Day 3

Lab - Building your custom Docker Image

```
cd ~/openshift-tekton-june2024
git pull
cd Day3/spring-ms
docker build -t hello:1.0 .
docker images
```

Lab - Rolling update

```
- Assume you have your microservice version 2.0 in the openshift cluster
and you wish to upgrade to version 3.0 without any downtime
- in such cases, you can go for rolling update
```

Let's first deploy the sample spring-boot microservice as shown below

```
oc new-project jegan
oc create deployment hello --image=tektutor/hello:2.0 --replicas=3
oc get deploy,rs,po
```

Now you may check the version of container our hello pod are using as shown below

```
oc get pod/hello-5c886db5d8-72hln -o yaml | grep tektutor/hello
```

Now, let's upgrade our microservice version from 2.0 to 3.0 as shown below

```
oc set image deploy/hello hello=hello hello=tektutor/hello:3.0
```

Now you may chek the version of the container our hello pods are using as shown below

```
oc get pod/hello-779467b6bf-g8mgm -o yaml | grep tektutor/hello
```

You can check the rolling update status as shown below

```

```

```
oc rollout status deploy/hello
oc rollout history deploy/hello
```

You wish to rollback to previous version of image

```
oc rollout undo deploy/hello
```

Lab - Creating an internal service for hello deployment

```
- ClusterIP Service is accessible only with the Openshift cluster
- the practical use-case clusterIP internal is for database
- the service can be accessed using the service name or service IP address
```

```
oc expose deploy/hello --type=ClusterIP --port=8080
oc get services
oc get service
oc get svc
oc describe svc/hello
```

To access the clusterip service, we need get inside some pod running in the cluster

```
oc rsh deploy/hello
curl http://hello:8080
curl http://<service-ip>:<service-port>
exit
```

Lab - Creating an external NodePort service for hello deployment

First we need to delete the existing clusterip service, before we can create a nodeport service with the same name

```
oc delete svc/hello
```

Openshift has reservers port in the range 30000-32767 for the purpose of nodeport services. Openshift randomly assigns one of the ports that is available on the nodes in the openshift cluster.

Let's create the external nodeport service for hello deployment

```

```

```
oc get deploy -l app=hello
oc expose deploy/hello --type=NodePort --port=8080
oc get svc
oc describe svc/hello
```

Accessing the service from outside the openshift cluster

```
curl http://<any-node-ip>:<nodeport>
curl http://<any-node-hostname>:<nodeport>
curl http://master-2.ocp4.tektutor.org.labs:30106
curl http://master-3.ocp4.tektutor.org.labs:30106
```

Service discovery works in all type of services. The service discovery works within the scope of the openshift cluster(pod shell).

Lab - Creating an external loadbalancer service

For detailed instructions you can refer my medium blog

```
https://medium.com/tektutor/using-metallb-loadbalancer-with-bare-metal-
openshift-onprem-4230944bfa35
```

Things to note

```
- load balancer is used normally in public cloud like AWS, Azure, GCP,
Digital ocean, etc
- the public cloud has a load balancer service
  Example
  - AWS supports External Load balancer (ELB)
  - AWS also supports Application Load Balancer (ALB)
```

- In case, your Openshift cluster is a managed service running in AWS, then when we create a loadbalancer service for our application deployments, it creates an application loadbalancer within AWS, the AWS load balancer will perform load-balance of our pods
- In this case, the kube-proxy will not into picture, as we prefer an external load-balancer provisioned in AWS/Azure/GCP
- Hence, there will charge from public cloud vendor i.e AWS/Azure/GCP for the external load-balancer

Let's delete the nodeport service, in order to create the loadbalancer service with the same name

```
oc delete svc/hello
```

Let's create the loadbalancer service

```
oc get deploy -l app=hello
oc expose deploy/hello --type=LoadBalancer --port=8080
oc get svc
oc describe svc
oc get endpoints
```

For this to work, we need to install Metallb operator and configure it

```
cd ~/openshift-tekton-june2024
git pull
cd Day3/metallb
oc apply -f address-pool.yml
oc apply -f metallb.yml
```

Once you are sure all the pods under the metallb-system namespace are running, you can check the hello server for external ip

```
oc get svc
curl http://192.168.122.120:8080
```