

Exercise Two - Converting the Docker-compose flow to Kubernetes (3%)

Before we start exercise 2 first, we delete all previous deployments and make sure that no pods are running on minikube cluster. To do this I use the command:

Kubectl delete deployment hello-minikube

Task 1 (0.5%): Install Kompose. This is the tool that we will use to convert the docker-compose file to Kubernetes.

I installed Kompose by running the command in cmd:

curl -L https://github.com/kubernetes/kompose/releases/download/v1.26.0/kompose-windows-amd64.exe -o kompose.exe

This installed kompose on my laptop.


```
C:\Users\waleed\Desktop\PRACTICAL3 waleed_wazir_19396951\ex2>curl -L https://github.com/kubernetes/kompose/releases/download/v1.26.0/kompose-windows-amd64.exe -o kompose.exe
% Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
           Dload  Upload   Total   Spent    Left   Speed
0         0     0     0     0      0     0      0      0      0      0      0      0
100 21.3M  100 21.3M    0     0  9084k      0  0:00:02  0:00:02 --:--:-- 11.2M
```


Task 2 (0.5%): Use the source code from the first Docker tutorial (<https://docs.docker.com/compose/gettingstarted/>). You have to make some adjustments to the docker-compose file. You should make the following changes:


I followed the docker tutorial and created 4 files called app.py, dockerfile, requirements and docker-compose.yml and put the files into a folder called building_image which was in the ex2 folder.


I then ran the command docker compose up to build the image and run the code:

docker push waleeducd/getstarted2.0:v1 - this Pushes an image or a repository to a registry



 waleeducd / getstarted2.0

Description
This repository does not have a description 

 Last pushed: 2 hours ago

Tags and scans  VULNERABILITY SCANNING - DISABLED [Enable](#)

This repository contains 1 tag(s).

Tag	OS	Pulled	Pushed
 v1		--	2 hours ago

[See all](#) [Go to Advanced Image Management](#)

After publishing the image to my repository, I went to the folder “building image” and copied and pasted the docker-compose.yml file it into “ex2” folder, I then opened up that file up and change the following:

```
docker-compose.yml C:\...ex2  docker-compose.yml C:\...building image X
C: > Users > waleed > Desktop > PRACTICAL3_waleed_wazir_19396951 > ex2 > building image > docker-compose.yml
1  version: "3.9"
2  services:
3    web:
4      build: .
5      ports:
6        - "8000:5000"
7    redis:
8      image: "redis:alpine"

docker-compose.yml C:\...ex2 X  docker-compose.yml C:\...building image
C: > Users > waleed > Desktop > PRACTICAL3_waleed_wazir_19396951 > ex2 > docker-compose.yml
1  version: "3.7"
2  services:
3    web:
4      image: waleeducd/getstarted2.0:v1
5      ports:
6        - "8000:5000"
7      restart: always
8    redis:
9      image: redis:alpine
10     ports:
11       - "6379:6379"
```

version: "3.9" to "3.7", build: to image: and set image to waleeducd/getstarted and added the restart: policy set as always and I also defined the ports for redis service as you can see from the images.

Task 3 (1%): Convert the docker-compose file to Kubernetes, and create a deployment to your cluster

Now that I have changed the docker compose file I then convert the file to kubernetes and created deployments for our cluster using the command:

Kompose convert

This converts the docker-compose.yml file to files that you can use with kubectl

```
C:\Users\waleed\Desktop\PRACTICAL3_waleed_wazir_19396951\ex2>kompose convert
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
```

I then added the deployments using the command **kubectl apply -f redis-service.yaml,web-service.yaml,redis-deployment.yaml,web-deployment.yaml**

This Applies a configuration to a resource

```
C:\Users\waleed\Desktop\PRACTICAL3_waleed_wazir_19396951\ex2>kubectl apply -f redis-service.yaml,web-service.yaml,redis-deployment.yaml,web-deployment.yaml
service/redis configured
service/web configured
deployment.apps/redis configured
deployment.apps/web configured
```

Then using the command **minikube service name** this will automatically open our browser to the nodes proxy address.

I first ran **minikube service redis**

```
C:\Users\waleed\Desktop\PRACTICAL3_waleed_wazir_19396951\ex2>minikube service redis
|-----|-----|-----|-----|
| NAMESPACE | NAME | TARGET PORT | URL |
|-----|-----|-----|-----|
| default | redis | | No node port |
|-----|-----|-----|-----|
* service default/redis has no node port
* Starting tunnel for service redis.
|-----|-----|-----|-----|
| NAMESPACE | NAME | TARGET PORT | URL |
|-----|-----|-----|-----|
| default | redis | | http://127.0.0.1:52288 |
|-----|-----|-----|-----|
* Opening service default/redis in default browser...
! Because you are using a Docker driver on windows, the terminal needs to be open to run it.
* Stopping tunnel for service redis.
```

Then I ran the command **minikube service web** and a browser opened up similar to, the same output with the one you saw when you used the docker-compose.

```
C:\Users\waleed\Desktop\PRACTICAL3_waleed_wazir_19396951\ex2>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://127.0.0.1:52470

```
* Opening service default/web in default browser...
! Because you are using a Docker driver on windows, the terminal needs to be open to run it.
```

← → ↻ ⓘ 127.0.0.1:52470

Hello Waleed! You've visited me 3 times.