GP - GPU Enabled Containers Documentation

Relevant Background Information
- a. OS Choice
- b. AnyDesk installation
- c. Driver choice

Installation
- a. Installation of OS
- b. Documentation of Console Commands
  - i. Docker Installation
  - ii. Flannel Installation
    1. Explanation why we chose Flannel over Calico

Console Commands Clustering
- a. Cluster Setup on Master Node

Containerisation
- a. Restarting the worker node to apply driver installation

Relevant Background Information

a. OS Choice
   i. Depending on the choice of OS the outcome will defer, we selected RHEL 8.2 however we recommend to use ubuntu or container-optimised OS. This is due to RHEL 8> dropping support for docker and difficulties with driver initialisation in kubernetes clusters.

b. Driver Choice
   i. The installation for drivers will depend on what type of GPU which the worker node(s) would be utilising. Within our implementation we only had one worker node which was installed with an NVIDIA Tesla T4. If you are using a NVIDIA GPU much like we did, NVIDIA provides a search function to identify the appropriate driver for your GPU.
   ii. Please note that the most recent release of a NVIDIA driver may not have the same level of quality and stability than older supported driver versions. Within our implementation we originally utilised the latest driver version 460.73.01 but it was unstable due to a kernel mismatch. It was found that the older driver version 440.118.02 provided a higher level of quality and stability over the newer version.

OS Installation

To install RHEL 8.2, have your UEFI boot key on hand and connected to the system you wish to install the OS on. This process must be done across both the master (control-plane) and worker node(s).

1. Boot the system and open the BIOS, in the case of the system we used, we had to hit the DEL key to get to the BIOS.

2. Select the boot drive as the UEFI boot key and move it to the top of the boot order and restart or alternatively you can override the boot drive selection manually to force it to boot from the UEFI boot key.
3. The system will prepare requirements in the background and will eventually get the “Welcome to Red Hat Enterprise Linux 8.2” menu, here you select your chosen language and dialect region. In our case we had English and English (Australia).
4. Installation Summary will populate to choose the drive to target the installation on. Select “Installation Destination”, choose the drive and then “Done”, a window will popup asking for reclamation of space, click “Reclaim Space” - this will format the drive. A window named Reclaim Disk Space will popup, select “Delete All” and then “Reclaim Space”
5. Next select “Network & Host Name” and toggle on the Ethernet selection or configure a wireless connection. It will automatically configure IP and host names.
6. Now click on “Connect to Red Hat” this step will need to be completed by a Hyperscalers employee.
7. Now click on “Software Selection” this is important, failure to configure the software selection will result in RHEL defaulting to a CLI based server. Select the Base Environment to be “Workstation” and then “Done”
8. Click on “Begin Installation” and set Root Password and User Creation by following the prompts. Make sure to check the box of “Make this user administrator” for the user profile you create. One profile was Master on the Master Node and the other was Worker on the Worker Node for us.
9. Wait for the OS to install and follow prompts, the OS is now installed, repeat these steps for all server nodes.

AnyDesk Install

1. AnyDesk is used for remote access, first download AnyDesk from their website
2. Locate the file in Downloads and double click to install, follow prompts to complete.
3. In case the current version does not install smoothly, use an older version found here, and follow prompts to install.

Driver Install

1. This needs to occur on nodes with GPU, for NVIDIA Tesla T4 we predominantly followed NVIDIA Data Center Documentation found here
2. Before installing the drivers, ensure the driver is compatible with the OS. For RHEL 8, NVIDIA Driver Stream 440 is stable.
3. Open up a terminal session to install the following: GCC, Make and Linux Kernel Headers. For GCC use: `sudo dnf install GCC` check with `gcc --version` to check installation version. For Make use `sudo dnf install make` check with `make --version` to check and for the Kernel Headers use `sudo dnf install elfutils-libelf-devel.`
4. Before drivers can be installed, the current nouveau driver must be uninstalled. To do this in the terminal session, `sudo bash -c "echo blacklist nouveau > /etc/modprobe.d/blacklist-nvidia-nouveau.conf"` and `sudo bash -c "echo options nouveau modeset=0 >>`

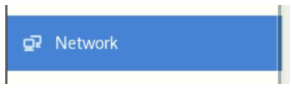
`/etc/modprobe.d/blacklist-nvidia-nouveau.conf"` confirm the modprobe config file by: `cat /etc/modprobe.d/blacklist-nvidia-nouveau.conf` it will give you the echo inputs.

