Week3 : 19.06.23 – 23.06.23

Setup and deploy applications on K8s Cluster

- ➔ This week I got familiarized myself with kubernetes. The applications mentioned in task1 are successfully deployed on kubernetes cluster using minikube.
- ➔ I prepared one master node and two worker nodes on the testbeds assigned for this thesis. Details can be found in “Login to servers” section.
- ➔ I created a kubernetes cluster consisting of 1 master node and 2 worker nodes on testbeds.

Install minikube on server

Install Kubectl and minikube on servers by following these two links

https://minikube.sigs.k8s.io/docs/start/

https://kubernetes.io/docs/tasks/tools/install-kubectl-linux/

Before that, we need to install minikube and kubectl on our machine from official documentation.

Login to servers

We changed the servers the hostname and users for the purpose of thesis.

Steps to change hostname and user

pc3: ssh tung@192.168.0.229 → workernode2@192.168.0.229 pwd: 123

pwd: 123

change hostname by;

1- sudo nano /etc/hostname, override hostanme to p4kube

2- sudo nano /etc/cloud/cloud.cfg; change preserve_hostname to true

3- sudo reboot

Add a user workernode2 by;

sudo adduser workernode2

login to workernode2 by;

ssh workernode2@192.168.0.229 pwd 123

login again to tung@192.168.0.229 pwd 123

sudo adduser workernode2 sudo

reboot

you will have the sudo access to workernode2 now.

SSH into pcs;

ssh root@172.31.54.10

pwd: rootsie

ssh master@192.168.0.241

pwd: 123

ssh workernode1@192.168.0.133

pwd: 123

ssh workernode2@192.168.0.229

pwd: 123

ssh workernode3@192.168.0.143

pwd: 123

pc5: ssh workernode4@192.168.0.52

pwd: 123

Setting Kubernetes Cluster consisting of 1 master node and two worker nodes

Setting up all nodes

 # to check ubuntu properties

lsb_release -a

sudo swapoff -a # to disable swapoff

free -m # to see RAM memory allocated to swap, in this case it will be zero. Free tells total used memory used by system and swap, cache

 # remove all docker or k8s previous installations

sudo apt remove < >

#install necessary libraries

sudo apt install apt-transport-https ca-certificates curl software-properties-common -y

 # install docker https://docs.docker.com/engine/install/ubuntu/

→ Install go https://go.dev/doc/install

→ install cri-dockerd https://github.com/Mirantis/cri-dockerd (not required as using containerd as cri)

→ install kubectl, kubeadm, kubelet https://kubernetes.io/docs/setup/production-environment/tools/kubeadm/install-kubeadm/

→ configure cgroup drivers for kubelet and container runtime https://kubernetes.io/docs/tasks/administer-cluster/kubeadm/configure-cgroup-driver/

We donot need to configure cgroups as In v1.22 and later, if the user does not set the `cgroupDriver` field under `KubeletConfiguration`, kubeadm defaults it to `systemd`.

Setting up Master Node

→ make sure swap is off; sudo swapoff -a # to disable swapoff

→ run on master node; sudo kubeadm init

→ got the error :D, container runtime is not running

→ resolved it by; https://github.com/containerd/containerd/issues/8139

→ run; sudo nano /etc/containerd/config.toml

→ change disabled to enabled

→ deploy a pod network by installing network add-on,

kubectl apply -f https://github.com/weaveworks/weave/releases/download/v2.8.1/weave-daemonset-k8s.yaml

→ Error :D, message:Network plugin returns error: cni plugin not initialized

→ resolved through sudo systemctl restart kubelet containerd.service docker.service docker.socket

Setting workernodes

→ run this on master node to find join command; kubeadm token create --print-join-command

→ run the output of above command on worker nodes, I.e ;

sudo kubeadm join 192.168.0.241:6443 --token cgdog6.gcc03udj2aj55gc1 --discovery-token-ca-cert-hash sha256:672b0059f619bf1e57d09e4a8aa5eb8e3d05d67fd7c85447900bf0f861a5c2fd --cri-socket unix:///var/run/containerd/containerd.sock