**CAPESTONE PROJECT-1**

**POST GRADUATE PROGRAM IN DEVOPS**

**NAME: TATA SAIKRISHNA**

**MAIL:dcgm.crmr@gmail.com**

**Mobile:7382801892**

**Registration ID:731272**

**https://github.com/tatasaikrishna/devops-capstone-project.git**

**Infra Optimization.**

Course-end Project 1

DESCRIPTION

Create a DevOps infrastructure for an e-commerce application to run on high-availability mode.

**Background of the problem statement:**  
A popular payment application, **EasyPay**where users add money to their wallet accounts, faces an issue in its payment success rate. The timeout that occurs with  
the connectivity of the database has been the reason for the issue.  
While troubleshooting, it is found that the database server has several downtime instances at irregular intervals. This situation compels the company to create their own infrastructure that runs in high-availability mode.  
Given that online shopping experiences continue to evolve as per customer expectations, the developers are driven to make their app more reliable, fast, and secure for improving the performance of the current system.

**Implementation requirements:**

1. Create the cluster (EC2 instances with load balancer and elastic IP in case of AWS)
2. Automate the provisioning of an EC2 instance using Ansible or Chef Puppet
3. Install Docker and Kubernetes on the cluster
4. Implement the network policies at the database pod to allow ingress traffic from the front-end application pod
5. Create a new user with permissions to create, list, get, update, and delete pods
6. Configure application on the pod
7. Take snapshot of ETCD database
8. Set criteria such that if the memory of CPU goes beyond 50%, environments automatically get scaled up and configured

**The following tools must be used:**

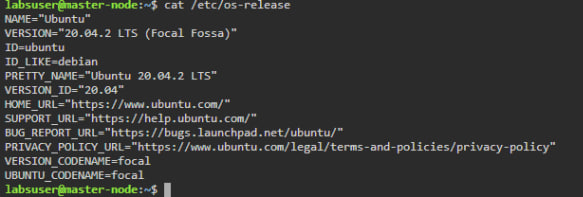
1. EC2
2. Kubernetes
3. Docker
4. Ansible or Chef or Puppet

**The following things to be kept in check:**

1. You need to document the steps and write the algorithms in them.
2. The submission of your GitHub repository link is mandatory. In order to track your tasks, you need to share the link of the repository.
3. Document the step-by-step process starting from creating test cases, then executing them, and recording the results.
4. You need to submit the final specification document, which includes:

* Project and tester details
* Concepts used in the project
* Links to the GitHub repository to verify the project completion
* Your conclusion on enhancing the application and defining the USPs (Unique Selling Points)

1. **Install Docker and Kubernetes on the cluster**



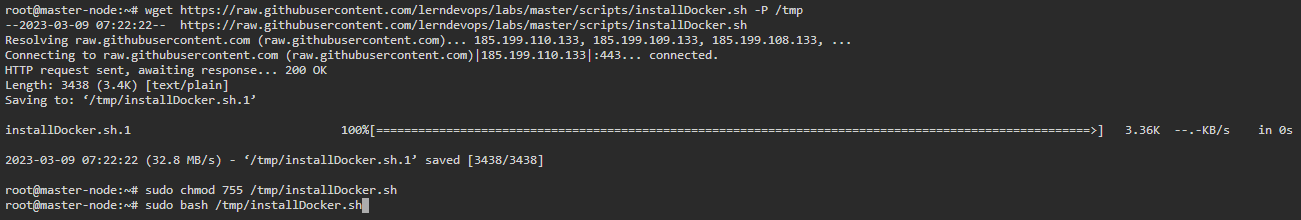
. **Install Docker**

#sudo wget #https://raw.githubusercontent.com/lerndevops/labs/master/scripts/installDocker.sh P /tmp

#sudo chmod 755 /tmp/installDocker.sh

#sudo bash /tmp/installDocker.sh

#sudo systemctl restart docker.service



**Install CRI-Docker**

#sudo wget https://raw.githubusercontent.com/lerndevops/labs/master/scripts/installCRIDockerd.sh -P /tmp

#sudo chmod 755 /tmp/installCRIDockerd.sh

#sudo bash /tmp/installCRIDockerd.sh

#sudo systemctl restart cri-docker.service

**Install kubeadm,kubelet,kubectl**

#sudo wget https://raw.githubusercontent.com/lerndevops/labs/master/scripts/installK8S.sh -P /tmp

#sudo chmod 755 /tmp/installK8S.sh

#sudo bash /tmp/installK8S.sh

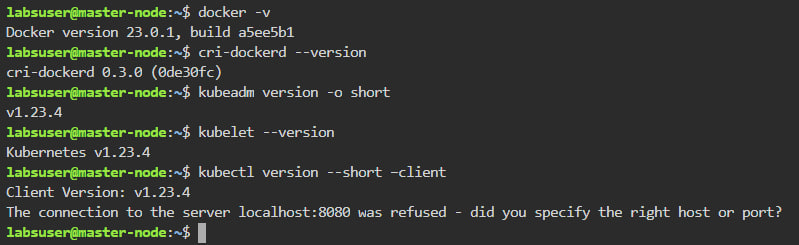
#docker -v

#cri-dockerd --version

#kubeadm version -o short

#kubelet --version

#kubectl version --short –client



**Initialize kubernetes Master Node**

#sudo kubeadm init --cri-socket unix:///var/run/cri-dockerd.sock --ignore-preflight-errors=all

