docker-compose

This is the documentation for an **Introduction to Docker** for Redapt's "Docker Workshop".

by Christoph Champ for Redapt, Inc. (March 2018)

Docker compose

Compose is a tool for defining and running multi-container Docker applications. With Compose, you use a YAML file to configure your application's services. Then, with a single command, you create and start all the services from your configuration. To learn more about all the features of Compose, see the list of features.

Using Compose is basically a three-step process:

- Define your app's environment with a `Dockerfile` . so it can be reproduced anywhere.
- Define the services that make up your app in `docker-compose.yml` so they can be run together in an isolated environment.
- Run `docker-compose up` and Compose starts and runs your entire app.

Basic example

Note: This is based off of this article.

In this basic example, we will build a simple Python web application running on Docker Compose. The application uses the Flask framework and maintains a hit counter in Redis.

Note: This section assumes you already have Docker Engine and Docker Compose installed.

- Create a directory for the project:

```
$ mkdir compose-test && cd $_
```

- 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)

if __name__ == "__main__":
    app.run(host="0.0.0.0", debug=True)
```

In this example, `redis` is the hostname of the redis container on the application's network. We use the default port for Redis: `6379`.

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

```
flask
redis
```

- Create a Dockerfile (this Dockerfile will be used to build an image that contains all the dependencies the Python application requires, including Python itself):

```
FROM python:3.4-alpine
ADD . /code
WORKDIR /code
RUN pip install -r requirements.txt
CMD ["python", "app.py"]
```

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

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

- Build and run this app with Docker Compose:

```
$ docker-compose up
```

Compose pulls a Redis image, builds an image for your code, and starts the services you defined. In this case, the code is statically copied into the image at build time.

- Test the application:

```
$ curl localhost:5000
Hello World! I have been seen 1 times.

$ for i in $(seq 1 10); do curl -s localhost:5000; done
Hello World! I have been seen 2 times.
Hello World! I have been seen 3 times.
Hello World! I have been seen 4 times.
Hello World! I have been seen 5 times.
Hello World! I have been seen 6 times.
Hello World! I have been seen 7 times.
Hello World! I have been seen 8 times.
Hello World! I have been seen 9 times.
```

```
Hello World! I have been seen 10 times.
Hello World! I have been seen 11 times.
```

- Display the running processes:

```
$ docker-compose top
compose-test_redis_1
  UID       PID    PPID    C   STIME   TTY     TIME         CMD
------------------------------------------------------------------
systemd+   29401   29367   0   15:28   ?     00:00:00   redis-server

compose-test_web_1
UID     PID    PPID    C   STIME   TTY     TIME                CMD
-------------------------------------------------------------------------------
root   29407   29373   0   15:28   ?     00:00:00   python app.py
root   29545   29407   0   15:28   ?     00:00:00   /usr/local/bin/python app.py
```

- Shutdown the application:

```
$ Ctrl+C
#~OR~
$ docker-compose down
```