Docker ORCHESTRATION

LAB: PYTHON WEB SERVER

In this lab we will configure a custom network and startup a simple python3 based web server.

# Step 1: Build Network

The default network used by Docker is the bridge network.

Create your own managed network:

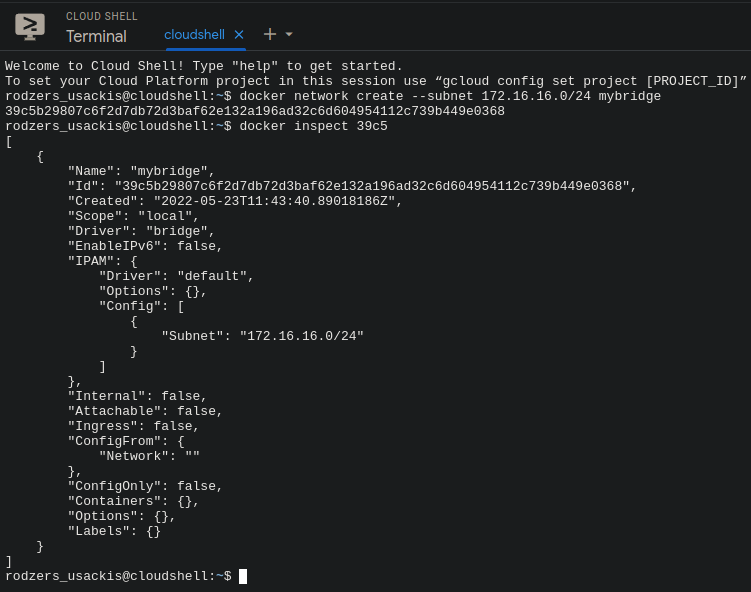
$ docker network create --subnet **172.16.16.0/24** *mybridge*

Use inspect to check your new network:

$ docker inspect *mybridge*

"Name": "mybridge",  
 "Id": "1d5ee093edc78ad85da76af9f7434e28bf35c806fee0ecb361bd819696889187",  
 "Created": "2019-02-29T16:32:06.590604735+01:00",  
 "Scope": "local",  
 "Driver": "bridge",  
 "EnableIPv6": false,  
 "IPAM": {  
 "Driver": "default",  
 "Options": {},  
 "Config": [  
 {  
 "Subnet": "**172.16.16.0/24**"

…

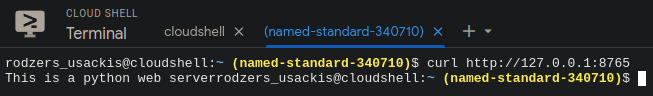


# Step 2: Create Python3 Script

Create the file webserver.py:

#!/usr/bin/env python3  
# Take care of the indentation!   
from http.server import BaseHTTPRequestHandler, HTTPServer  
import time  
hostName = "127.0.0.1"  
hostPort = 8765  
class MyServer(BaseHTTPRequestHandler):  
 def do\_GET(self):  
 self.send\_response(200)  
 self.send\_header("Content-type","text/html")   
 self.end\_headers()  
 self.wfile.write(bytes("This is a python web server","utf-8"))  
myServer = HTTPServer((hostName, hostPort), MyServer)  
print(time.asctime(), "Server Starts - %s:%s" % (hostName, hostPort))  
try:  
 myServer.serve\_forever()  
except KeyboardInterrupt:  
 pass  
myServer.server\_close()  
print(time.asctime(), "Server Stops - %s:%s" % (hostName, hostPort))

Test run with: python3 webserver.py. Open a browser on http://127.0.0.1:8765



If necessary install python with:  
$ sudo apt-get install python3 pip3  
$ pip3 install http.server

# Step 3: Create Dockerfile

In webserver.py change hostName = "127.0.0.1" to hostName = "172.16.16.16". We will run the python webserver in the container with that IP.

Create a Dockerfile:

- With python:3.6-stretch as a base image (This is a Debian image)  
- Which copies webserver.py to your image  
- Which starts up python3 webserver.py when the container is started

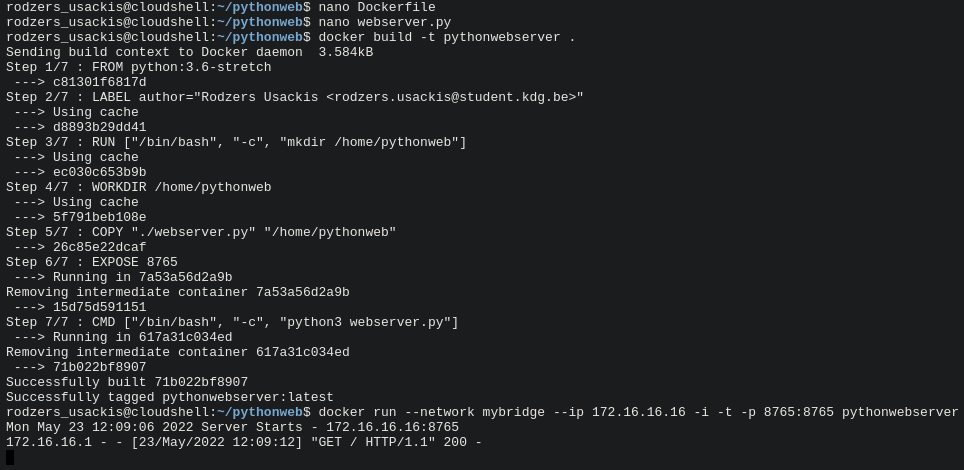
Build the image with docker build -t pythonwebserver .

Test the image with:  
$ docker run --network mybridge --ip 172.16.16.16 -i -t -p 8765:8765 pythonwebserver

Test by opening the URL at <http://172.16.16.16:8765>

Stop the container.

Remove unused networks: docker network prune





# Step 4: Create Docker Compose YAML

Instead of running the container with the above docker run command, we can also create a docker-compose.yml and run the container with docker compose up

Create a docker-compose.yml file with following options:  
- defines service pythonweb under services:  
- image: uses python:3.6-stretch as a base image

- command: the command to run is python3 webserver.py  
- volumes: Maps the current directory . to /app  
- build: the current directory  
- working\_dir: uses /app  
- ports: maps 8765 on the host to 8765 on the container

Add this to the pythonweb service:  
 networks:  
 mybridge:  
 ipv4\_address: 172.16.16.16  
  
Networks should also be defined in the compose file. So add these lines:

networks:  
 mybridge:  
 driver: bridge  
 ipam:  
 driver: default  
 config:  
 - subnet: 172.16.16.0/24

# Step 5: Start your Container

$ docker compose up  
  
Open a browser on <http://localhost:8765>

