Run Dockerized JAVA App on K8 Cluster and exposing using NodePort

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.

1. Run the below commands on your AWS-Workstation.

```
$ sudo su
# cd application/
# curl -f https://pastebin.com/raw/Zv3GL67Z > java-app-<your-name>.yaml
```

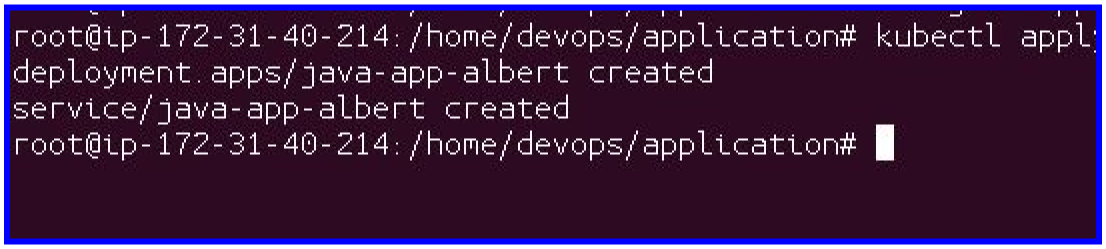
2. Edit the java-app-<your-name>.yaml file.

```
# vim java-app-<your-name>.yaml
```

Update the image: **lovescloud/java-app:latest** with your dockerhub image name that you uploaded to docker hub in docker lab Pushing images to docker Hub for java-app.
Replace <your-name> with your name in the above script.
Save and exit by pressing the ESC key and type :wq to save and quit by pressing enter

3. Run the below commands to deloy the JAVA application on your Kubernetes Cluster

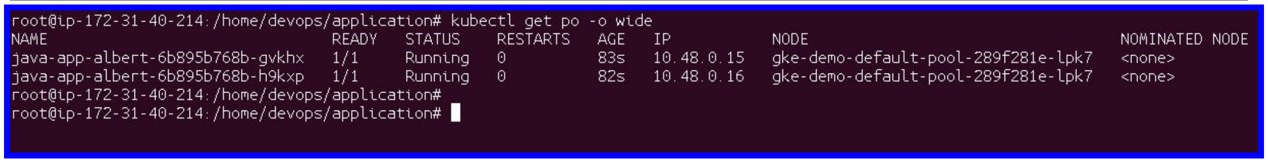
```
# kubectl apply -f java-app-<your-name>.yaml
```



```
root@ip-172-31-40-214:/home/devops/application# kubectl appl
deployment.apps/java-app-albert created
service/java-app-albert created
root@ip-172-31-40-214:/home/devops/application#
```

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

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

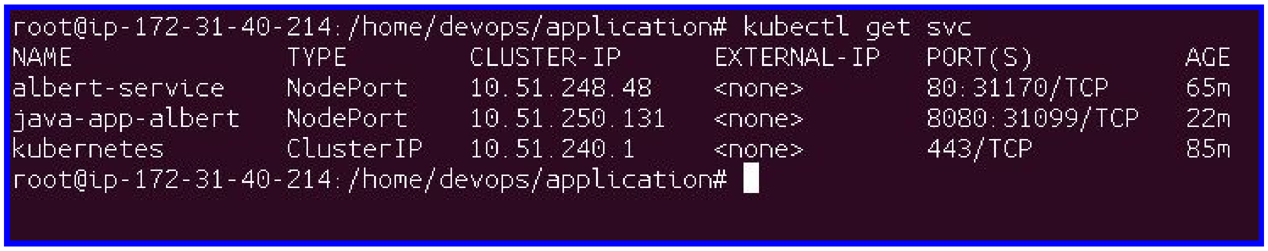


```
root@ip-172-31-40-214:/home/devops/application# kubectl get po -o wide
NAME                               READY   STATUS    RESTARTS   AGE   IP           NODE                                    NOMINATED NODE
java-app-albert-6b895b768b-gvkhx   1/1     Running   0          83s   10.48.0.15   gke-demo-default-pool-289f281e-lpk7     <none>
java-app-albert-6b895b768b-h9kxp   1/1     Running   0          82s   10.48.0.16   gke-demo-default-pool-289f281e-lpk7     <none>
root@ip-172-31-40-214:/home/devops/application#
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**

5. Check the NODEPORT on which the JAVA application has been exposed

