Deploying Stateless Application with Deployment Objects.

SSH to your AWS Workstation
ssh devops@<public-ip-addr> of your Workstation
Password is : **Dev0p$!!/**
Replace <your-name> with your name throughout the lab.

You can run an application by creating a Kubernetes Deployment object, and you can describe a Deployment in a YAML file.

1. Run the below commands on your AWS-Workstation

```
$ sudo su
# cd /home/devops
# mkdir application
# cd application/
# curl -f https://pastebin.com/raw/eTJTi1d0 > <your-name>-deployment.yaml
```

```
# vim <your-name>-deployment.yaml
```

Press 'i' to start the edit mode in the vim editor. Update <your-name> with your name.
Save and exit by pressing the **ECS key** and type **:wq** and press **enter** to exit.
Example

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: albert-deployment
spec:
  selector:
    matchLabels:
      app: albert-app
  replicas: 2 # tells deployment to run 2 pods matching the template
  template:
    metadata:
      labels:
        app: albert-app
    spec:
      containers:
      - name: albert-container
        image: nginx:latest
        ports:
        - containerPort: 80
```

2. Create a Deployment based on the YAML file:

```
# kubectl apply -f <your-name>-deployment.yaml
```

3. Display information about the Deployment:

```
# kubectl describe deployment <your-name>-deployment
```

4. Expose the Deployment with the below command.

```
# kubectl expose deployment <your-name>-deployment --type=NodePort
--name=<your-name>-service
```

```
root@ip-172-31-40-214:/home/devops/application# kubectl expose deployment albert-deployment --type=NodePort --name=albert-service
service/albert-service exposed
root@ip-172-31-40-214:/home/devops/application#
```

5. Check the **NODE** where your app has been deployed.

```
# kubectl get po -o wide
```

```
root@ip-172-31-40-214:/home/devops/application# kubectl get po -o wide
NAME                                 READY   STATUS    RESTARTS   AGE    IP           NODE                                          NOMINATED NODE
albert-deployment-66465fdb8d-7gw8z   1/1     Running   0          104s   10.48.0.13   gke-demo-default-pool-289f281e-lpk7   <none>
albert-deployment-66465fdb8d-pwlxl   1/1     Running   0          104s   10.48.0.14   gke-demo-default-pool-289f281e-lpk7   <none>
root@ip-172-31-40-214:/home/devops/application#
```

In this example the app has been deployed to the NODE **gke-demo-default-pool-289f281e-lpk7**

6. Check the **NODEPORT** on which the application has been exposed

```
# kubectl get svc <your-name>-service
```

```
root@ip-172-31-40-214:/home/devops/application# kubectl get svc albert-service
NAME             TYPE       CLUSTER-IP     EXTERNAL-IP   PORT(S)        AGE
albert-service   NodePort   10.51.248.48   <none>        80:31170/TCP   2m38s
root@ip-172-31-40-214:/home/devops/application#
```

In this example the **albert** application has been exposed on Port **31170** as shown above.

7. Get the Public IP of the GKE NODE by running the below command.

```
# kubectl get nodes -o wide
```

```
root@ip-172-31-40-214:/home/devops/application# kubectl get nodes -o wide
NAME                                 STATUS   ROLES    AGE   VERSION          INTERNAL-IP   EXTERNAL-IP    OS-IMAGE                             KERNE
L-VERSION   CONTAINER-RUNTIME
gke-demo-default-pool-289f281e-lpk7   Ready    <none>   30m   v1.12.6-gke.10   10.160.0.9    35.244.57.29   Container-Optimized OS from Google   4.14.
91+         docker://17.3.2
root@ip-172-31-40-214:/home/devops/application#
```

8. You can access the Application on the PUBLIC IP of the NODE where the POD has been deployed and the NodePort on which the POD is exposed.

As shown in the above example the application is deployed on NODE **gke-demo-default-pool-289f281e-lpk7** which has a public IP **35.244.57.29** and the deployment is exposed on NodePort **31170** as seen in step 6

Access the application from the public IP of the NODE and the NodePort as shown below

Example.
http://35.244.57.29:31170/