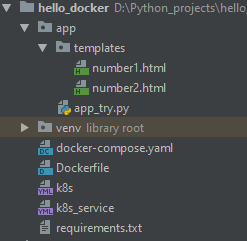
**FROM PYTHON TO MINIKUBE**

**PROJECT STRUCTURE**



**DOCKER**

*Dockerfile (don’t use an extension)*

FROM python:3.7 # your version  
  
WORKDIR /app  
  
COPY requirements.txt .  
  
RUN pip install Flask RandomWords # required libraries  
RUN pip install -r requirements.txt  
  
COPY /app . # the files needed to run your app  
  
EXPOSE 5000 # port number  
  
CMD ["python", "app\_try.py"]

*Docker-compose.yaml*

version: "3.7" # python version  
services:  
 python:  
 build:  
 context: .  
 container\_name: python  
 image: aimvector/python:1.0.0  
 ports:  
 - 5000:5000

*Requirements.txt*

Flask==1.1.2

Python Terminal => docker build –t hi-docker

Terminal => docker run -p 5000:5000 hi-docker

Alternatively:

Python Terminal => docker-compose build python

Python Terminal => docker-compose up python

**KUBERNETES**

--- Login DockerHub account ---

Python Terminal => docker login --username=user123

Password:\*\*\*\*

Login Succeeded

--- Tag your image ---

Python Terminal => docker tag hi-docker user123/hi-docker:latest

-- Push your image to DockerHub --

Python Terminal => docker push user123/hi-docker

-- After pushed –

Terminal => minikube start

-- Check status –

Terminal => minikube status

-- Check for deployment –

Terminal => kubectl get nodes

*k8s.yaml*

apiVersion: apps/v1  
kind: Deployment  
metadata:  
 name: hello-docker  
spec:  
 selector:  
 matchLabels:  
 app: hello-docker  
 replicas: 2 *# tells deployment to run 2 pods matching the template* strategy:  
 type: RollingUpdate  
 rollingUpdate:  
 maxSurge: 1  
 maxUnavailable: 0  
 template:  
 metadata:  
 labels:  
 app: hello-docker  
 spec:  
 containers:  
 - name: hello-docker  
 image: user123/hi-docker:latest  
 imagePullPolicy: Always  
 ports:  
 - containerPort: 5000

Python Terminal => kubectl apply -f k8s

-- Check its ready or not --

Terminal => kubectl get deployments

Terminal => kubectl get pods

-- When its ready --

k8s\_service.yaml

apiVersion: v1  
kind: Service  
metadata:  
 name: hello-docker  
spec:  
 type: LoadBalancer  
 selector:  
 app: hello-docker  
 ports:  
 - protocol: TCP  
 port: 5000  
 targetPort: 5000

Python Terminal => kubectl apply -f k8s\_service

Terminal => kubectl get svc

-- If external IP is “localhost”, its done! --

-- If external IP is “pending” --

Run another terminal as administrator => minikube tunnel

-- When it starts to tunnel --

Terminal => kubectl get svc

-- Now you can test your service locally with the given external IP and port number.

References:

<https://www.youtube.com/watch?v=BIM4W_c1kKc&list=LL&index=2&t=597s&ab_channel=CloudDev.Engineering>

<https://www.youtube.com/watch?v=d1ZMnV4yM1U&list=LL&index=4&ab_channel=ThatDevOpsGuy>

<https://www.youtube.com/watch?v=dVEjSmKFUVI&list=LL&index=3&t=1143s&ab_channel=JulianNash>