Project 2 -Deploying PHP Guestbook application with Redis(kubeadm)

Build and deploy a simple, multi-tier web application using kubeadm cluster and containerd which must consist of the below components.

1.A single-instance Redis to store guestbook entries.

2.Multiple web frontend instances

Ref: https://kubernetes.io/docs/tutorials/stateless-application/guestbook/

Hint:

1.Start up a Redis leader.

2.Start up two Redis followers.

3.Start up the guestbook frontend.

4.Expose and view the Frontend Service (Use Kubernetes NodePort Service)

Installation:

As per the project requirement, we launched 3 instances, one with t2.medium and the other with t2.micro.

☐	K8s_Master	i-0a375cc24718efc3a	Running	t2.medium	2/2 checks passed	No alarms	+	us-west-2c	ec2-52-35-182-155.us-...	52.35.182.155
☐	Worker_1	i-0de962177fd428b9e	Running	t2.micro	2/2 checks passed	No alarms	+	us-west-2a	ec2-52-27-144-202.us-...	52.27.144.202
☑	Worker_2	i-012ecdd8c0d7b00d5	Running	t2.micro	2/2 checks passed	No alarms	+	us-west-2a	ec2-35-90-252-232.us-...	35.90.252.232

The installation of k8s was completed successfully on the respective server, and the worker nodes are added to the master node successfully.

```
ubuntu@master:~$ kubectl get nodes -o wide
NAME      STATUS   ROLES           AGE     VERSION   INTERNAL-IP    EXTERNAL-IP   OS-IMAGE             KERNEL-VERSION    CONTAINER-RUNTIME
master    Ready    control-plane   21m     v1.27.4   172.31.7.158   <none>        Ubuntu 20.04.6 LTS   5.15.0-1036-aws   containerd://1.6.12
worker1   Ready    <none>          4m1s    v1.27.4   172.31.24.89   <none>        Ubuntu 20.04.6 LTS   5.15.0-1036-aws   containerd://1.6.12
worker2   Ready    <none>          3m48s   v1.27.4   172.31.22.52   <none>        Ubuntu 20.04.6 LTS   5.15.0-1036-aws   containerd://1.6.12
```

The master node is the Redis leader and the worker node 1 and 2 are Redis follower.

1. Creating the Redis Leader Deployment:

```
root@master:/home/ubuntu# kubectl apply -f redis-leader-deployment.yaml
deployment.apps/redis-leader created
root@master:/home/ubuntu# kubectl get pods
NAME                              READY   STATUS    RESTARTS   AGE
redis-leader-58b566dc8b-2wc8p     1/1     Running   0          12s
root@master:/home/ubuntu# kubectl logs -f deployment/redis-leader
```

2. Creating the Redis leader Service:

```
root@master:/home/ubuntu# vi redis-leader-service.yaml
root@master:/home/ubuntu# kubectl apply -f redis-leader-service.yaml
service/redis-leader created
root@master:/home/ubuntu# kubectl get svc
NAME           TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)    AGE
kubernetes     ClusterIP   10.96.0.1       <none>        443/TCP    47m
redis-leader   ClusterIP   10.100.216.61   <none>        6379/TCP   8s
```

3. Redis Follower set-up to make it highly available:

```
root@master:/home/ubuntu# vi redis-follower-deployment.yaml
root@master:/home/ubuntu# kubectl apply -f redis-follower-deployment.yaml
deployment.apps/redis-follower created
root@master:/home/ubuntu# kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
redis-follower-6f6cd6cbdb-t6v5f     1/1     Running   0          9s
redis-follower-6f6cd6cbdb-vg9rl     1/1     Running   0          9s
redis-leader-58b566dc8b-2wc8p       1/1     Running   0          3m25s
```

4. Creating Redis follower service:

```
root@master:/home/ubuntu# vi redis-follower-service.yaml
root@master:/home/ubuntu# kubectl apply -f  redis-follower-service.yaml
service/redis-follower created
root@master:/home/ubuntu# kubectl get svc
NAME             TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)    AGE
kubernetes       ClusterIP   10.96.0.1       <none>        443/TCP    51m
redis-follower   ClusterIP   10.104.241.11   <none>        6379/TCP   7s
redis-leader     ClusterIP   10.100.216.61   <none>        6379/TCP   4m29s
```

5. Creating the Guest Book Frontend Deployment:

```
root@master:/home/ubuntu# vi frontend-deployment.yaml
root@master:/home/ubuntu# kubectl apply -f frontend-deployment.yaml
deployment.apps/frontend created
```

```
root@master:/home/ubuntu# kubectl get pods -l app=guestbook -l tier=frontend
NAME                        READY   STATUS    RESTARTS   AGE
frontend-697bd54cd4-c6ls6   1/1     Running   0          59s
frontend-697bd54cd4-wjj9b   1/1     Running   0          59s
frontend-697bd54cd4-xj8tt   1/1     Running   0          59s
```