5. Rebuild the kernel as this is kernel level change by running: `dracut --force` or `dracut -f -v` they both do the same. Then reboot the node.
6. Check that the Nouveau driver is no longer in use by `sudo lshw -numeric -C display` the configuration section should no longer have a driver.
7. Download the desired driver from here, we used this driver from here, locate where the driver is downloaded to and make note, we had it in Downloads.
8. To minimise any conflict during installation we change the environment to minimal CLI by `sudo init 3`, login to the system by targeting your user profile and password. Then change directory to where the driver is located by `cd /home/worker/Downloads.`
9. You can now install the driver using the runtime file by `sudo sh NVIDIA-Linux-x86_64-440.118.02.run`
10. Follow the prompts on screen during the installation. Reboot the system after the installation is complete. If upon the restart the OS is showing as blank, the driver is not compatible for the version on RHEL, uninstall the driver, install a different version. For any further issues refer to the readme for troubleshooting
11. Confirm install by running the command `nvidia-smi,` giving you:

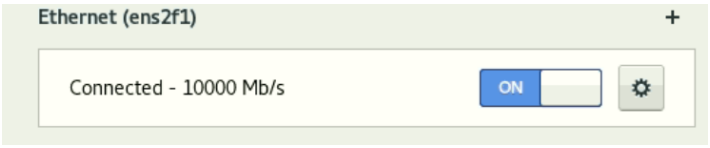
```
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 440.118.02    Driver Version: 440.118.02    CUDA Version: 10.2   |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|===============================+======================+======================|
|   0  Tesla T4            Off  | 00000000:01:00.0 Off |                    0 |
| N/A   38C    P0    20W /  70W |      0MiB / 15109MiB |      0%      Default |
+-------------------------------+----------------------+----------------------+

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

Setting a Static IP

1. Open the RHEL ‘Settings’ application
2. Select network 

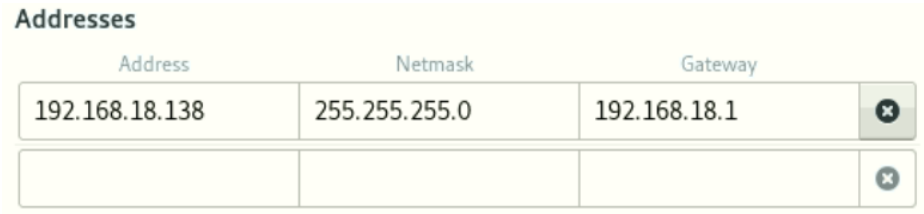
3. Ethernet, click the cog on the right.



4. Go to IPv4 

5. Select IPv4 Method as Manual

6. Open a terminal session and use the `ifconfig` command to retrieve current network details.
7. Back in the settings window, enter in desired static IP address we used: 192.168.18.138 for master and 192.168.18.139 for worker.



8. Set DNS as desired: 

9. You now have a static IP address required for kubernetes clustering.

Kubernetes Installation Commands on All Nodes

Please note that the following commands are to be run on both the control plane which we refer to as the Master Node and worker node(s).

1. Rename Node

```
[master@master-node home]$ sudo hostnamectl set-hostname master-node
[worker@worker-node home]$ sudo hostnamectl set-hostname worker-node
```

If both nodes are freshly installed servers, they will have the same host name which defaults to localhost@localdomain. If this is the case, as clustering requires both nodes to have different hostnames, it will not be possible to cluster the nodes. To remedy this, it is required to set a particular host name through the above command before attempting to cluster. We have two entries of this command and the hostname string is different.

2. Disable Selinux

