Deploy the Application Using the Kubernetes Dashboard.

Course-end Project 1

DESCRIPTION

Project Agenda: To deploy the application using the Kubernetes dashboard.

Description:

Your organization is looking to create a multi-tier application based on PHP and MySQL. Your job is to deploy this application using the Kubernetes dashboard. Create a user (service account) with the name of Sandry and make sure to assign her an admin role. WordPress and MySQL Pods should use node3 as an NFS storage server using static volumes. WordPress applications must verify the MySQL Service before getting it deployed. If the MySQL Service is not present, then the WordPress Pod should not be deployed. These all should be restricted to the namespace called cep-project1 and must have 3 svcs, and 3 Pods as a max quota. All sensitive data should be used using secrets and non-sensitive data using configmaps.

Tools Required: kubeadm, kubectl, kubelet, and Docker

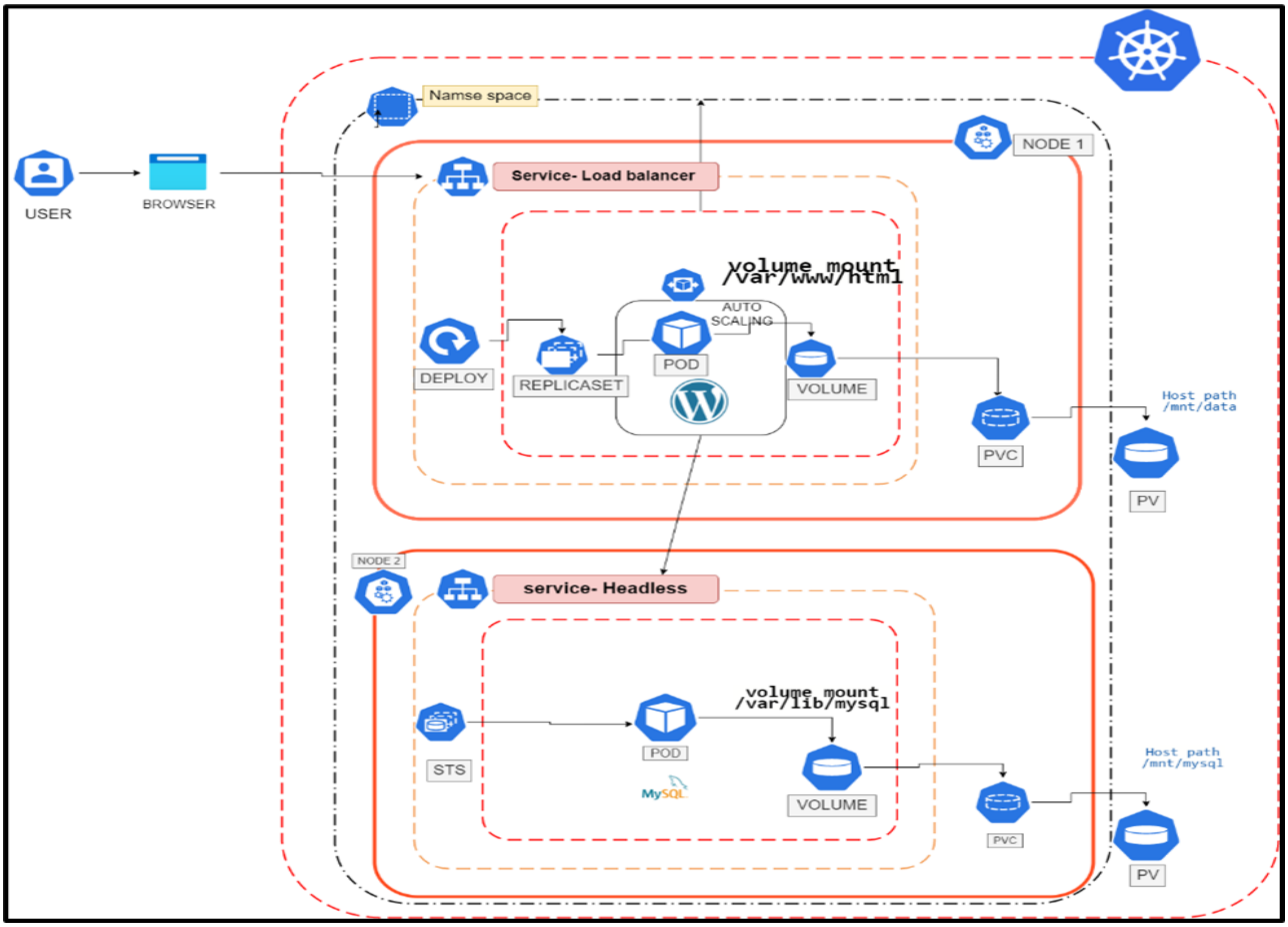
Expected Deliverables:

