Practical 3 Ex2

Dawid Skraba 19433692

1. I installed Kompose via homebrew..

```
dawidskraba@dhcp-892b1ae8 ex2 % brew install kompose
Running `brew update --auto-update`...
==> Auto-updated Homebrew!
```

2. A) Below I used the docker build command to build the image for web service. I then push this image to docker hub to its repository. This is done so we can pull its image from the hub when we are building the deployment.

```
dawidskraba@dhcp-892b1ae8 ex2 % docker build -t ex2app .
[+] Building 17.3s (17/17) FINISHED
 => [internal] load build definition from Dockerfile
 => => transferring dockerfile: 325B
```

```
dawidskraba@dhcp-892b1ae8 ex2 % docker images
REPOSITORY                  TAG         IMAGE ID       CREATED          SIZE
ex2app                      latest      d2f7fa4d1abb   12 seconds ago   179MB
```

```
dawidskraba@dhcp-892b1ae8 ex2 % docker image push dawidskra007/ex2app
Using default tag: latest
The push refers to repository [docker.io/dawidskra007/ex2app]
d6518127eaa9: Pushed
f2901dcb80c1: Pushed
```

b) c) I updated the docker compose file by adding the image tag with our image name, added the restart policy and added ports to the redis service using its default port. This is done as we are later translating this file to Kubernetes and ports need to be specified as its going to be put into a container and then a pod. The restart policy ensures that the pod whenever it fails it will automatically restart.

```yaml
version: "3.9"
services:
  web:
    image: dawidskra007/ex2app
    ports:
      - "8000:5000"
    restart: always
  redis:
    image: "redis:alpine"
    ports:
      - "6379:6379"
```

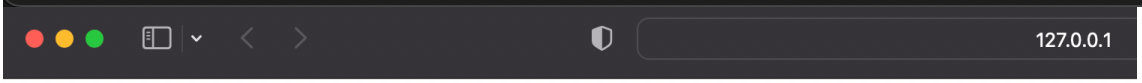
3.

```
dawidskraba@dhcp-892b1ae8 ex2 % kompose convert -f docker-compose.yml
  INFO Kubernetes file "redis-service.yaml" created
  INFO Kubernetes file "web-service.yaml" created
  INFO Kubernetes file "redis-deployment.yaml" created
  INFO Kubernetes file "web-deployment.yaml" created
dawidskraba@dhcp-892b1ae8 ex2 % kubectl apply -f .
deployment.apps/redis configured
service/redis configured
deployment.apps/web configured
service/web configured
dawidskraba@dhcp-892b1ae8 ~ % kubectl get pods
NAME                     READY   STATUS    RESTARTS   AGE
redis-7cc5f6d9bb-tlzrv   1/1     Running   0          85s
web-599d5cf886-whjkm     1/1     Running   0          85s
dawidskraba@dhcp-892b1ae8 ~ % kubectl get deployments
NAME    READY   UP-TO-DATE   AVAILABLE   AGE
redis   1/1     1            1           94s
web     1/1     1            1           94s
```

We used the Kompose command to translate this docker compose file to a Kubernetes file. Following the commands we first convert the file and then using this translated file we create our deployment using the apply command. This creates two pods with our services. Pods can communicate with each other even if they are on different nodes. We see this creates two deployments, this is because of the translation from docker-compose.

4.

```
dawidskraba@dhcp-892b1ae8 ~ % kubectl get services
NAME         TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)    AGE
kubernetes   ClusterIP   10.96.0.1       <none>        443/TCP    2d14h
redis        ClusterIP   10.105.40.15    <none>        6379/TCP   6m46s
web          ClusterIP   10.100.69.54    <none>        8000/TCP   6m45s
dawidskraba@dhcp-892b1ae8 ~ % minikube service web
|-----------|------|-------------|--------------|
| NAMESPACE | NAME | TARGET PORT |     URL      |
|-----------|------|-------------|--------------|
| default   | web  |             | No node port |
|-----------|------|-------------|--------------|
😿  service default/web has no node port
🏃  Starting tunnel for service web.
|-----------|------|-------------|------------------------|
| NAMESPACE | NAME | TARGET PORT |          URL           |
|-----------|------|-------------|------------------------|
| default   | web  |             | http://127.0.0.1:63792 |
|-----------|------|-------------|------------------------|
🎉  Opening service default/web in default browser...
❗  Because you are using a Docker driver on darwin, the terminal needs to be open to run it.
```



Hello World! I have been seen 7 times.

Here we access our web service and we can see that we get the expected result. We have successfully moved our two containers to a Kubernetes cluster. And tested that our containers can communicate with each other. We could access our service using the minikube service command as it returns a URL to connect to said service. By

default pods are only accessible internally, but when they are declared as a service they can be accessed from outside the Kubernetes virtual network.