```
# kubectl get svc
```

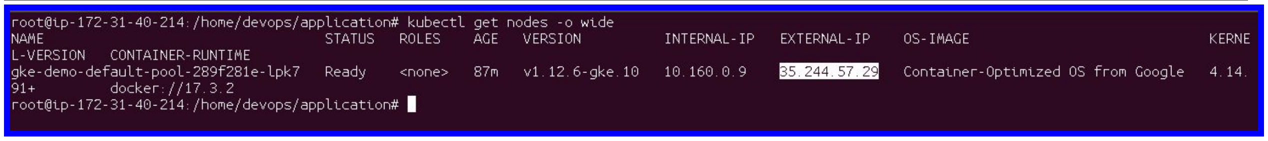


```
root@ip-172-31-40-214:/home/devops/application# kubectl get svc
NAME                TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)          AGE
albert-service      NodePort    10.51.248.48    <none>        80:31170/TCP     65m
java-app-albert     NodePort    10.51.250.131   <none>        8080:31099/TCP   22m
kubernetes          ClusterIP   10.51.240.1     <none>        443/TCP          85m
root@ip-172-31-40-214:/home/devops/application# 
```

In this example the **albert java application** has been exposed on port **31099** as shown above.

6. Get the Public IP of the NODE where the POD has been deployed.

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

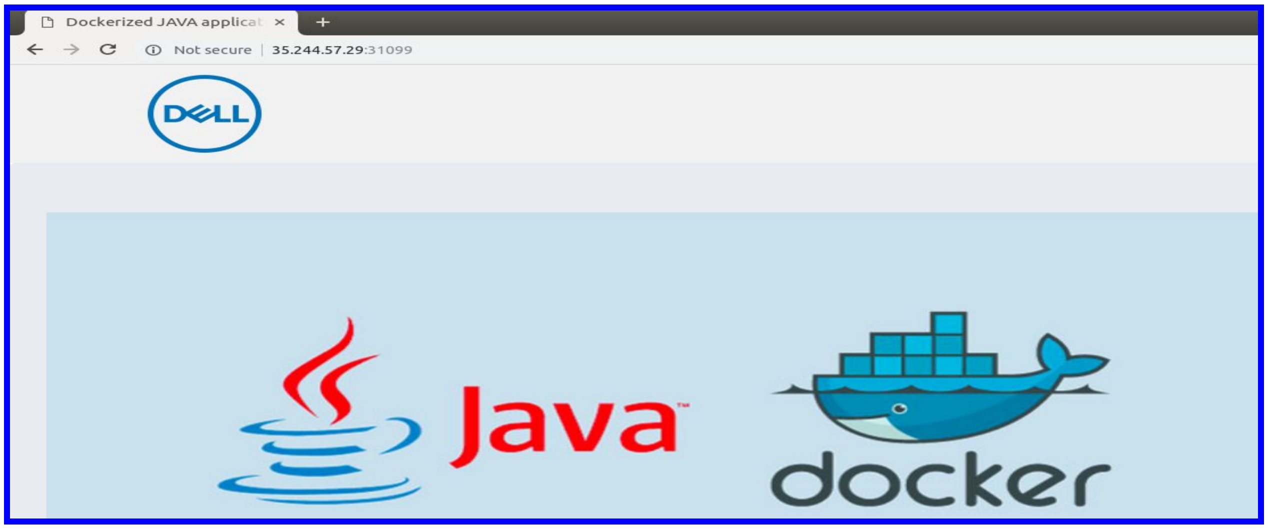


```
root@ip-172-31-40-214:/home/devops/application# kubectl get nodes -o wide
NAME                                   STATUS   ROLES    AGE   VERSION          INTERNAL-IP   EXTERNAL-IP    OS-IMAGE                             KERNE
L-VERSION   CONTAINER-RUNTIME
gke-demo-default-pool-289f281e-lpk7    Ready    <none>   87m   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# 
```

The pubic associated to the NODE **gke-demo-default-pool-289f281e-lpk7**
is **35.244.57.29 as shown in the above screenshot, and the NodePort on which the application is exposed is 31099 as shown in step 5 example.**

7. Access the Application from the PublicIP of NODE and the NodePort as shown below.

http://<NODE-PUBLIC-IP>:NODEPORT
http://35.244.57.29:31099/


Dockerized JAVA applicat
Not secure
35.244.57.29:31099
DELL
Java
docker