Note: Kubernetes cluster needs to be set up in your lab machine to perform the above project

1. Introduction

In this we will create a Kubernetes multi-node Cluster in a Virtual Machine Instance (EC2) and deploy a WordPress application on top of it.

Here, I am developing a cluster that gives us complete control over the nodes.



3. Perquisites:
To Deploy a WordPress application in Kubernetes, you will need the following:

- An AWS account
- Kubeadm Tool
- Calico Network plugin
- Docker

4. Kubernetes Components Leveraged:

- Deployment
- Stateful set
- Services
- Secrets
- Volumes
- 3 EC2 instances: One master node, two worker nodes

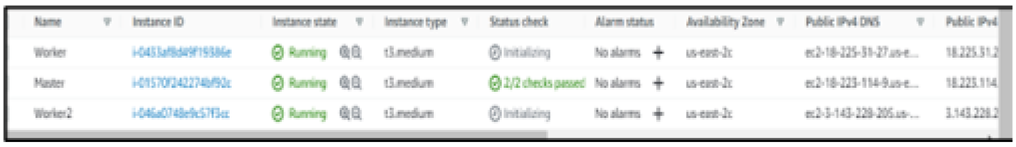
5. A brief overview of the steps involved:

1. Launching three instances
2. Setup kubeadm cluster
3. Joining the worker nodes with the master node
4. Deploying WordPress application

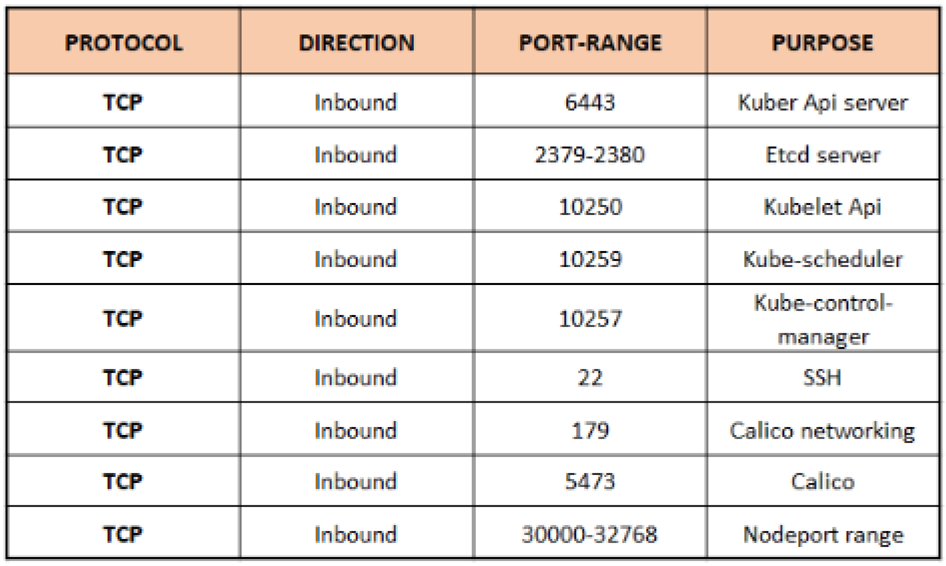
6. Step-by-Step Guide

Step 1: Create three instances of t3.medium running ubuntu 20.14 in the AWS console in any region.