```
[root@master-node home]# setenforce 0
```

This disables Selinux, as this is required to allow containers to access the host filesystem, which is needed by pod networks and other services. Setting setenforce to 0 effectively sets SELinux to permissive, which effectively disables SELinux until the next reboot.

3. Configure ports

Kubernetes makes use of various ports for communication and access and these ports shown below need to be accessible to Kubernetes and not limited by the firewall.

Protocol	Direction	Port Range	Purpose	Used By
TCP	Inbound	6443*	Kubernetes API server	All
TCP	Inbound	2379-2380	etcd server client API	kube-apiserver, etcd
TCP	Inbound	10250	Kubelet API	Self, Control plane
TCP	Inbound	10251	kube-scheduler	Self
TCP	Inbound	10252	kube-controller-manager	Self

```
[root@master-node home]# firewall-cmd --permanent --add-port=6443/tcp
[root@master-node home]# firewall-cmd --permanent --add-port=2379-2380/tcp
[root@master-node home]# firewall-cmd --permanent --add-port=10250/tcp
[root@master-node home]# firewall-cmd --permanent --add-port=10251/tcp
[root@master-node home]# firewall-cmd --permanent --add-port=10252/tcp
[root@master-node home]# firewall-cmd --permanent --add-port=10255/tcp
[root@master-node home]# firewall-cmd --reload
[root@master-node home]# modprobe br_netfilter
[root@master-node home]# echo '1' > /proc/sys/net/bridge/bridge-nf-call-iptables
```

Alternatively you can also outright disable the firewall by

```
[root@master-node home]# sudo systemctl disable firewalld
[root@master-node home]# sudo systemctl stop firewalld
```

4. Install Docker-CE

Add the docker repository as it is no longer in the default package list:

```
[root@master-node home]# dnf config-manager
--add-repo=https://download.docker.com/linux/rhel/docker-ce.repo
```

Then install docker

```
[root@master-node home]# dnf install docker
```

5. Enable, Start and Check docker running

```
[root@master-node home]# systemctl enable docker
[root@master-node home]# systemctl start docker
[root@master-node home]# systemctl status docker
```

6. Disable swapping

```
[root@master-node home]# sudo swapoff -a
```

Disable the swapping because to install Kubernetes we need to disable memory swapping

7. Enable usage of iptables

```
[root@master-node home]# bash -c 'echo "net.bridge.bridge-nf-call-ip6tables =
1" > /etc/sysctl.d/k8s.conf'
[root@master-node home]# sysctl --system
```

Enables the usage of iptables which will prevent the routing errors happening.

8. Install Kubernetes (Kubeadm, Kubelet & Kubectl)

First add the kubernetes repo to yum repos

```
[root@master-node home]# vi /etc/yum.repos.d/kubernetes.repo
```

Add these details:

```
[kubernetes]
name=Kubernetes
baseurl=https://packages.cloud.google.com/yum/repos/kubernetes-el7-x86_64
enabled=1
gpgcheck=1
repo_gpgcheck=1
gpgkey=https://packages.cloud.google.com/yum/doc/yum-key.gpg
       https:&#47;&#47;packages.cloud.google.com/yum/doc/rpm-package-key.gpg
```

Install:

```
[root@master-node home]# yum install kubeadm kubelet kubectl
```

9. Enable and Start Kubelet

```
[root@master-node home]# systemctl enable kubelet
[root@master-node home]# systemctl start kubelet
```

Clustering Commands on Master Node

Please note that the following commands are to be run on the control plane which we refer to as the Master Node.

1. Initialise the Cluster

```
[root@master-node home]# kubeadm init
--apiserver-advertise-address=192.168.18.138 --pod-network-cidr=10.244.0.0/16
```

After the required preflight checks, the generation of the certificates and keys, and other initialization steps have successfully run, you receive the following output:

```
...
```

```
Your Kubernetes control-plane has initialized successfully!

...

