Lab – Microservices

Note that this lab will require that you have an Ubuntu installed PC. If you're using Windows or any other operating system, make sure that you have Docker, Kubectl and minikube installed successfully. You may refer the following web URLs to install those tools/frameworks.

https://docs.docker.com/desktop/install/windows-install/

https://minikube.sigs.k8s.io/docs/start/

1. On your Ubuntu PC, open a terminal window and install docker. You may follow the instructions given in the following documentation.

https://docs.docker.com/engine/install/ubuntu/

2. Install minikube using the following command. Minikube runs a single node Kubernetes cluster in your local PC.

```
curl -LO https://storage.googleapis.com/minikube/releases/latest/minikube-linux-amd64

sudo install minikube-linux-amd64 /usr/local/bin/minikube
```

3. Download kubectl, which is a command line tool using which you can access your minikube cluster using the following command.

```
snap install kubectl -classic
```

4. Start minikube with the following command.

```
minikube start
```

5. Start the minikube dashboard using the following command. It lets you monitor the minikube cluster from your browser.

```
minikube dashboard
```

Now let us build a Spring Boot application using Docker and deploy in the local minikube cluster.

6. Extract the contents of the attached Spring Boot application. Create a file called Dockerfile in its root directory and add the following text to that file.

```
FROM openjdk:8-jdk-alpine
EXPOSE 8080
ARG JAR_FILE=target/sample-1.0-SNAPSHOT.jar
ADD ${JAR_FILE} app.jar
ENTRYPOINT ["java","-jar","/app.jar"]
```

7. Open a terminal window and run the following command to make minikube work with your local Docker application.

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

8. Build the docker image using the following command.

```
docker build -t springboot-kubernetes:1.0 .
```

9. Run the following command to deploy the docker image in minikube.

```
kubectl create deployment springboot-kubernetes --image=springboot-
kubernetes:1.0 --port=8080
```

10. Use the following command to check the pods that are up and running. Each pod may contain a certain application.

```
kubectl get pods
```

11. Use the following command to view the logs. You should see that the newly built application is started at port 8080.

```
 kubectl logs <pod_name>
```

12. Get the name of the deployment using the following command.

```
kubectl get deployments
```

13. Now we need to create a service based on the deployed application. Use the following command to create a service based on the application.

```
kubectl expose deployment springboot-kubernetes --type=NodePort
```

14. You can view the currently available services by using the following command.

```
kubectl get services
```

15. Get the service url to access the service using the following command.

```
minikube service springboot-kubernetes
```

16. Go to the URL shown in the command window. You may see an error message.

17. Now append */api/user* to the URL and try again. If you see the phrase "list user called successfully", you have successfully deployed your first Spring Boot application using Docker and Kubernetes. You may use a REST client (e.g. SoapUI, Postman or RESTClient browser plugin) to try out other HTTP commands such as POST and DELETE using the same URL.

As an optional exercise, now try to deploy any other application (developed using Spring Boot or any other technology) on minikube following similar set of steps.

If you have access to a commercial cloud such as Google Cloud or Microsoft Azure cloud, you may try to deploy an application in a cloud-based Kubernetes cluster. Following is an tutorial on how to deploy a Kubernetes application in Microsoft Azure cloud.

https://learn.microsoft.com/en-us/azure/aks/tutorial-kubernetes-deploy-application?tabs=azure-cli