Deploying Stateless Application with Deployment Objects.

Kubectl - kubectl controls the Kubernetes cluster manager

You can run an application by creating a Kubernetes Deployment object, and you can describe a Deployment in a YAML file. For example, this YAML file describes a Deployment that runs the **<your-docker-hub-username>/dell:v1** Docker image

Replace **<your-docker-hub-username>/dell:v1** with the image name,repository name and the tag from your **docker hub** account that you pushed to docker hub in **docker labs**

1. SSH to your AWS Instance and make a dir /home/<your-user-name>/application

```
$ cd /home/devops
$ sudo mkdir application
$ cd application/
```

Create a New Deployment file starting with your name.
1. $ curl -k https://pastebin.com/raw/krKi6xsh > <your-name>-deployment.yam

$ vim <your-name>-deployment.yaml

#Note : press 'i' to start the edit mode in the vim editor. Replace <your-name> with your name and then save and exit by (:wq (enter))

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**

The output is similar to this:

```
Name:     <your-name>-deployment
Namespace:    default
CreationTimestamp:  Tue, 30 Aug 2016 18:11:37 -0700
Labels:     app=dell
Annotations:    deployment.kubernetes.io/revision=1
Selector:   app=dell
Replicas:   2 desired | 2 updated | 2 total | 2 available | 0 unavailable
StrategyType:   RollingUpdate
```

```
MinReadySeconds:  0
RollingUpdateStrategy:  1 max unavailable, 1 max surge
Pod Template:
Labels:       app=dell
Containers:
nginx:
 Image:             asyed755/dell:v1
Port:                  80/TCP
Environment:    <none>
Mounts:           <none>
Volumes:          <none>
Conditions:
Type Status  Reason
----         ------  ------
Available     True    MinimumReplicasAvailable
Progressing   True    NewReplicaSetAvailable
OldReplicaSets:   <none>
NewReplicaSet:    nginx-deployment-1771418926 (2/2 replicas created)
No events.
```

4. List the pods created by the deployment:

$ kubectl get pods -l app=<your-app-name>

The output is similar to this:

```
NAME                              READY    STATUS    RESTARTS   AGE
dell-deployment-1471416983-7o5ac   1/1      Running   0          16h
dell-deployment-1541148254-318ad   1/1      Running   0          16h
```

5. To display information about a pod:

$ kubectl describe pod <pod-name>

6. Expose the Deployment with the below command.

$ kubectl expose deployment <your-app-name> --type=NodePort --name=<your-service-name>

7. Check the Node details where the POD has been deployed

$ kubectl get pod -o wide | grep <your-app-name>

```
NAME                        READY   STATUS    RESTARTS   AGE   IP             NODE
arshad-5fdff48b48-7r4pg      1/1     Running   0          2m    100.96.4.43   ip-172-20-55-125.ec2.internal
arshad-5fdff48b48-gg7j4      1/1     Running   0          2m    100.96.4.44   ip-172-20-55-125.ec2.internal
rajni-deployment-d667        1/1     Running   0          3h    100.96.4.13   ip-172-20-55-125.ec2.internal
```

ip-172-20-55-125.ec2.internal is the Worker node where the app has been deployed

8. Login to the **AWS Console** and Copy the **Public IP** of the NODE where the POD has been deployed to access the application.

Access the application as shown below

http://<node-public-ip>:NodePort ##see step 5 to check the port

Example

http://18.208.206.161:32587/s