Then you can join any number of worker nodes by running the following on each
as root:

kubeadm join 192.168.18.138:6443 --token cdrzf8.e0hss3ji1b1e3o5y \
      --discovery-token-ca-cert-hash
sha256:8d7f5a00806803d4b512cb5e541b57080a56c87f183580b4bf175224d2d4d36c
```

It is important within this step to specify both the API server advertisement address and the pod network CIDR. If you do not specify during the initialisation of the Kubernetes control-plane node then errors and issues will occur further down the line.

The flag '`--apiserver-advertise-address=192.168.18.138`' will specify the IP address the API Server will advertise it's listening on. For our case the IP address is the static IP address which the server had been provided. If not specified it will use the default network interface. This is an issue if attempting to implement additional features such as the Kubernetes Web UI (Dashboard) which will be unreachable.

The flag '`--pod-network-cidr=10.244.0.0/16`' is a required flag if implementing Flannel as the CIDR of choice. If using Flannel the address in the flag needed to be the same as the hard coded address within the `kube-flannel.yaml`. If this flag is not specified, any attempts to run any pods on the cluster will fail.

If you had not specified the above flags then you will be required to perform a `kubeadm reset` to then reinitialise the Kubernetes control-plane node. The output of the `kubeadm init` will also contain the generated token which will be required to run on all the worker nodes to join them to the cluster. *If you are following these steps please note that your token will be different.*

2. The Three-Line B*tch

```
[root@master-node home]# su master
[master@master-node home]$ mkdir -p $HOME/.kube
[master@master-node home]$ sudo cp -i /etc/kubernetes/admin.conf
HOME/.kube/config
[master@master-node home]$ sudo chown $(id -u):$(id -g) $HOME/.kube/config
```

In the above command we are first switching from root to user to then make the directory `$HOME/.kube` which has the `admin.conf` file pasted into the newly created directory. Then the ownership over the directory is changed from the user (`-u`) to group (`-g`).

3. Viewing Nodes & Pods

```
[master@master-node home]$ kubectl get nodes
NAME           STATUS   ROLES                  AGE     VERSION
master-node    Ready    control-plane,master   3m41s   v1.21.0
```

```
[root@master-node home]# kubectl get pods -o wide --all-namespaces
NAMESPACE     NAME                                   READY   STATUS    RESTARTS   AGE     IP              NODE          NOMINATED NODE   READINESS GATES
kube-system   coredns-558bd4d5db-4zxhl               1/1     Running   0          2m47s   10.244.0.2      master-node   <none>           <none>
kube-system   coredns-558bd4d5db-jhblj               1/1     Running   0          2m47s   10.244.0.3      master-node   <none>           <none>
kube-system   etcd-master-node                       1/1     Running   0          3m2s    192.168.18.138  master-node   <none>           <none>
kube-system   kube-apiserver-master-node             1/1     Running   0          3m2s    192.168.18.138  master-node   <none>           <none>
kube-system   kube-controller-manager-master-node    1/1     Running   0          3m1s    192.168.18.138  master-node   <none>           <none>
kube-system   kube-proxy-8dhfj                       1/1     Running   0          2m47s   192.168.18.138  master-node   <none>           <none>
kube-system   kube-scheduler-master-node             1/1     Running   0          3m1s    192.168.18.138  master-node   <none>           <none>
```

Within the first command '`kubectl get nodes`' we are listing all of the nodes within the cluster and each node's status. At this point we have not joined the Worker-Node to the cluster so only the Master-Node should be listed. The second command '`kubectl get pods -o wide --all-namespaces`' will provide information of all pods currently running. Please note that if you were following this guide and are getting the following error '`Unable to connect to the server: x509: certificate signed by unknown authority (possibly because of "crypto/rsa: verification error" while trying to verify candidate authority certificate "kubernetes")`', it is because you are in `Root` rather than `Master`.

4. Apply Flannel CNI

```
[master@master-node home]$ kubectl apply -f
https://raw.githubusercontent.com/coreos/flannel/2140ac876ef134e0ed5af15c65e414cf26827915/Documentation/kube-flannel.yml
```

5. Apply Kubernetes Dashboard

```
[master@master-node home]$ kubectl apply -f
https://raw.githubusercontent.com/kubernetes/dashboard/v2.0.0/aio/deploy/recommended.yaml
```

We can check if the features had been correctly applied into the cluster by running the below command.