Step 2: Add the security group Kubernetes-related port number and calico port number



Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4
Worker	i-0453af8d49f19386e	Running	t3.medium	Initializing	No alarms	us-east-2c	ec2-18-225-31-27.us-e...	18.225.31.2
Master	i-01570f242274bf92c	Running	t3.medium	2/2 checks passed	No alarms	us-east-2c	ec2-18-223-114-9.us-e...	18.223.114
Worker2	i-046a0748e9c57f3cc	Running	t3.medium	Initializing	No alarms	us-east-2c	ec2-3-143-228-205.us-...	3.143.228.2



PROTOCOL	DIRECTION	PORT-RANGE	PURPOSE
TCP	Inbound	6443	Kuber Api server
TCP	Inbound	2379-2380	Etcd server
TCP	Inbound	10250	Kubelet Api
TCP	Inbound	10259	Kube-scheduler
TCP	Inbound	10257	Kube-control-manager
TCP	Inbound	22	SSH
TCP	Inbound	179	Calico networking
TCP	Inbound	5473	Calico
TCP	Inbound	30000-32768	Nodeport range

Step 3: Tag one Instance as a Master node,

Remaining instances as worker one and worker two nodes.

SSH into the master and worker instance and Run the following commands.

#REMOVE OLD DOCKER

```
sudo apt-get remove docker docker-engine docker.io contained runs
```

#INSTALL DOCKER PRE-REQUISITES

```
sudo apt-get update
sudo apt-get install -y apt-transport-https ca-certificates curl gnupg lsb-release
```

#ADD GPG KEY

```
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /usr/share/keyrings/docker-archive-keyring.gpg
echo \
  "deb [arch=amd64 signed-by=/usr/share/keyrings/docker-archive-keyring.gpg] https://download.docker.com/linux/ubuntu \
  $(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
```

#INSTALL DOCKER ENGINE

```
sudo apt-get update
sudo apt-get install -y docker-ce docker-ce-cli containerd.io
sudo docker run hello-world
cat <<EOF | sudo tee /etc/modules-load.d/k8s.conf
overlay
br_netfilter
EOF
```

Restart nodes to load them the br_netfilter and overlay

```
sudo modprobe overlay
sudo modprobe br_netfilter
```

#ALLOW BRIDGED TRAFFIC FOR KUBEADM

```
cat <<EOF | sudo tee /etc/sysctl.d/k8s.conf
net.bridge.bridge-nf-call-ip6tables = 1
net.bridge.bridge-nf-call-iptables = 1
EOF
sudo sysctl --system
```

#INSTALL K8S PRE-REQUISITES

```
sudo apt-get update
sudo apt-get install -y apt-transport-https ca-certificates curl
```

#DOWNLOAD GOOGLE CLOUD PUBLIC SIGNINIG KEY

```
sudo curl -fsSLo /usr/share/keyrings/kubernetes-archive-keyring.gpg
https://packages.cloud.google.com/apt/doc/apt-key.gpg
```

ADD K8S APT REPO

```
echo "deb [signed-by=/usr/share/keyrings/kubernetes-archive-keyring.gpg]
https://apt.kubernetes.io/ kubernetes-xenial main" | sudo tee
/etc/apt/sources.list.d/kubernetes.list
```

INSTALL K8S COMPONENTS

```
sudo apt-get update
 sudo apt-get install -y kubelet=1.20.1-00 kubeadm=1.20.1-00 kubectl=1.20.1-00
sudo apt-mark hold kubelet kubeadm kubectl
kubectl taint nodes --all node-role.kubernetes.io/master-
sudo touch /etc/docker/daemon.json
cat <<EOF | sudo tee /etc/docker/daemon.json
{
      "exec-opts": ["native.cgroupdriver=systemd"]
}
EOF
sudo swapoff –a      #to disable the swapping
sudo sed -i '/ swap / s/^/#/' /etc/fstab      # To persist the swap disable
```

#init

```
sudo systemctl daemon-reload
sudo systemctl restart docker
```

Step 4: On the Master, node instance, follow the below commands to Initiate the Kubeadm, generate the token and Install Calico network plugin.

