kubernetes*. https://kubernetes.io/docs/setup/production-environment/tools/kubeadm/install-kubeadm/ (accessed Jan. 13, 2022).

[15]

262588213843476, “Instructions for running RKE2/K3s with an external CRI/container-runtime,” *Gist*. https://gist.github.com/bgulla/3b725f0eea54fdd49f4d7066e16b1d89 (accessed Jan. 19, 2022).

[16]

“K3s vs. K8s: When to Use Lightweight Kubernetes.” https://www.containiq.com/post/k3s-vs-k8s (accessed Jan. 12, 2022).

[17]

H. Roy, “k8s vs k3s: The Comprehensive Difference | p3r,” Jul. 22, 2021. https://www.p3r.one/k8s-vs-k3s/ (accessed Nov. 09, 2021).

[18]

“Kubernetes Documentation,” *Kubernetes*. https://kubernetes.io/docs/home/ (accessed Nov. 09, 2021).

[19]

“Manual Quick Start,” *Rancher Labs*. https://rancher.com/docs/rancher/v2.6/en/quick-start-guide/deployment/quickstart-manual-setup/ (accessed Jan. 14, 2022).

[20]

“MicroK8s - Zero-ops Kubernetes for developers, edge and IoT | MicroK8s,” *microk8s.io*. http://microk8s.io (accessed Jan. 10, 2022).

[21]

NVIDIA GPU Operator. NVIDIA Corporation, 2022. Accessed: Jan. 13, 2022. [Online]. Available: https://github.com/NVIDIA/gpu-operator

[22]

GitAnswer, "Nvidia gpu support with containerd - rke2 | GitAnswer," *https://gitanswer.com*. https://gitanswer.com/nvidia-gpu-support-with-containerd-843478232 (accessed Jan. 18, 2022).

[23]

"Nvidia gpu support with containerd · Issue #822 · rancher/rke2," *GitHub*. https://github.com/rancher/rke2/issues/822 (accessed Jan. 17, 2022).

[24]

"nvidia-cuda-validator pods crashlooping in OKD4.7 · Issue #259 · NVIDIA/gpu-operator," *GitHub*. https://github.com/NVIDIA/gpu-operator/issues/259 (accessed Jan. 17, 2022).

[25]

"nvidia-cuda-validator pods crashlooping in OpenShift 4.6 · Issue #253 · NVIDIA/gpu-operator," *GitHub*. https://github.com/NVIDIA/gpu-operator/issues/253 (accessed Jan. 17, 2022).

[26]

"Overview — NVIDIA Cloud Native Technologies documentation." https://docs.nvidia.com/datacenter/cloud-native/container-toolkit/overview.html (accessed Jan. 10, 2022).

[27]

"Quickstart for Calico on Kubernetes." https://projectcalico.docs.tigera.io/getting-started/kubernetes/quickstart (accessed Jan. 13, 2022).

[28]

R. Admin, "Rancher vs. RKE: What Is the Difference?," *SUSE Communities*. https://www.suse.com/c/rancher_blog/rancher-vs-rke-what-is-the-difference/ (accessed Jan. 13, 2022).

[29]

"Requirements," *Rancher Labs*. https://rancher.com/docs/rke/latest/en/os/ (accessed Jan. 13, 2022).

[30]

"toolkit validation crashloopbackoff - CUDA driver version is insufficient for CUDA runtime version · Issue #176 · NVIDIA/gpu-operator," *GitHub*. https://github.com/NVIDIA/gpu-operator/issues/176 (accessed Jan. 17, 2022).