```
[master@master-node home]$ kubectl get pods --all-namespaces
NAMESPACE              NAME                                         READY   STATUS              RESTARTS   AGE
kube-system            coredns-558bd4d5db-4zxhl                     1/1     Running             0          5m50s
kube-system            coredns-558bd4d5db-jhblj                     1/1     Running             0          5m50s
kube-system            etcd-master-node                             1/1     Running             0          6m5s
kube-system            kube-apiserver-master-node                   1/1     Running             0          6m5s
kube-system            kube-controller-manager-master-node          1/1     Running             0          6m4s
kube-system            kube-flannel-ds-amd64-8xxcm                  1/1     Running             0          55s
kube-system            kube-proxy-8dhfj                             1/1     Running             0          5m50s
kube-system            kube-scheduler-master-node                   1/1     Running             0          6m4s
kubernetes-dashboard   dashboard-metrics-scraper-5594697f48-x6qf9   0/1     ContainerCreating   0          15s
kubernetes-dashboard   kubernetes-dashboard-57c9bfc8c8-qmzdt        0/1     ContainerCreating   0          15s
```

If the pods are in the status of '`ContainerCreating`' instead of '`Running`' it is because they were checked before being properly initialised. If the command is run again, the pods will most likely be in the '`Running`' state. To troubleshoot any pods which are not in the desired status, run the '`kubectl logs <Pod Name> <Container Name>`' command to investigate any errors and their causes.

```
[master@master-node home]$ kubectl get pods --all-namespaces
NAMESPACE              NAME                                         READY   STATUS    RESTARTS   AGE
kube-system            coredns-558bd4d5db-4zxhl                     1/1     Running   0          6m12s
kube-system            coredns-558bd4d5db-jhblj                     1/1     Running   0          6m12s
kube-system            etcd-master-node                             1/1     Running   0          6m27s
kube-system            kube-apiserver-master-node                   1/1     Running   0          6m27s
kube-system            kube-controller-manager-master-node          1/1     Running   0          6m26s
kube-system            kube-flannel-ds-amd64-8xxcm                  1/1     Running   0          77s
kube-system            kube-proxy-8dhfj                             1/1     Running   0          6m12s
kube-system            kube-scheduler-master-node                   1/1     Running   0          6m26s
kubernetes-dashboard   dashboard-metrics-scraper-5594697f48-x6qf9   1/1     Running   0          37s
kubernetes-dashboard   kubernetes-dashboard-57c9bfc8c8-qmzdt        1/1     Running   0          37s
```

Clustering Commands On Worker Node

Please note that the following commands are to be run on the Worker Node.

1. Join Worker to Cluster

```
[root@localhost home]# su worker
[worker@worker-node home]$ sudo kubeadm join 192.168.18.138:6443 --token
cdrzf8.e0hss3ji1b1e3o5y --discovery-token-ca-cert-hash
sha256:8d7f5a00806803d4b512cb5e541b57080a56c87f183580b4bf175224d2d4d36c
```

The join token was specified in the output of the `kubeadm init` which was shown in the previous section **Clustering Commands on Master Node,** 1. Initialise the Cluster.

Expected Output:

```
[preflight] Running pre-flight checks
[preflight] Reading configuration from the cluster...
[preflight] FYI: You can look at this config file with 'kubectl -n
kube-system get cm kubeadm-config -o yaml'
[kubelet-start] Writing kubelet configuration to file
"/var/lib/kubelet/config.yaml"
[kubelet-start] Writing kubelet environment file with flags to file
"/var/lib/kubelet/kubeadm-flags.env"
[kubelet-start] Starting the kubelet
[kubelet-start] Waiting for the kubelet to perform the TLS Bootstrap...

This node has joined the cluster:
* Certificate signing request was sent to apiserver and a response was
received.
```

```
* The Kubelet was informed of the new secure connection details.

