Containerization assignment

Q2 &3

Create a file called app.py in your project directory and paste this in:

```
import time
import redis
from flask import Flask

app = Flask(__name__)
cache = redis.Redis(host='redis', port=6379)

def get_hit_count():
    retries = 5
    while True:
        try:
            return cache.incr('hits')
        except redis.exceptions.ConnectionError as exc:
            if retries == 0:
                raise exc
            retries -= 1
            time.sleep(0.5)

@app.route('/')
def hello():
    count = get_hit_count()
    return 'Hello World! I have been seen {} times.\n'.format(count)
```

Create another file called requirements.txt in your project directory and paste this in:

```
flask

redis
```

Create a Dockerfile

```
FROM python:3.7-alpine
WORKDIR /code
ENV FLASK_APP app.py
ENV FLASK_RUN_HOST 0.0.0.0
RUN apk add --no-cache gcc musl-dev linux-headers
COPY requirements.txt requirements.txt
RUN pip install -r requirements.txt
COPY . .
CMD ["flask", "run"]
```

Define services in a Compose file

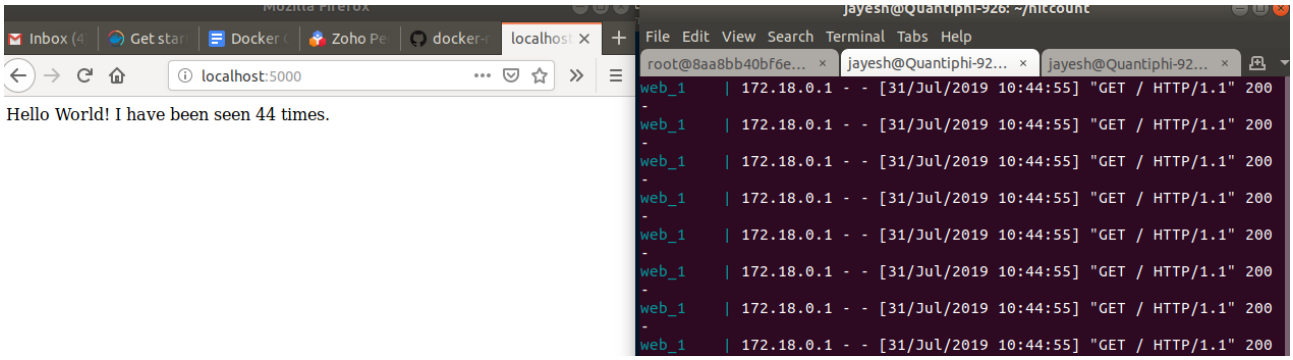
```
version: '3'
services:
  web:
    build: .
    ports:
      - "5000:5000"
  redis:
    image: "redis:alpine"
```

Build and run your app with Compose

From your project directory, start up your application by running docker-compose up.

Enter http://localhost:5000/ in a browse

Refresh the page.The number should increment.



Edit the Compose file to add a bind mount

Edit docker-compose.yml in your project directory to add a volume for the web service. File is attched in folder.

Re-build and run the app with Compose

From your project directory, type docker-compose up to build the app with the updated Compose file, and run it.

Update the application

Change the greeting in app.py and save it.File is attched in folder. Refresh the app in your browser.