Kubernetes Playbook

SETUP MASTER

Setup Sudo user

Sudo su

Disable Swap Memory

```
swapoff -a
```

Install Docker

```
apt install docker.io
```

Add Key to local repo

```
curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | apt-
key add -
```

Add Kubernetes packages to local repo

```
cat <<EOF >/etc/apt/sources.list.d/kubernetes.list
deb http://apt.kubernetes.io/ kubernetes-xenial main
EOF
```

Update repo

```
apt update
```

Install kubernetes packages

```
apt install -y kubelet kubeadm kubectl
```

Configure Kubernetes Master

```
kubeadm init
```

Update the changes

```
mkdir -p $HOME/.kube
sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
sudo chown $(id -u):$(id -g) $HOME/.kube/config
```

Setup Weaver as Pod Network

```
sysctl net.bridge.bridge-nf-call-iptables=1
```

```
kubectl apply -f "https://cloud.weave.works/k8s/net?k8s-version=$
(kubectl version | base64 | tr -d '\n')"
```

Preserve the token – to be used by kubernetes worker nodes

```
kubeadm join 10.0.0.7:6443 --token 6gwnc1.luzwxzyh5iou6dh3 --discov-
ery-token-ca-cert-hash sha256:329404342f0269401a3b0c49d595b-
b58e28037e9793936b7dc1e446755be27f2
```

Install Kubernetes UI – Dashboard

```
kubectl apply -f https://raw.githubusercontent.com/kubernetes/
dashboard/master/src/deploy/recommended/kubernetes-dashboard-
.yaml
```

Access Kubernetes Dashboard remotely

Get Master IP address

```
kuberctl cluster-info
```

```
kubectl proxy --port=8001 -address=10.0.0.7(the master host address) -
-accept-hosts='^.*$' -accept-paths='.'
```

SETUP WORKER NODE

Setup Sudo user
Sudo su

Disable Swap Memory

```
swapoff -a
```

Install Docker

```
apt install docker.io
```

Add Key to local repo

```
curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | apt-
key add -
```

Add Kubernetes packages to local repo

```
cat <<EOF >/etc/apt/sources.list.d/kubernetes.list
deb http://apt.kubernetes.io/ kubernetes-xenial main
EOF
```

Update repo

```
apt update
```

Install kubernetes packages

```
apt install -y kubelet kubeadm kubectl
```

Join Master Node

```
kubeadm join 10.0.0.7:6443 --token 6gwnc1.luzwxzyh5iou6dh3 --discov-
ery-token-ca-cert-hash sha256:329404342f0269401a3b0c49d595b-
b58e28037e9793936b7dc1e446755be27f2
```

DEPLOY SAMPLE NODE APPLICATION

Create Folder

```
mkdir node-sample
```

```
cd node-sample/
```

Create Node application

```
nano server.js
var http = require('http');
var handleRequest = function(request, response) {
  response.writeHead(200);
  response.end("Hello World!");
}
var www = http.createServer(handleRequest);
www.listen(8080);
```

Save the content to server.js

Create Dockerfile

Add below content

```
FROM node:4.4
EXPOSE 8080
COPY server.js .
CMD node server.js
```

Build Docker Image

```
sudo docker build -t node-hello-world:v1 .
```

Run Node application via docker

```
sudo docker run -d -p 8081:8080 --name node-hello-world node-hello-world:v1
```

Access the application

```
curl localhost:8081
```

Remove Docker container

```
docker rm -f node-hello-world
```

Push Image to Docker Hub

sudo docker login - Login with your docker repo credentials

```
sudo docker tag 94603241d694 mohsinkd786/node-hello-world
sudo docker push mohsinkd786/node-hello-world
```

Deploy Application using Kubernetes

Follow the below steps incase of private repo

```
kubectl run hello-world-1 --image=mohsinkd786/node-hello-world --port=8081
```

Verify Deployment

kubectl get deployments

Verify Pod

kubectl get pods

Expose Service

kubectl expose deployment hello-world-1 –type="LoadBalancer"

Verify Service

kubectl get services

Wait for few minutes the service takes some time to allocate an IP

AutoScaling

kubectl scale deployment hello-world-1 --replicas=2

Horizontal Scaling

kubectl autoscale deployment hello-world-1 --cpu-percent=50 --min=1 --max=10
horizontalpodautoscaler.autoscaling/hello-world-1 autoscaled

Get Metrics

kubectl get hpa

INCASE OF Private Registry

Create the Secret for the registry

kubectl create secret docker-registry regcred --docker-server=<your-registry-server> --docker-username=<your-name> --docker-password=<your-pwrd> –docker-email=<your-email>

Convert/ View that into Yaml

kubectl get secret regcred --output=yaml

Now the spec for the image while using deploy should look like below

```
apiVersion: apps/v1 # for versions before 1.8.0 use apps/v1beta1
kind: Deployment
metadata:
  name: spring-spotify-boot
spec:
  selector:
    matchLabels:
      app: spring-spotify-boot
  replicas: 2 # tells deployment to run 3 pods matching the tem-
plate
  template: # create pods using pod definition in this template
    metadata:
      labels:
        app: spring-spotify-boot
    spec:
      containers:
      - name: spring-spotify-boot
        image: mohsinkd786/spring-spotify-boot
        imagePullSecrets:
           - name: regcred
        ports:
        - containerPort: 8080
          name: server
```

Save the content as **spring-spotify-boot-app.yml**

Deploy the application in Kubernetes
kubectl create -f **spring-spotify-boot-app.yml**