# Deploying the app locally (Docker):

The same steps used in the previous tutorial are to be used. The difference here is that you have to treat mysql and your springboot app as separately.

* Create mysql image (pull from docker hub)
* Create mysql container
* Create image for your springboot app
* Create container for your spring boot app

NB: You cannot connect to mysql container from your springboot app running on your local machine. You can only connect to mysql running on your local machine with ‘localhost’. Your app should run on docker to connect to mysql running on docker. Meaning, create a container for mysql and a container for your app. The way your app should connect to mysql container is by replacecing ‘localhost’ on datasource url by the name of mysql container. Unfortunatley, you cannot do that while your app is running in your local machine.

This is to say:

Your app when running on local machine mysql should connect to localhost (or to local mysql)

spring.datasource.url=jdbc:mysql://localhost/database

Your app when running on docker (container) should connect to mysql container (docker machiner)

spring.datasource.url=jdbc:mysql://mysql\_container\_name/database

You should therefore use dynamic values. E.g:

spring.datasource.url=jdbc:mysql://${DB\_HOST}/${DB\_NAME}

spring.datasource.username=${DB\_USERNAME}

spring.datasource.password=${DB\_PASSWORD}

On your local machine you can set these values using environment variables

On mac set environment variables by editing .zshrc file found in /User/${name}. This is a hidden file. To view hidden files use shortcut ‘**commant+shift+.**’ . At the end of the file put:

export DB\_HOST="localhost"

export DB\_NAME="your\_database"

export DB\_USERNAME="your\_username"

export DB\_PASSWORD="your\_password"

on your docker machine your can set these values during app deployment. Create an image for your app (docker build -t image-name) and mysql (docker pull mysql:5.7)

create network

docker network create network-name

run your app and mysql on this network (same network) and pass the dynamic values

docker run --name mysqldb --hostname mysqldb --network network-name -e MYSQL\_ROOT\_PASSWORD=your\_password -e DB\_NAME=your\_database -e DB\_USERNAME=your\_username -e DB\_PASSWORD= your\_password -e MYSQL\_HOST=mysql -e DB\_HOST=mysqldb -p 3306:3306 -d mysql:5.7

docker run --network network-name --name container-name -p 9090:9090 -d image-name

Alternatively run **docker compose up**

# Using docker composer…

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)

$ docker compose up

# Deploying the app internally or externally (kubernetes)

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>

See more about a pod:

$ kubectl describe pod <pod-name>

View services:

$ kubectl get svc

Run service:

$ minikube service <existing-service> --url