1) First launch a instance on aws and access server on putty
2) Create a IAM user role in aws with administration access
3) Install aws cli and do

```
   aws configure
```

4) Install eksctl and kubectl for creating cluster and pods

```
  #link for installing :
https://www.hackerxone.com/2021/08/20/steps-to-install-kubectl-eksctl-on-ubuntu-20-04/
  #installing kubectl:
    apt-get update
    curl -o kubectl
https://amazon-eks.s3-us-west-2.amazonaws.com/1.21.2/2021-07-05/bin/linux/amd64/kubectl
    chmod +x ./kubectl
    mkdir -p $HOME/bin && cp ./kubectl $HOME/bin/kubectl && export
PATH=$PATH:$HOME/bin
    kubectl version
  #installing eksctl
    apt-get update
    curl --silent --location
"https://github.com/weaveworks/eksctl/releases/latest/download/eksctl_$(uname
-s)_amd64.tar.gz" | tar xz -C /tmp
    mv /tmp/eksctl /usr/local/bin
    eksctl version
```

5)Create cluster

```
eksctl create cluster \
 --name Kubernetes-cluster \
 --region us-east-2 \
 --version 1.21 \
 --managed \
 --nodegroup-name workergroup \
 --node-type t2.small \
 --nodes-min 1 \
 --nodes-max 4 \
 --node-volume-size 20 \
 --ssh-access \
 --ssh-public-key firstEc2 \
 --asg-access \
 --external-dns-access \
 --full-ecr-access \
 --kubeconfig /home/ubuntu/.kube/config
```

6)kubectl get nodes

```
  eksctl get cluster
  #if u want to delete your cluster then do
    eksctl delete cluster --name=clusterName
```

7)#create deployment and service set for database

```
 sudo nano database.yml
******************************database.yml**********************************************
apiVersion: apps/v1
kind: Deployment
metadata:
  name: mysql
  labels:
    app: mysql
spec:
  replicas: 1
  selector:
    matchLabels:
      app: mysql
  template:
    metadata:
      labels:
        app: mysql
    spec:
      containers:
      - name: mysql
        image: moshtab/database_imageupdated:latest
        resources:
              limits:
                cpu: 500m
              requests:
                cpu: 200m


---
kind: Service
apiVersion: v1
metadata:
  name: database-service
spec:
  selector:
    app: mysql
  type: ClusterIP
  ports:
    - name: mysql
      port: 3306
      targetPort: 3306
```

****

```
kubectl apply -f database.yml
kubectl get pods
```

8)#create deployment and service set for backend
  #Note:- The backend Image should contain the host name as the name of database service
   sundo nano backend.yml

***********************************backend.yml**

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: backend
  labels:
    app: backend
spec:
  replicas: 1
  selector:
    matchLabels:
      app: backend
  strategy:
    type: Recreate

  template:
    metadata:
      labels:
        app: backend
    spec:
      containers:
        - name: backend
          image: moshtab/backend_imageupdated:latest
          imagePullPolicy: Always
          ports:
            - name: tcp
              containerPort: 8000
          resources:
              limits:
                cpu: 500m
              requests:
                cpu: 200m
```

```
---
kind: Service
apiVersion: v1
metadata:
  name: backend
spec:
  selector:
    app: backend
  type: ClusterIP
  ports:
    - name: backend
      port: 8000
      targetPort: 8000
```
****