#Kubeadm init

```
kubeadm init --pod-network-cidr=10.244.0.0/16 --ignore-preflight-errors=all
export KUBECONFIG=/etc/kubernetes/admin.conf
sudo cp /etc/kubernetes/admin.conf $HOME/admin.conf
sudo chown $(id -u):$(id -g) $HOME/admin.conf
```

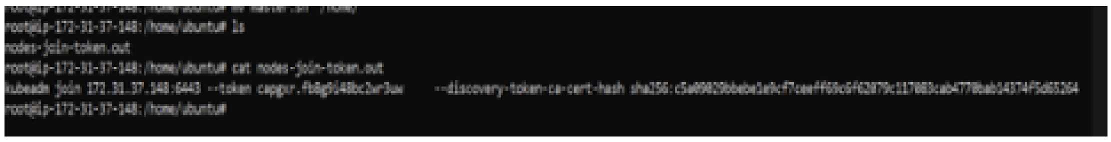
#Generate Token

```
token=$(kubeadm token generate)
rm -f home/ubuntu/nodes-join-token.out
kubeadm token create $token --print-join-command --ttl=0 > /home/ubuntu/nodes-join-token.out
cat /home/ubuntu/nodes-join-token.out
```

#Install calico networking

```
kubectl get cs
kubectl get components tatus
kubectl cluster-info
kubectl get pods -n kube-system

mv $HOME/.kube $OME/.kube.bak
mkdir $HOME/.kube
sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
sudo chown $(id -u):$(id -g) $HOME/.kube/config
sudo systemctl restart docker.service
sudo systemctl enable docker.service
sudo service kubelet restart
```

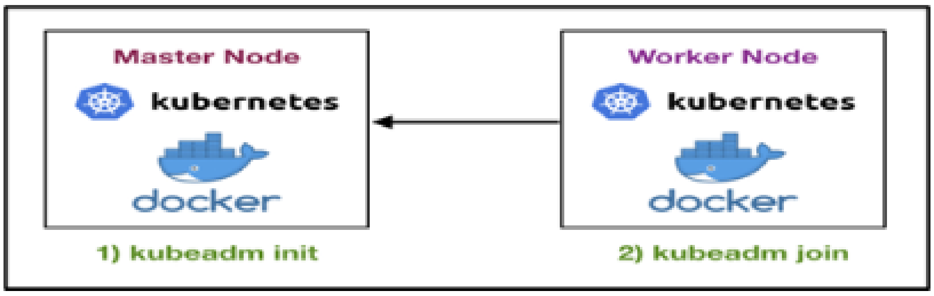
Step 5: A file is created in the master node named "nodes-join-token". Open it and copy the token

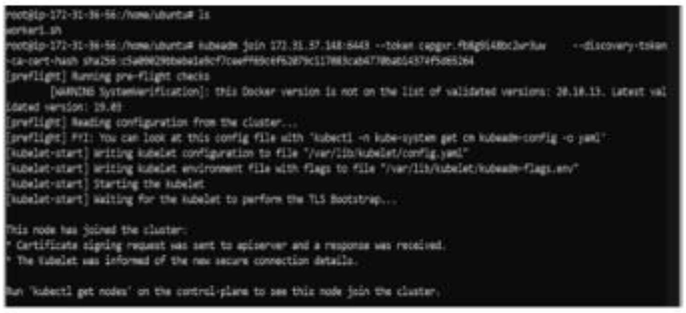


```
root@ip-172-31-37-148:/home/ubuntu# mv master.sh /home/
root@ip-172-31-37-148:/home/ubuntu# ls
nodes-join-token.out
root@ip-172-31-37-148:/home/ubuntu# cat nodes-join-token.out
kubeadm join 172.31.37.148:6443 --token cqpgvr.fb8g6148bc2wr3uw     --discovery-token-ca-cert-hash sha256:c5a09829bbebe1a9cf7ceeff69c6f62879c117083cab4778bab14374f5d65264
root@ip-172-31-37-148:/home/ubuntu#
```

```
Cat nodes-join-token
```

Step 6: Past the token in both the worker nodes to join them into the cluster.

