Task4

February 6, 2021

Task4: Deploying an Application
Change directory to the Project folder present on the Desktop

```
cd /home/rhyme/Desktop/Project
```

Clone the application repository from Github

```
git clone "https://github.com/alihussainia/Deploying-Web-Apps-on-a-Kubernetes-Single-Node-Clust
```

Change the name of the app directory to simple_app

```
mv Deploying-Web-Apps-on-a-Kubernetes-Single-Node-Cluster-using-Minikube simple_app
```

Then move into the app folder using:

```
cd simple_app
```

0.0.1 Application Structure

In this github repository there are files for a simple application needed to build a *Docker* image and run it on *Kubernetes*.

NOTE: Below is just the tree not code so don't run it on terminal

```
- Repository
  - html/
      - index.html
  - kubernetes/
      - deployment.yaml
      - service.yaml
  - Dockerfile
```

0.0.2 Dockerfile

By looking at the `Dockerfile` we will see that it is very basic. Using an *Alpine* version of *Nginx* to serve a single page of static content.

NOTE: Below is just the content not code so don't run it on terminal

```
FROM nginx:1.15.0-alpine
COPY html/ /usr/share/nginx/html/
```

0.0.3 Docker Image

Let's first build the docker image using the docker daemon provided by the minikube

```
eval $(minikube docker-env)
```

The whole CMD should be used including the "."

```
docker build -t simple-app:v1 .
```

Let's test the image, first by running the container

```
docker run -d -p 8000:80 simple-app:v1
```

Then calling the minikube ip address with port 8000

```
curl `minikube ip`:8000
```

0.0.4 *Kubernetes* manifests

The `deployment.yaml` manifest describes to *Kubernetes* how to run the *Docker* container inside a *Pod*.

NOTE: This is just the content not code, so don't run it on terminal

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: simple-app
spec:
  selector:
    matchLabels:
      app: simple-app
  replicas: 2
  template:
    metadata:
      labels:
        app: simple-app
    spec:
      containers:
      - name: simple-app
        image: simple-app:v1
        ports:
        - containerPort: 80
```

The `service.yaml` manifest describes to *Kubernetes* how to make the *Pods* available on the network by using a *Service* resource type

```
kind: Service
apiVersion: v1
metadata:
```

```
  name: simple-app
spec:
  selector:
    app: simple-app
  type: NodePort
  ports:
  - protocol: TCP
    port: 80
    targetPort: 80
```

0.0.5 Deployment

lets deploy our application to *Minikube*.

```
kubectl -n default create -f kubernetes
```

0.0.6 Pod Check

```
kubectl -n default get pod
```

We can see that we have 2 *Pods* running. This is because we asked for 2 `replicas` inside the `deployment.yaml`.

0.0.7 Service Check

```
kubectl -n default get service
```

NodePort makes the mapped container port directly available on the *Kubernetes* node.

0.0.8 Access Test

let's test that we can access the application running inside *Minikube*

```
curl `minikube service -n default simple-app --url`
```

0.0.9 Dashboard

Make sure that you are looking at the `default` namespace and then click on *Deployments*.

```
minikube dashboard
```