Docker Compose with Django Apps

We can use Docker Compose to build and manage multiple-container setups. In this example, we'll take a Django application using PostgreSQL for our database. (Good development habits mean we separate our concerns and allow one container to run the database and the other to run our web application.) Docker Compose lets us use a single host to handle multiple containers and builds the network layer so our containers in this project can talk to one another.

Set Up Your Host

In four steps, we'll have a fully dockerized Django application configured and set up using Docker Compose.

Set Up Docker

We need a Dockerfile, Python dependencies, and a Docker Compose file.

1. Create and change into your directory:

```
$ mkdir compose_django
$ cd compose_django
```

2. You need a `requirements.txt` file outlining the Python dependencies:

```
Django
psycopg2
```

3. Write your `Dockerfile`:

```
FROM python:2.7
ENV PYTHONUNBUFFERED 1
RUN mkdir /code
WORKDIR /code
ADD requirements.txt /code/
RUN pip install -r requirements.txt
ADD . /code/
```

4. Write your Docker Compose file (`docker-compose.yml`):

```
version: '2'
services:
    web:
        build: .
        command: python manage.py runserver 0.0.0.0:8000
        volumes:
            - .:/code
        ports:
            - "8000:8000"
        depends_on:
            - db
    db:
        image: postgres
```

Notice the composition defines two services, `db` and `web`. `db` loads the `postgres` image from Docker Hub. Compose builds the `web` service from the `Dockerfile` in the local host directory, mounts it to `/code` in the container (so you can alter your code without rebuilding the image every time), and links the `web` service to the `db` service.

Set Up A Django Project

From the root of your project directory (`~/compose_django` in our example), build your project:

```
$ docker-compose run web django-admin.py startproject compose_django_example
```

Amongst other output, you should see an error.

We tell Docker Compose to run `django-admin.py startproject compose_django_example` in a container using the `web` service image and configuration. However, the `web` image does not exist yet, so Compose builds it from the current directory (look at `build` in the configuration file). Compose runs and executes `django-admin.py startproject` after building the `web` image (`django-admin.py startproject` tells Django to scaffold a Django project). Take a look at what Django creates:

```
drwxr-xr-x 3 root root 4096 Jul 23 19:09 compose_django_example
-rw-r--r-- 1 pdctut pdctut  256 Jul 23 19:07 docker-compose.yml
-rw-r--r-- 1 pdctut pdctut  147 Jul 23 19:07 Dockerfile
-rw-r--r-- 1 pdctut pdctut   17 Jul 23 19:07 requirements.txt
```

Notice root owns the `compose_django_example` file (as opposed to `pdctut` on the other files); remember the container runs as the root user. On a Mac or Windows machine, you should already have access to the other files. On Linux, you'll need to `chown` the new files: `sudo chown -R pdctut:pdctut` (for user `pdctut`; you may substitute with `$USER` if you do not wish to type your username; the `-R` option tells `chown` to recursively change ownership of the files).

You need to move all of your files out of `compose_django_example` as well:

```
$ ls -l
drwxr-xr-x 3 pdctut pdctut 4096 Jul 23 19:09 compose_django_example
-rw-r--r-- 1 pdctut pdctut  256 Jul 23 19:07 docker-compose.yml
-rw-r--r-- 1 pdctut pdctut  147 Jul 23 19:07 Dockerfile
-rw-r--r-- 1 pdctut pdctut   17 Jul 23 19:07 requirements.txt
$ mv compose_django_example compose_django_example_OLD
$ mv compose_django_example_OLD/* .
$ ls -l
drwxr-xr-x 2 pdctut pdctut 4096 Jul 23 19:09 compose_django_example_OLD
drwxr-xr-x 2 pdctut pdctut 4096 Jul 23 19:35 compose_django_example
-rw-r--r-- 1 pdctut pdctut  256 Jul 23 19:29 docker-compose.yml
-rw-r--r-- 1 pdctut pdctut  147 Jul 23 19:07 Dockerfile
-rwxr-xr-x 1 pdctut pdctut  265 Jul 23 19:09 manage.py
-rw-r--r-- 1 pdctut pdctut   17 Jul 23 19:07 requirements.txt
$ rmdir compose_django_example_OLD
$ ls -l
drwxr-xr-x 2 pdctut pdctut 4096 Jul 23 19:35 compose_django_example
-rw-r--r-- 1 pdctut pdctut  256 Jul 23 19:29 docker-compose.yml
-rw-r--r-- 1 pdctut pdctut  147 Jul 23 19:07 Dockerfile
```

```
-rwxr-xr-x 1 pdctut pdctut  265 Jul 23 19:09 manage.py
-rw-r--r-- 1 pdctut pdctut   17 Jul 23 19:07 requirements.txt
```

Set Up PostgreSQL

1. Edit `compose_django/compose_django_example/compose_django_example/settings.py` and replace the `DATABASES` section with the following:

    ```
    DATABASES = {
        'default': {
            'ENGINE': 'django.db.backends.postgresql_psycopg2',
            'NAME': 'postgres',
            'USER': 'postgres',
            'HOST': 'db',
            'PORT': 5432,
        }
    }
    ```

2. Set the user and password for Postgres in your shell:

    ```
    $ POSTGRES_PASSWORD="pleaseUseAStr0ngPassword"
    $ POSTGRES_USER="postgres"
    $ echo $POSTGRES_PASSWORD
    pleaseUseAStr0ngPassword
    $ echo $POSTGRES_USER
    postgres
    ```

Run It

Run `docker-compose up -e POSTGRES_PASSWORD=pleaseUseAStrongPassw0rd` and Docker Compose launches your Django app.

If you have localhost access to your host (i.e., you do not use a remote solution to deploy Docker), point your browser to `http://0.0.0.0:8000`, `http://127.0.0.1:8000`, or `http://localhost:8000`. On a Mac, you need to use `docker-machine ip MACHINE_VM` to get your Docker host's IP address (then use that address like `http://MACHINE_IP:8000` to access your web page). If you have remote access, simply go to your server IP (e.g. `http://8.8.4.4:8000`).

When visiting the page, you should see confirmation that the Django installation took (something to the effect of “It worked! Congratulations on you first Django-powered page.”).

Common Issues

Building Your Project

Amongst other output, you should see a warning when executing the Compose `run` command:

```
WARNING: Image for service web was built because it did not already exist. To rebuild
```

Avoid this error by building the image *first* with `docker build -t web .`.

Django Setup

Make sure you move all of your Django files into your main directory, or Docker cannot find `manage.py`.

PostgreSQL Setup

You need to set your PostgreSQL username and password (at least your password, PostgreSQL defaults to user and database `postgres`).

Next: Docker Compose with Flask Apps
Create repeatable Flask (Python) environments

using Docker Compose.

By Runnable: The service that speeds up development by providing full-stack environments for every code branch.

Visit Runnable