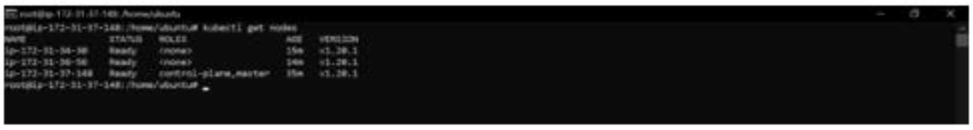




```
root@ip-172-31-36-56:/home/ubuntu# ls
worker1.sh
root@ip-172-31-36-56:/home/ubuntu# kubeadm join 172.31.37.148:6443 --token capgxr.fb8g9148bc2wr3uw     --discovery-token
-ca-cert-hash sha256:c5a0002988ebe1a0cf7caeff69cef62879c117883cab4778bab14374f5d85264
[preflight] Running pre-flight checks
        [WARNING SystemVerification]: this Docker version is not on the list of validated versions: 20.10.13. Latest val
idated version: 19.03
[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.
```

Step 7: After the above step, run the below command in the master node to verify whether nodes are joined.

```
$:-Kubectl gets nodes
```



```
root@ip-172-31-37-148: /home/ubuntu
root@ip-172-31-37-148:/home/ubuntu# kubectl get nodes
NAME               STATUS   ROLES                  AGE   VERSION
ip-172-31-34-38    Ready    <none>                 15m   v1.20.1
ip-172-31-36-56    Ready    <none>                 14m   v1.20.1
ip-172-31-37-148   Ready    control-plane,master   15m   v1.20.1
root@ip-172-31-37-148:/home/ubuntu# _
```

Successfully created Kubernetes Cluster using Kubeadm.

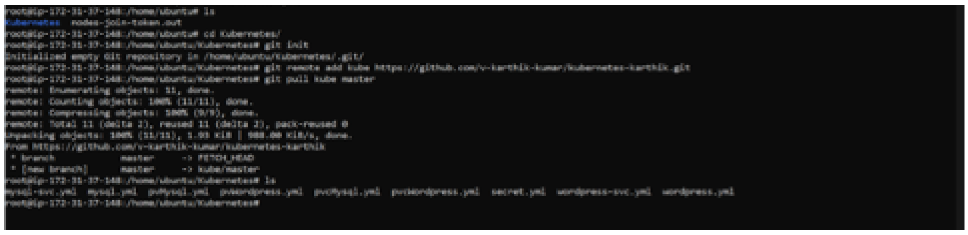
Step 8: SSH into the master node and execute the below code To Download the WordPress application.

#Install GIt

```
Sudo apt-get install git-all
```

#make a directory

```
mkdir /home/ubuntu/Kubernetes
cd /home/ubuntu/Kubernetes
git init
git remote add kube https://github.com/v-karthik-kumar/kubernetes-karthik.git
git pull kube master
```



```
root@ip-172-31-37-148:/home/ubuntu# ls
Kubernetes  nodes-join-token.out
root@ip-172-31-37-148:/home/ubuntu# cd Kubernetes/
root@ip-172-31-37-148:/home/ubuntu/Kubernetes# git init
Initialized empty Git repository in /home/ubuntu/Kubernetes/.git/
root@ip-172-31-37-148:/home/ubuntu/Kubernetes# git remote add kube https://github.com/v-karthik-kumar/kubernetes-karthik.git
root@ip-172-31-37-148:/home/ubuntu/Kubernetes# git pull kube master
remote: Enumerating objects: 11, done.
remote: Counting objects: 100% (11/11), done.
remote: Compressing objects: 100% (9/9), done.
remote: Total 11 (delta 2), reused 11 (delta 2), pack-reused 0
Unpacking objects: 100% (11/11), 1.93 KiB | 988.00 KiB/s, done.
From https://github.com/v-karthik-kumar/kubernetes-karthik
 * branch            master     -> FETCH_HEAD
 * [new branch]      master     -> kube/master
root@ip-172-31-37-148:/home/ubuntu/Kubernetes# ls
mysql-svc.yml  mysql.yml  pvMysql.yml  pvWordpress.yml  pvcMysql.yml  pvcWordpress.yml  secret.yml  wordpress-svc.yml  wordpress.yml
root@ip-172-31-37-148:/home/ubuntu/Kubernetes#
```

Step 9: Deploy yaml code in a sequence on the master nodes.