```
kubectl apply -f backend.yml
kubectl get pods
kubectl describe pod podName
```
9)#To enter into a backend pod and check whether the application is running or not
```
   kubectl exec -it NameOfPod -- /bin/bash     (NameOfPod will get from: kubectl get pods)
   curl http://localhost:8000
```
#If it shows connection refused to connect 8000 port then do (ps -elf)
```
   python manage.py runserver
```
#Then u can see the error like it is not connecting to your database service, so change the database service name as host name in settings.py
then apply it again for database service
```
   kubectl apply -f database.yml
```
10)#create deployment,service and configMap set for frontend
  #Note:- In frontend image it should have the proxypass as the backend service name
```
sudo nano frontend.yml
```
****************************frontend.yml**

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: frontend
  labels:
    app: frontend
spec:
  replicas: 1
  selector:
    matchLabels:
      app: frontend
  strategy:
    type: Recreate

  template:
    metadata:
      labels:
        app: frontend
    spec:
      containers:
      - name: frontend
        image: moshtab/frontend_imageupdated:latest
        imagePullPolicy: Always
        volumeMounts:
          - name: chatapp
            mountPath: /etc/nginx/conf.d/
            readOnly: true
        ports:
        - containerPort: 80
        resources:
          limits:
            cpu: 500m
          requests:
            cpu: 200m
      volumes:
      - name: chatapp
        configMap:
          name: configmap


kind: Service
apiVersion: v1
metadata:
  name: frontend
spec:
  selector:
    app: frontend
  type: ClusterIP
  ports:
    - name: frontend
      port: 80
      targetPort: 80

kind: ConfigMap
apiVersion: v1
metadata:
  name: configmap
data:
  nginxconf.conf: |
      server {
      listen 80 ;
      server_name _default;
      root /new_chatapp/fundoo;
      location / {
      proxy_pass http://backend:8000;
      }
      }
```
****

```
kubectl apply -f frontend.yml
kubectl get pods
```
9)#To enter into a frontend pod and check whether the application is running or not

```
    kubectl exec -it NameOfPod -- /bin/bash     (NameOfPod will get from: kubectl get pods)
    curl http://localhost
```
#If it shows connection refused to connect 80 port then do
#Check your nginx config syntax:

```
   nginx -t
```
#If you get an error, you would need to fix that problem like your having incorrect proxypass in nginx configuration in sites-available folder and then you could restart nginx:

```
   apt install systemctl
   systemctl restart nginx
   systemctl status nginx
```
#For troubleshooting nginx server link:
https://www.digitalocean.com/community/questions/how-to-troubleshoot-common-nginx-issues-on-linux-server

#If you get Syntax OK when running nginx -t then your configuration is correct, so I would recommend checking your error logs:

```
   tail -f /var/log/nginx/error.log
```
10)#Create a domain from freenom
   #Link: https://my.freenom.com/clientarea.php?action=domains
   #watch this video for creation of domain
   #https://youtu.be/3Uopc4AFjOY
11)After creation of domain configure it by creating a hosted zone in route 53 and copy four servers from route 53 to freenom (Manage Domain-Management tools-Nameservers-Use Custom name servers-change Nameservers)

12) #Now install Ingress-Controller

```
     kubectl apply -f
https://raw.githubusercontent.com/kubernetes/ingress-nginx/controller-v1.0.0/deploy/static/provider/cloud/deploy.yaml
```
13) #Create ingress to access your application from outside world

```
     sudo nano ingress.yml
```

ingress.yml

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: ingress
  annotations:
    kubernetes.io/ingress.class: nginx

spec:
 rules:
 - host: www.moshtab.tk
   http:
      paths:
      - path: /
        pathType: Prefix
        backend:
          service:
            name: frontend
            port:
              number: 80
```
****

```
kubectl apply -f ingress.yml
```
14) Now go to route 53 and create a record in that click on alias,select Application Load Balancer, select DNS of LB.

15)#Now check by

```
    curl www.moshtab.tk
```
#or check on browser

```
  www.moshtab.tk
```

16)#For AutoScaling Install Metric Server

```
   #link:  https://docs.aws.amazon.com/eks/latest/userguide/metrics-server.html
    kubectl apply -f
https://github.com/kubernetes-sigs/metrics-server/releases/latest/download/components.yaml
    kubectl get deployment metrics-server -n kube-system
   #paste resources in spec like
    resources:
              limits:
```

```
              cpu: 500m
            requests:
              cpu: 200m
```

17)#for horizontal pod autoscaling do

```
kubectl autoscale deployment frontend-6789f5654c-jl57w --cpu-percent=50 --min=1
--max=10
```

18)#Now go to duplicate session and increase the load by

```
kubectl run -i --tty load-generator --rm --image=busybox --restart=Never -- /bin/sh -c "while sleep
0.01; do wget -q -O- http://php-apache; done"
```

19)#Now come to Main session and check whether the pods are increasing or not by

```
kubectl get hpa
```

links for study:
https://www.hackerxone.com/2021/08/20/steps-to-install-kubectl-eksctl-on-ubuntu-20-04/
kubectl apply -f
https://raw.githubusercontent.com/kubernetes/ingress-nginx/controller-v1.0.0/deploy/static/provider/cloud/deploy.yaml
https://youtu.be/3Uopc4AFjOY
https://youtu.be/3BnrXapY7zo

Important commands:

```
eksctl create cluster
eksctl delete cluster --name=Kubernetes-cluster
kubectl apply -f ymlfile
kubectl delete -f ymlfile
kubectl describe pod podName
kubectl get pods
kubectl exec -it frontend-6789f5654c-jl57w -- /bin/bash
```

if you get post 8080 connection fail then run
---> aws eks update-kubeconfig --region ap-south-1 --name Kubernetes-cluster