1 Cluster Architecture
    1.1 Master Node Configurations
    1.2 Worker Node Configurations
2 Application Process

1 Cluster Architecture

- The kubernetes cluster consists of 2 nodes
    - 1 Master
    - 1 worker node
- I have used Microsoft Azure cloud platform to create virtual machines using the ubuntu 20.02 linux distribution.
- I have used kubeadm to create a kubernetes cluster.

1.1 Master Node Configurations

- As per the official documentation following ports need to be opened on the master node.

Control plane

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	10259	kube-scheduler	Self
TCP	Inbound	10257	kube-controller-manager	Self

- After that i have disabled the swap memory in order for the kubelet to work properly by using the following commands
    - **sudo swapoff -a**
    - **sed -i '/ swap / s/^\(.*\)$/#\1/g' /etc/fstab**
    - **sudo apt update**
- As in order to run pods we need to have a container runtime present inside the cluster, so i have used the following commands to make these changes.
- Please use the following link to get these kernel related configurations
- https://kubernetes.io/docs/setup/production-environment/container-runtimes/

```
cat <<EOF | sudo tee /etc/modules-load.d/k8s.conf
overlay
br_netfilter
EOF

sudo modprobe overlay
sudo modprobe br_netfilter

# sysctl params required by setup, params persist across reboots
cat <<EOF | sudo tee /etc/sysctl.d/k8s.conf
net.bridge.bridge-nf-call-iptables  = 1
net.bridge.bridge-nf-call-ip6tables = 1
net.ipv4.ip_forward                 = 1
EOF

# Apply sysctl params without reboot
sudo sysctl --system
```

- I have used the docker container platform which uses the containerd under the hood for container run time, docker can be installed using the following command
    - **sudo apt install docker.io**
    - **sudo systemctl enable docker.service**
    - **sudo systemctl start docker.service**
    - **sudo systemctl status docker.service**
- After installing the container runtime i have installed the kubeadm, kubectl and kubelet with version 1.25.9 using the following commands
    - **sudo apt install -y kubelet=1.25.9-00 kubeadm=1.25.9-00 kubectl=1.25.9-00**
    - **sudo systemctl enable kubelet.service**
    - **sudo systemctl start kubelet.service**
    - **sudo systemctl status kubelet.service**
- After installing the required dependencies i have pulled the images that are used to run the kubernetes core components such as api server, controller, scheduler and etcd using the following command
    - **sudo kubeadm config images pull**
- Now all of the required dependencies are installed i have initialised the cluster using following command
    - **sudo kubeadm init --kubernetes-version=1.25.9 --pod-network-cidr=192.168.0.0/16 --ignore-preflight-errors=all**
- I have used the kubeadm join token for the worker node from the output of kubeadm init command.
- After all of this i have installed the container network interface (CNI) calico plugin using the following command
    - **kubectl create -f https://raw.githubusercontent.com/projectcalico/calico/v3.25.1/manifests/tigera-operator.yaml**
    - **kubectl create -f https://raw.githubusercontent.com/projectcalico/calico/v3.25.1/manifests/custom-resources.yaml**
    - **watch kubectl get pods -n calico-system**

1.2 Worker Node Configurations

- As per the official documentation following ports need to be opened on the worker node.

Worker node(s)

Protocol	Direction	Port Range	Purpose	Used By
TCP	Inbound	10250	Kubelet API	Self, Control plane
TCP	Inbound	30000-32767	NodePort Services†	All

- After that i have disabled the swap memory in order for the kubelet to work properly by using the following commands
    - **sudo swapoff -a**
    - **sed -i '/ swap / s/^\(.*\)$/#\1/g' /etc/fstab**
    - **sudo apt update**
- As in order to run pods we need to have a container runtime present inside the cluster, so I have used the following commands to make these changes.
- Please use the following link to get these kernel related configurations
- https://kubernetes.io/docs/setup/production-environment/container-runtimes/

```
cat <<EOF | sudo tee /etc/modules-load.d/k8s.conf
overlay
br_netfilter
EOF

sudo modprobe overlay
sudo modprobe br_netfilter

# sysctl params required by setup, params persist across reboots
cat <<EOF | sudo tee /etc/sysctl.d/k8s.conf
net.bridge.bridge-nf-call-iptables  = 1
net.bridge.bridge-nf-call-ip6tables = 1
net.ipv4.ip_forward                 = 1
EOF

# Apply sysctl params without reboot
sudo sysctl --system
```

- I have used the docker container platform which uses the containerd under the hood for container run time, docker can be installed using the following command
    - **sudo apt install docker.io**
    - **sudo systemctl enable docker.service**
    - **sudo systemctl start docker.service**
    - **sudo systemctl status docker.service**

- After installing the container runtime i have installed the kubeadm, kubectl and kubelet with version 1.25.9 using the following commands
    - **sudo apt install -y kubelet=1.25.9-00 kubeadm=1.25.9-00 kubectl=1.25.9-00**
    - **sudo systemctl enable kubelet.service**
    - **sudo systemctl start kubelet.service**
    - **sudo systemctl status kubelet.service**
- I have used the kubeadm join token from the master node so that the worker node can be a part of the cluster.

```
kubeadm join 10.1.0.4:6443 --token 1ybkzq.fqx6a2rjojuw230x \
        --discovery-token-ca-cert-hash sha256:d057acaa8cc683de1d2810861c388236003b2a638c4cddc6ea8012934bf53cd1
```

2 Application Process

- The technology stack for the application consists of the following components
    - **Programming language: Python**
    - **CI CD: Jenkins**
    - **Container Registry: gitlab**
    - **Operating System: Linux**
- So following are the application build and deploy steps
    - Developer will make the changes to the github repository.
    - Jenkins job will be triggered to build a docker image and push it to the gitlab container registry.
    - Update the image tag in the kubernetes deployment file with the updated tag.
    - Apply the updated deployment yaml file.
- You can use the below curl command to test the application
    - **curl 40.112.62.205:30104 -H 'Host: app.com'**