Step 9.1: Go to the Kubernetes directory

```
Cd /home/ubuntu/Kubernetes
```

Step 9.2: Deploy persistent volume for MySQL

```
Kubectl apply -f pvMysql.yml
```

Step 9.3: Deploy persistent volume claim for MySQL

```
Kubectl apply -f  pvcMysql.yml
```

Step 9.4: Deploy service for MySQL

```
Kubectl apply -f mysql-svc.yml
```

Step 9.5: Deploy secrets

```
Kubectl apply -f  secret.yml
```

Step 9.6: Deploy Deployment of mysql

```
Kubectl apply -f  mysql.yml
```

Step 9.7: Deploy persistent volume for WordPress

```
Kubectl apply -f  pvWordpress.yml
```

Step 9.8: Deploy persistent volume claim for WordPress

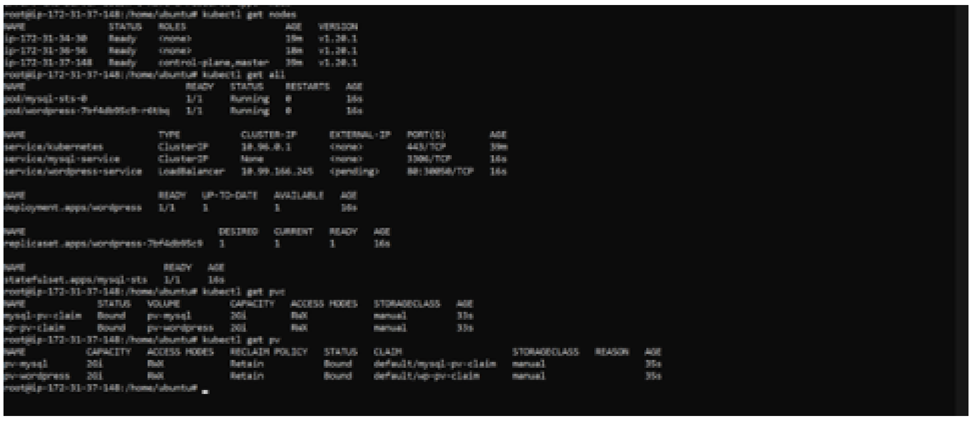
```
Kubectl apply -f  pvcWordpress.yml
```

Step 9.9: Deploy service for WordPress

```
Kubectl apply -f wordpress-svc.yml
```