Run 'kubectl get nodes' on the control-plane to see this node join the
cluster.
```

Please note that the following command is to be run on the control plane which we refer to as the Master Node.

2. Confirm that Worker has Joined the Cluster

```
[master@master-node home]$ kubectl get nodes --all-namespaces
NAME          STATUS   ROLES                  AGE   VERSION
master-node   Ready    control-plane,master   14m   v1.21.0
worker-node   Ready    <none>                 66s   v1.21.0
```

As can be seen from the output above, the worker node has successfully joined the cluster and currently has the 'Ready' status.

Viewing the Kubernetes Dashboard Website

Please note that the following command is to be run on the control plane which we refer to as the Master Node.

1. Create ServiceAccount Dashboard

```
[root@master-node home]# su master
[master@master-node home]$ kubectl create serviceaccount dashboard -n default
serviceaccount/dashboard created
```

2. Get ServiceAccount Token

```
[master@master-node home]$ kubectl get secret $(kubectl get serviceaccount
dashboard -o jsonpath="{.secrets[0].name}") -o jsonpath="{.data.token}" |
base64 --decode
```


```
eyJhbGciOiJSUzI1NiIsImtpZCI6IncwREZBbzNOMU1ldHJkT3hQUUdsV0RaVjZLWmMwcHNEeHlCeXowM2g5QTgifQ.eyJpc3MiOiJrdWJlcm5ldGVzL3NlcnZpY2VhY2NvdW50Iiwia3
ViZXJuZXRlcy5pby9zZXJ2aWNlYWNjb3VudC9uYW1lc3BhY2UiOiJkZWZhdWx0Iiwia3ViZXJuZXRlcy5pby9zZXJ2aWNlYWNjb3VudC9zZWNyZXQubmFtZSI6ImRhc2hib2FyZC10b2t
lbi1xN3h4YyIsImt1YmVybmV0ZXMuaW8vc2VydmljZWFjY291bnQvc2VydmljZS1hY2NvdW50Lm5hbWUiOiJkYXNoYm9hcmQiLCJrdWJlcm5ldGVzLmlvL3NlcnZpY2VhY2NvdW50L3Nl
cnZpY2UtYWNjb3VudC51aWQiOiJjOTFlNmMzMy0wOTljLTRjOWEtOWIyMi04ZDkzZGQ3MmFkOGUiLCJzdWIiOiJzeXN0ZW06c2VydmljZWFjY291bnQ6ZGVmYXVsdDpkYXNoYm9hcmQif
Q.QdgA1-2QtIt4mcfcrGMPKa4CtsAX2t5BUKkWxjobHkGEUZIQHT1vvi5s_lHvPOSTOCZ_Kc6ZMJVX7rigDClrlsnPDRuH6v6zW_jobfWUGjaObkA7waO7mizF2ba3u13dabtjPZtgfx-
lwhpYR20n30-L1R_OYHTJtvjE5iv5VCVb74SoS59d1E01p2Z4DOFHPQeuMa1FX7F0yEgC6LTuLzgFE1cmjPjofqHtApMwZDPZ_tdOO1JHeg1uVzYuR_bPCUCPpyOjE0T-I-iNVmIBEieX
N3zw0v-AjBo1HaLvxe3IoeKrM3-zDq_zC0kj9su9lkA9Dvx-kETvwtHc9eY0Tg
```


3. Check Status of Nodes and Pods

```
[master@master-node home]$ kubectl get nodes --all-namespaces
NAME          STATUS   ROLES                  AGE   VERSION
master-node   Ready    control-plane,master   14m   v1.21.0
worker-node   Ready    <none>                 66s   v1.21.0
[master@master-node home]$ kubectl get po --all-namespaces
NAMESPACE     NAME                                  READY   STATUS    RESTARTS   AGE
kube-system   coredns-558bd4d5db-4zxhl              1/1     Running   0          15m
kube-system   coredns-558bd4d5db-jhblj              1/1     Running   0          15m
```