#sudo mkdir -p $HOME/.kube

#sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config

#sudo chown $(id -u):$(id -g) $HOME/.kube/config

**Install networking driver -- Weave/flannel/canal/calico etc... below installs calico networking driver**

#kubectl apply -f https://raw.githubusercontent.com/projectcalico/calico/v3.24.1/manifests/calico.yaml

#kubectl get nodes

**Master Node to get join token**

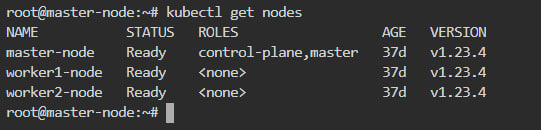
#kubeadm token create --print-join-command

copy the kubeadm join token from master & ensure to add --cri-socket unix:///var/run/cri-dockerd.sock as below & then run on worker nodes

kubeadm join 10.128.15.231:6443 --cri-socket unix:///var/run/cri-dockerd.sock --token mks3y2.v03tyyru0gy12mbt \

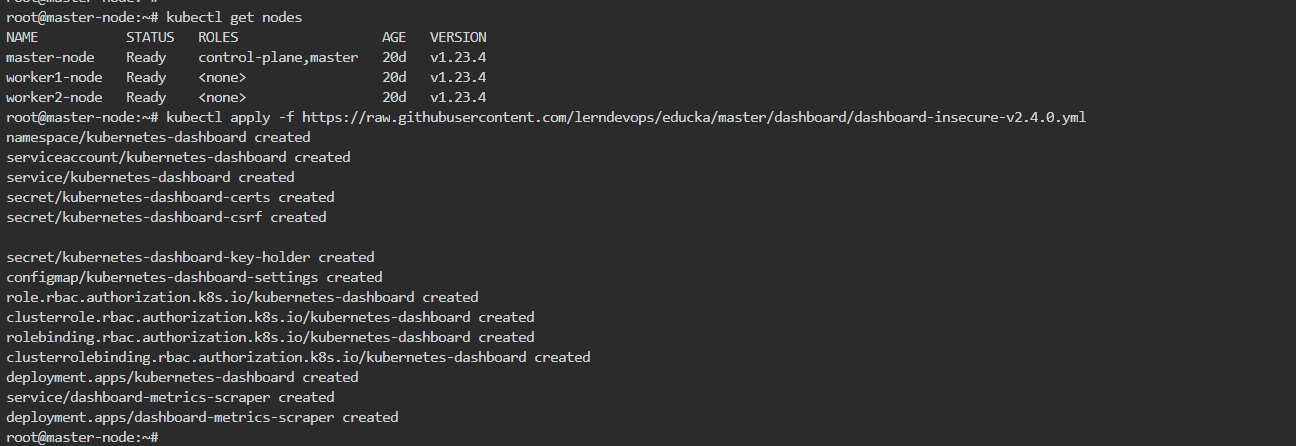
--discovery-token-ca-cert-hash sha256:3de23d42c7002be0893339fbe558ee75e14399e11f22e3f0b34351077b7c4b56

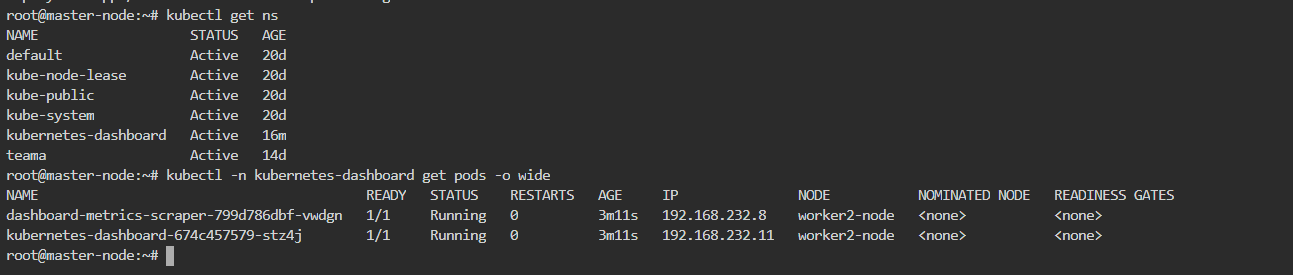
#kubectl get nodes



## Deploying the Dashboard UI

kubectl apply -f https://raw.githubusercontent.com/kubernetes/dashboard/v2.7.0/aio/deploy/recommended.yaml





# kubectl get ns

#kubectl –n kubernetes-dashboard get pods –o wide