Step 9.10: Deploy Deployment of WordPress

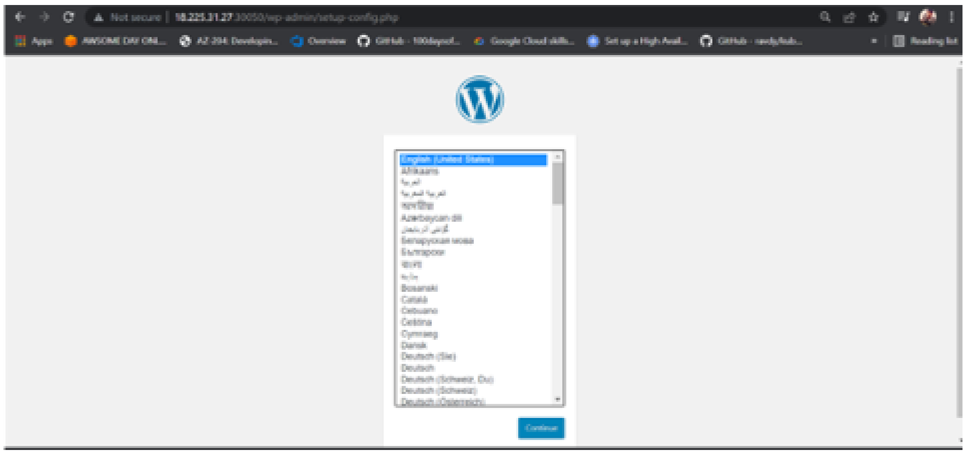
```
Kubectl apply -f  wordpress.yml
```



```
root@ip-172-31-37-148:/home/ubuntu# kubectl get nodes
NAME               STATUS   ROLES                  AGE   VERSION
ip-172-31-34-30    Ready    <none>                 19m   v1.20.1
ip-172-31-36-56    Ready    <none>                 18m   v1.20.1
ip-172-31-37-148   Ready    control-plane,master   39m   v1.20.1
root@ip-172-31-37-148:/home/ubuntu# kubectl get all
NAME                             READY   STATUS    RESTARTS   AGE
pod/mysql-sts-0                  1/1     Running   0          16s
pod/wordpress-7bf4db95c9-r6tbq   1/1     Running   0          16s

NAME                        TYPE           CLUSTER-IP      EXTERNAL-IP   PORT(S)        AGE
service/kubernetes          ClusterIP      10.96.0.1       <none>        443/TCP        39m
service/mysql-service       ClusterIP      None            <none>        3306/TCP       16s
service/wordpress-service   LoadBalancer   10.99.166.245   <pending>     80:30050/TCP   16s

NAME                        READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/wordpress   1/1     1            1           16s

NAME                                   DESIRED   CURRENT   READY   AGE
replicaset.apps/wordpress-7bf4db95c9   1         1         1       16s

NAME                         READY   AGE
statefulset.apps/mysql-sts   1/1     16s
root@ip-172-31-37-148:/home/ubuntu# kubectl get pvc
NAME             STATUS   VOLUME         CAPACITY   ACCESS MODES   STORAGECLASS   AGE
mysql-pv-claim   Bound    pv-mysql       20Gi       RWX            manual         33s
wp-pv-claim      Bound    pv-wordpress   20Gi       RWX            manual         33s
root@ip-172-31-37-148:/home/ubuntu# kubectl get pv
NAME           CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS   CLAIM                    STORAGECLASS   REASON   AGE
pv-mysql       20Gi       RWX            Retain           Bound    default/mysql-pv-claim   manual                  35s
pv-wordpress   20Gi       RWX            Retain           Bound    default/wp-pv-claim      manual                  35s
root@ip-172-31-37-148:/home/ubuntu# _
```

Step 10: Access the WordPress application in the browser.

```
http://<master/worker instance public Ip address>:30050
```



Not secure | 18.225.31.27:30050/wp-admin/setup-config.php

mysql-svc.yml

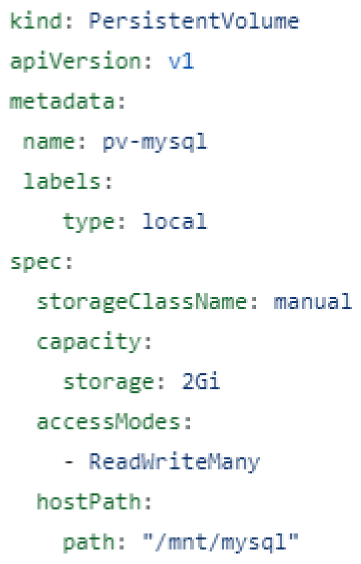
```
apiVersion: v1
kind: Service
metadata:
  name: mysql-service
  labels:
    app: capstone
    tier: backend
spec:
  ports:
#     - protocol: TCP
    - port: 3306
      targetPort: 3306
  selector:
    app: capstone
    tier: backend
  clusterIP: None
```

mysql.yml

```
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: mysql-sts
spec:
  serviceName: mysql-service
  replicas: 1
  selector:
    matchLabels:
      app: capstone
      tier: backend
  template:
    metadata:
      labels:
        app: capstone
        tier: backend
    spec:

      containers:
      - name: mysql
        image: mysql:5.6
        env:
        - name: MYSQL_ROOT_PASSWORD
          valueFrom:
            secretKeyRef:
              name: mysql-pass
              key: password
        - name: MYSQL_DATABASE
          valueFrom:
            secretKeyRef:
              name: mysql-pass
              key: databaseName
        - name: MYSQL_USER
          valueFrom:
            secretKeyRef:
              name: mysql-pass
              key: username
        - name: MYSQL_PASSWORD
          valueFrom:
            secretKeyRef:
```

