What you need:

1. Google cloud account

Login to console: <https://console.cloud.google.com/>

1. Docker
   1. Docker hub account:

Login to dashboard <https://hub.docker.com/>

* 1. Docker Desktop

1. Terminal and, :
   1. Homebrew

Enter this command on terminal to install brew

/usr/bin/ruby -e "$(curl -fsSL <https://raw.githubusercontent.com/Homebrew/install/master/install>)"

NB: You should have Xcode installed first

Can be installed with Homebrew:

* 1. java
  2. maven
  3. docker
  4. kubectl
  5. minikube

NB: for pushing and pulling process it is required to use Mobile data instead if WiFi to avoid firewall bockages or EOF error…

When deploying to Kubernetes always check pods logs if ports are like the one exposed by the applications, if not fix them

# Deploying the app locally (Docker):

Your App

Local

**Image** Create an image

docker login -u "<username> " -p "<password>" docker.io

docker build -t <image-name> .

**Repository** Push image to dockerhub

docker commit <existing-container> <username/existing-repository>

docker image tag <existing-image> <username/existing-repository>

docker push <username/existing-repository>

**Container** Create and run a container

docker container run -d -p <port-number> --name <container-name> <existing-image>

*or*

docker container create -i -t -p <port-number> --name <container-name> <existing-image>

docker container start --attach -i <container-name>

docker stop <existing-container >

1. Deploy your app to the docker local server and run it using docker
   1. Create a Dockerfile (without extension, case sensitive) and place it in the project directory

FROM openjdk:8-jdk-alpine

EXPOSE 9090

ADD ./target/your\_war\_filename.war your\_war\_filename.war

ENTRYPOINT ["java", "-jar", "/ your\_war\_filename.war "]

Pom.xml should be configured to create a war file:

<packaging>war</packaging> <!--for war packaging, default packaging is jar if not mentioned-->

<dependencies>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-tomcat</artifactId>

<scope>provided</scope> <!--for war packaging-->

</dependency>

</dependencies>

<build>

<finalName> your\_war\_filename</finalName>

</build>

* 1. Create an image
  2. Create a container

1. Push to docker hub repository
   1. Create repository in docker hub
   2. Push to the docker hub repository

EOF error may be due to network… Recommended: Use mobile data for pushing and pulling tasks

# Deploying the app internally or externally

Start minikube

$ minikube start

Recommended: Use mobile data for pushing and pulling tasks included during the process of minikube initiation.

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

$ kubectl apply -f name\_of\_your\_yamlfile.yaml

Check you pods are running:

$ kubectl get pods

Check log of your pods:

$ kubectl logs <pod-name>

View services:

$ kubectl get svc

Run service:

$ minikube service <existing-service> --url

YAML file:

apiVersion: apps/v1

kind: Deployment

metadata:

name: docker-testing-non-jpa-k8s-deployment

labels:

app: docker-testing-non-jpa

spec:

replicas: 3

selector:

matchLabels:

app: docker-testing-non-jpa

template:

metadata:

labels:

app: docker-testing-non-jpa

spec:

containers:

- name: springboot-testing

image: shologuanda/docker-testing-non-jpa:latest

ports:

- containerPort: 9090

---

apiVersion: v1 # Kubernetes API version

kind: Service # Kubernetes resource kind we are creating

metadata: # Metadata of the resource kind we are creating

name: docker-testing-non-jpa-k8s-svc

spec:

selector:

app: docker-testing-non-jpa

ports:

- protocol: "TCP"

port: 80 # The port that the service is running on in the cluster

targetPort: 9090 # The port exposed by the service

type: NodePort # type of the service.