```
kube-system            etcd-master-node                            1/1     Running   0          15m
kube-system            kube-apiserver-master-node                  1/1     Running   0          15m
kube-system            kube-controller-manager-master-node         1/1     Running   0          15m
kube-system            kube-flannel-ds-amd64-8xxcm                 1/1     Running   0          10m
kube-system            kube-flannel-ds-amd64-ffzkv                 1/1     Running   0          2m15s
kube-system            kube-proxy-8dhfj                            1/1     Running   0          15m
kube-system            kube-proxy-8fvzh                            1/1     Running   0          2m15s
kube-system            kube-scheduler-master-node                  1/1     Running   0          15m
kubernetes-dashboard   dashboard-metrics-scraper-5594697f48-x6qf9  1/1     Running   0          9m30s
kubernetes-dashboard   kubernetes-dashboard-57c9bfc8c8-qmzdt       1/1     Running   0          9m30s
```

We are simply checking the status of the nodes and the pods through the above commands which were also used previously to ensure that no issues have arisen since the creation of the ServiceAccount Dashboard.

4. Using Kubectl to Start a Proxy Server

```
[master@master-node home]$ kubectl proxy
Starting to serve on 127.0.0.1:8001
```

With the proxy running the Kubernetes Dashboard can then be viewed in the local browser. Please note that running `kubectl proxy` will relegate the terminal window to this operation and you must open another terminal window if you wish to run any additional commands without closing the proxy. If you do wish to cancel the proxy simply input `CTRL + C` on the keyboard with the terminal selected. Once the proxy is cancelled, the localhost Kubernetes Dashboard will also be cancelled until the proxy is running using the above command again.

5. View Kubernetes Dashboard

Using the URL below the Kubernetes Dashboard can be viewed within the local browser given that the `kubectl proxy` command in the previous step is still running and has not been cancelled.

http://localhost:8001/api/v1/namespaces/kubernetes-dashboard/services/https:kubernetes-dashboard:/proxy/#/pod?namespace=_all

Training Neural Networks

Please note that the following commands are to be run on the Worker Node.

1. Clone TensorFlow Benchmarks

```
[worker@worker-node ~]$ sudo mkdir /data
[worker@worker-node ~]$ sudo chown worker:worker /data
[worker@worker-node data]$ cd /data
[worker@worker-node data]$ git clone
https://github.com/tensorflow/benchmarks.git
```

2. Running TensorFlow on **CPU** and **GPU**

The above commands will run the

```
[worker@worker-node data]$ time sudo docker run --gpus all -it --rm -v
/data:/data nvcr.io/nvidia/tensorflow:19.11-tf1-py3 python3
/data/benchmarks/scripts/tf_cnn_benchmarks/tf_cnn_benchmarks.py
--data_format=NHWC --batch_size=32 --model=resnet50
--variable_update=parameter_server
```

c. Context for Console Commands (add relevant error to relevant command?)
   i. Error in not being able to cluster
      1. Caused by both the worker and master nodes had the same name - both showing up as localhost@localdomain. Nodes cannot have the same name in a Kubernetes cluster.
      2. Fixed by configuring hostname with: hostnamectl set-hostname <String>
      3. In our case we used the hostnames of: master-node and worker-node
   ii. Error Causing the Worker Node to not Run
      1. Need to apply the Pod Network
      2. **Fixed by Abuzar running that one-line command**
      3. **We did not check what was causing it**
   iii. Error Causing Some Pods to not Run
      1. Caused by not specifying the Flannel specific virtual IP address
      2. --pod-network-cidr 10.244.0.0/16 **(Pretty sure this was the flag)**
   iv. Error Causing Pods to be in CrashLoopBackOff
      1. Multiple CNI's were still installed onto the server
      2. Error was caused by old calico installation conflicting with flannel and preventing the pod to be properly instantiated
      3. Fixed by deleting the unnecessary Calico files
   v. Error where it was stuck on ContainerCreating
   vi. Error with Kubernetes Proxy not being Reachable
      1. Caused by not specifying the advertising address during kubectl
      2. --apiserver-advertise-address <Internal IP Address>