BlackLinux

Docker Compose on Ubuntu 20

Get started with Docker Compose

In this tutorial, you build a simple python web application running on Docker Compose. The application uses the Flask Framework and maintains a hit counter in Redis.

Prerequisites

- Make sure you have already installed both Docker Engine and Docker Compose. You don’t need to install python or Redis, as both are provided by Docker Images.

Following Commands :

Define the application dependencies:

1. **Create a directory for the project:**

   ```
   # mkdir composetest

   # cd composetest
   ```

BlackLinux

2. Create a file called app.py in you 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)
```

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

```
flask
redis
```

BlackLinux

Create a Dockerfile:

1. **In your project directory, create a file named Dockerfile and paste the following:**

```
# syntax=docker/dockerfile:1

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
EXPOSE 5000
COPY . .
CMD ["flask", "run"]
```

2. **This tells docker to:**

    - Build an image starting with the **Python 3.7** image.
    - Set the working directory to **/code**.
    - Set environment variables used by the **flask** command.
    - Install **gcc** and other dependencies
    - Copy **requirements.txt** and install the Python dependencies.
    - Add metadata to the image to describe that the container is listening on port **5000**
    - Copy the current directory . in the project to the **workdir .** in the image.
    - Set the default command for the container to **flask run**.

BlackLinux

3. Define services in a compose file.

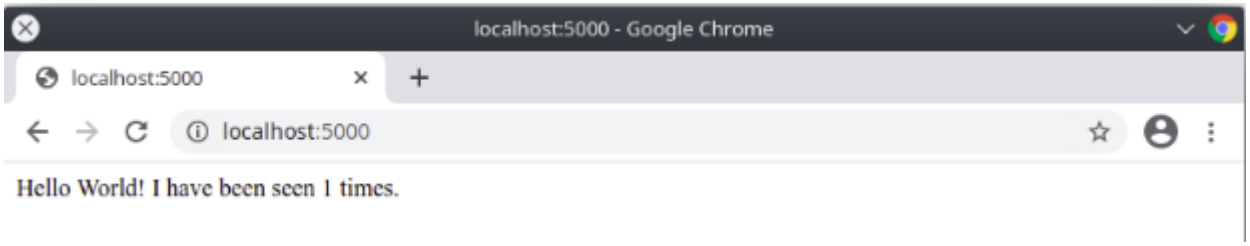
- Create a file called `docker-compose.yml` in your project directory and paste the following:

```
version: "3.9"
services:
  web:
    build: .
    ports:
      - "5000:5000"
  redis:
    image: "redis:alpine”
```

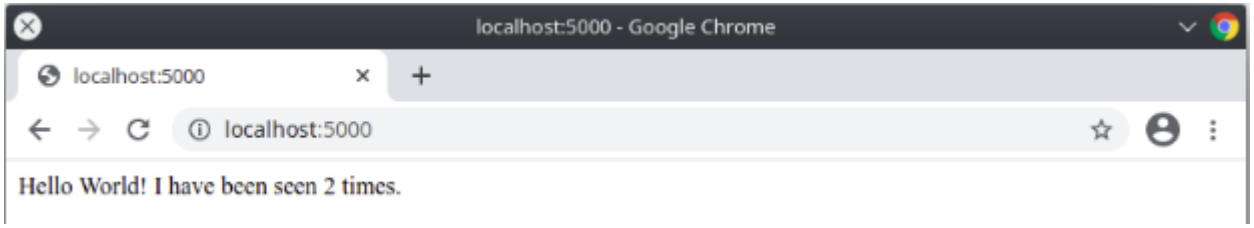
4. Build and run your app with compose.
   - From your project directory, start up your application by running `docker-compose up`.

     ```
     # docker-compose up
     ```

   - Enter http://localhost:5000/ in a browser to see the application running.



- Refresh the page.



i. The number should increase.

```
Hello World! I have been seen 2 times.
```

5. Edit the Compose file to add a bind mount

- Edit **docker-compose.yml** in your project directory to add a **bind mount** for the **web** service:

```
version: "3.9"
services:
  web:
    build: .
    ports:
      - "5000:5000"
    volumes:
      - .:/code
    environment:
      FLASK_ENV: development
  redis:
    image: "redis:alpine"
```

BlackLinux

6. **Re-build and run the app with compose.**

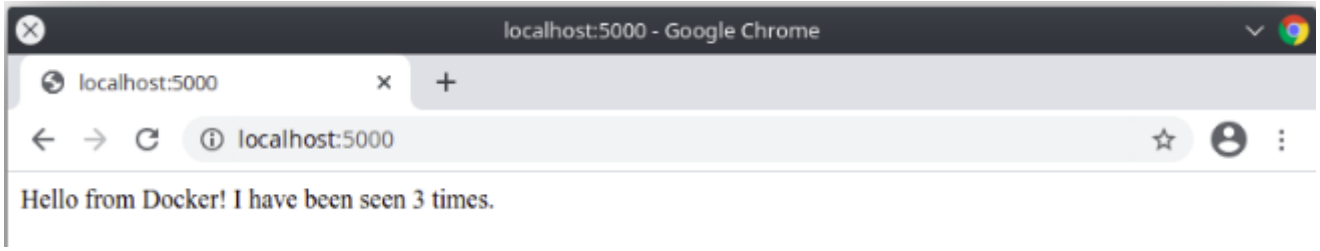
```
# docker-compose up
```

7. **Update the application**

   - Change the greeting in **app.py** and save it. For example, change the **Hello World!** message to **Hello from Docker!**:

```
return 'Hello from Docker! I have been seen {} times.\n'.format(count)
```

   - Refresh the app in your browser. The greeting should be updated, and the counter should still be incrementing.



8. **Experiment with some other commands:**
   - If you want to run your service in the background, you can pass the **-d** flag (for "detached" mode) to **docker-compose up** and use **docker-compose ps** to see what is currently running:

```
# docker-compose up -d
```

BlackLinux

```
# docker-compose ps

       Name                     Command               State           Ports
-------------------------------------------------------------------------------------
composetest_redis_1   docker-entrypoint.sh redis ...   Up              6379/tcp
composetest_web_1     flask run                        Up              0.0.0.0:5000->


# docker-compose stop
# docker-compose rm
# docker-compose down --volumes
```