**Setup Kubernetes Cluster on AWS**

Create Ubuntu EC2 instance (m4.xlarge)

**Install AWSCLI**

curl https://s3.amazonaws.com/aws-cli/awscli-bundle.zip -o awscli-bundle.zip

apt install unzip python

unzip awscli-bundle.zip

./awscli-bundle/install -i /usr/local/aws -b /usr/local/bin/aws

**Install kubectl**

curl -LO https://storage.googleapis.com/kubernetes-release/release/$(curl -s https://storage.googleapis.com/kubernetes-release/release/stable.txt)/bin/linux/amd64/kubectl

chmod +x ./kubectl

sudo mv ./kubectl /usr/local/bin/kubectl

**Create an IAM user/role with Route53, EC2, IAM and S3 full access and Attach IAM role to our server**

Actions-->Instance settings--> Attach/Replace IAM Role--> select our IAM role

aws configure

**Install kops on instance:**

curl -LO https://github.com/kubernetes/kops/releases/download/$(curl -s https://api.github.com/repos/kubernetes/kops/releases/latest | grep tag\_name | cut -d '"' -f 4)/kops-linux-amd64

chmod +x kops-linux-amd64

sudo mv kops-linux-amd64 /usr/local/bin/kops

**Create a Route53 private hosted zone (you can create Public hosted zone if you have a domain)**

**create an S3 bucket**

aws s3 mb s3://dev.k8s.kubernetes.in

**Expose environment variable:**

export KOPS\_STATE\_STORE=s3://dev.k8s.kubernetes.in

**Generate sshkeys before creating kubernetes cluster**

ssh-keygen

**Create kubernetes cluster definitions on S3 bucket**

kops create cluster --cloud=aws --zones=us-west-2c --name=dev.k8s.kubernetes.in --dns-zone=kubernetes.in --dns private

**Create kubernetes cluster**

kops update cluster dev.k8s.kubernetes.in --yes

**Validate your cluster**

kops validate cluster

**check whether nodes are initialized or not**

kubectl get nodes

**Deploying Nginx container on Kubernetes**

**Deploying Nginx Container**

kubectl run sample-nginx --image=nginx --replicas=2 --port=80

kubectl get deployments

**Expose the deployment as service. This will create 2 containers and allow us to publicly access them:**

kubectl expose deployment sample-nginx --port=80 --type=LoadBalancer

kubectl get services -o wide

**Deploying wordpress Web Application with MySQL in kubernetes**

we will setup Wordpress web site with MySQL backend support.**Create a Secret for MySQL Password**

kubectl create secret generic mysql-pass --from-literal=password=<YOUR\_PASSWORD>

**Verify the secret with below command**

kubectl get secrets

**Create MySQL Service, Persistent Volume Claim & Deployment**

**After creating volume and deployment (.yaml) give following commands**

kubectl create -f pv.yaml

kubectl create -f mysql-deployment.yaml

**Check overall status of the objects by using below command.**

kubectl get service,pvc,deployment,pods

**Go to shell inside the mysql container, log into mysql, and set up the DB:**

kubectl get pods

kubectl exec -it wordpress-mysql-abcd -- bash (replace with abcd with our pod name)

mysql -u root -ppassword

CREATE DATABASE wordpress;

**Create WordPress Service, Persistent Volume Claim & Deployment**

After creating volume and deployment (.yaml) give following commands

kubectl apply -f wordpress-datavolume-claim.yaml

kubectl apply -f wordpress-deployment.yaml

**Load Balancer Setup**

After creating loadbalancer (.yaml) give the following commands

kubectl apply -f DO-loadbalancer.yaml

**To delete cluster**

kops delete cluster dev.k8s.kubernetes.in --yes