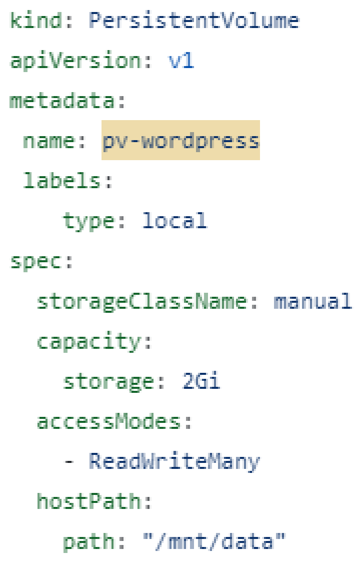
```
            name: mysql-pass
            key: password
      ports:
      - containerPort: 3306
        name: web
      volumeMounts:
      - name: mysql-persistent-storage
        mountPath: /var/lib/mysql
    volumes:
    - name: mysql-persistent-storage
      persistentVolumeClaim:
        claimName: mysql-pv-claim
```

pvMysql.yml



```
kind: PersistentVolume
apiVersion: v1
metadata:
 name: pv-mysql
 labels:
    type: local
spec:
  storageClassName: manual
  capacity:
    storage: 2Gi
  accessModes:
    - ReadWriteMany
  hostPath:
    path: "/mnt/mysql"
```

pvWordpress.yml



```
kind: PersistentVolume
apiVersion: v1
metadata:
 name: pv-wordpress
 labels:
    type: local
spec:
  storageClassName: manual
  capacity:
    storage: 2Gi
  accessModes:
    - ReadWriteMany
  hostPath:
    path: "/mnt/data"
```

pvcMysql.yml

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: mysql-pv-claim
  labels:
    app: capstone
spec:
  storageClassName: manual
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: 1Gi
```

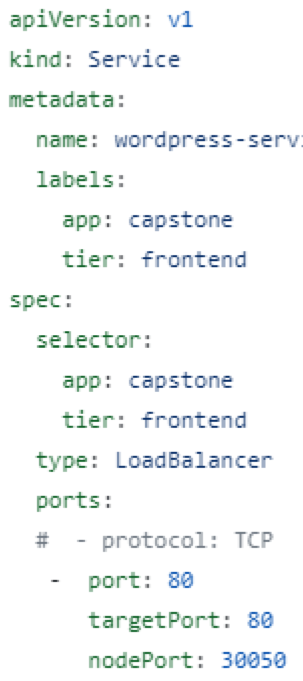
pvcWordpress.yml

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: wp-pv-claim
  labels:
    app: capstone
spec:
  storageClassName: manual
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: 1Gi
```

secret.yml

```
apiVersion: v1
kind: Secret
metadata:
  name: mysql-pass
type: Opaque
data:
  password: Y2Fwc3RvbmU=   #capstone
  username: Y2Fwc3RvbmU=   #capstone
  databaseName: Y2Fwc3RvbmU=   #capstone
```

wordpress-svc.yml



```yaml
apiVersion: v1
kind: Service
metadata:
  name: wordpress-service
  labels:
    app: capstone
    tier: frontend
spec:
  selector:
    app: capstone
    tier: frontend
  type: LoadBalancer
  ports:
  #  - protocol: TCP
   -  port: 80
      targetPort: 80
      nodePort: 30050
```

wordpress.yml

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: wordpress
  labels:
    app: capstone
    tier: frontend
spec:
  replicas: 1
  selector:
    matchLabels:
      app: capstone
      tier: frontend
  template:
    metadata:
      labels:
        app: capstone
        tier: frontend
    spec:
      volumes:
      - name: wordpress-persistent-storage
        persistentVolumeClaim:
          claimName: wp-pv-claim
      containers:
      - name: wordpress
        image: wordpress:5.1.1-php7.3-apache
        ports:
        - containerPort: 80
        volumeMounts:
        - name: wordpress-persistent-storage
          mountPath: /var/www/html
```