6. Creating Frontend service:

```
apiVersion: v1
kind: Service
metadata:
  name: frontend
  labels:
    app: guestbook
    tier: frontend
spec:
  type: LoadBalancer
  type: LoadBalancer
  ports:
    # the port that this service should serve on
  - port: 80
  selector:
    app: guestbook
    tier: frontend
```

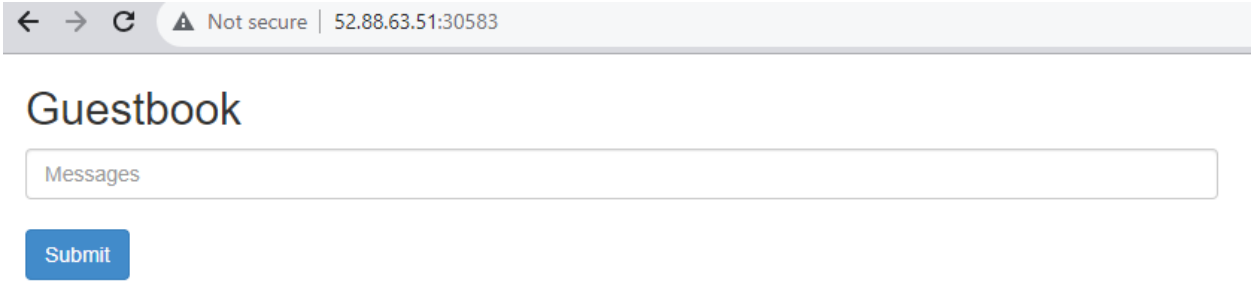
```
root@master:/home/ubuntu# vi frontend-service.yaml
root@master:/home/ubuntu# kubectl apply -f frontend-service.yaml
```

```
root@master:/home/ubuntu# kubectl get svc
NAME             TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)        AGE
frontend         NodePort    10.97.186.69    <none>        80:30583/TCP   5m55s
kubernetes       ClusterIP   10.96.0.1       <none>        443/TCP        61m
redis-follower   ClusterIP   10.104.241.11   <none>        6379/TCP       9m29s
redis-leader     ClusterIP   10.100.216.61   <none>        6379/TCP       13m
```

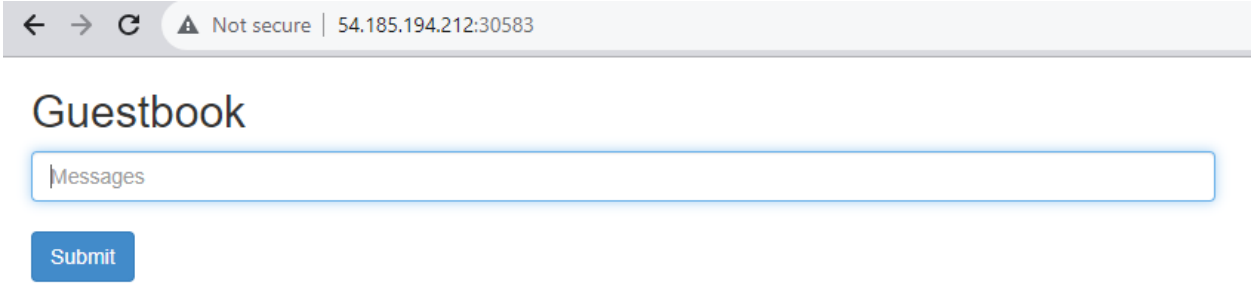
After deployment, open the respective port in the security group of Instances.

Now we can access the content of POD from the Public IP of Instances.

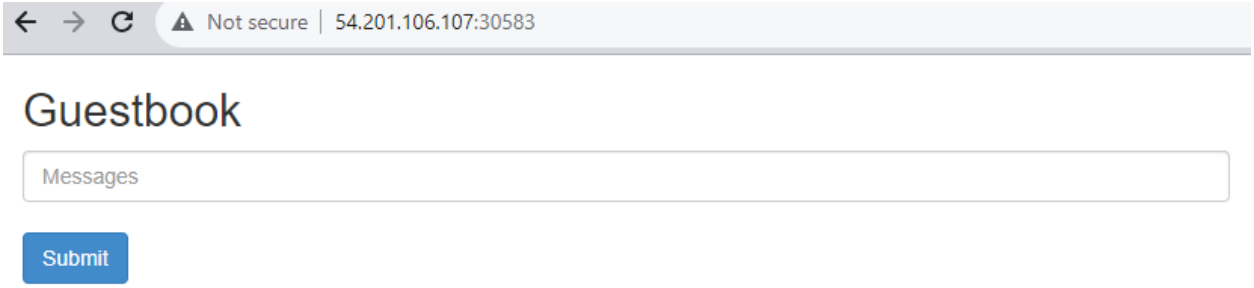
Master Node:



Worker 1:-


Not secure
54.185.194.212:30583
Guestbook
Messages
Submit

Worker 2:-


Not secure
54.201.106.107:30583
Guestbook
Messages
Submit