Search the Docs (/)

What is Docker? (http://www.docker.com/whatisdocker/) Use Cases (http://www.docker.com/resources/usecases/)

Try It! (http://www.docker.com/tryit/) Browse (https://registry.hub.docker.com)

Log In (https://hub.docker.com/account/login) Sign Up (https://hub.docker.com/account/signup)

About (/) Installation (/installation/ubuntulinux/) User Guide (/userguide/) Docker Hub (/docker-hub/)

Examples (/examples/nodejs_web_app/) Articles (/articles/basics/) Reference (/reference/commandline/cli/)

Contributor Guide (/project/who-written-for/)

Docker Compose

Overview

Compose is a tool for defining and running complex applications with Docker. With Compose, you define a multi-container application in a single file, then spin your application up in a single command which does everything that needs to be done to get it running.

Compose is great for development environments, staging servers, and CI. We don't recommend that you use it in production yet.

Using Compose is basically a three-step process.

First, you define your app's environment with a `Dockerfile` so it can be reproduced anywhere:

```
FROM python:2.7
WORKDIR /code
ADD requirements.txt /code/
RUN pip install -r requirements.txt
ADD . /code
CMD python app.py
```

Next, you define the services that make up your app in `docker-compose.yml` so they can be run together in an isolated environment:

```
web:
  build: .
  links:
   - db
  ports:
   - "8000:8000"
db:
  image: postgres
```

Lastly, run `docker-compose up` and Compose will start and run your entire app.

Compose has commands for managing the whole lifecycle of your application:

- Start, stop and rebuild services
- View the status of running services
- Stream the log output of running services
- Run a one-off command on a service

Compose documentation

- Installing Compose (/compose/install/)
- Command line reference (/compose/cli/)
- Yaml file reference (/compose/yml/)
- Compose environment variables (/compose/env/)
- Compose command line completion (/compose/completion/)

Quick start

Let's get started with a walkthrough of getting a simple Python web app running on Compose. It assumes a little knowledge of Python, but the concepts demonstrated here should be understandable even if you're not familiar with Python.

Installation and set-up

First, install Docker and Compose (/compose/install/).

Next, you'll want to make a directory for the project:

```
$ mkdir composetest
$ cd composetest
```

Inside this directory, create `app.py`, a simple web app that uses the Flask framework and increments a value in Redis:

```
from flask import Flask
from redis import Redis
import os
app = Flask(__name__)
redis = Redis(host='redis', port=6379)

@app.route('/')
def hello():
    redis.incr('hits')
    return 'Hello World! I have been seen %s times.' % redis.get('hits')

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

Next, define the Python dependencies in a file called `requirements.txt`:

```
flask
redis
```

Create a Docker image

Now, create a Docker image containing all of your app's dependencies. You specify how to build the image using a file called `Dockerfile` (http://docs.docker.com/reference/builder/):

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

This tells Docker to include Python, your code, and your Python dependencies in a Docker image. For more information on how to write Dockerfiles, see the Docker user guide (https://docs.docker.com/userguide/dockerimages/#building-an-image-from-a-dockerfile) and the Dockerfile reference (http://docs.docker.com/reference/builder/).

Define services

Next, define a set of services using `docker-compose.yml`:

```
web:
  build: .
  command: python app.py
  ports:
   - "5000:5000"
  volumes:
   - .:/code
  links:
   - redis
redis:
  image: redis
```

This defines two services:

- `web`, which is built from the `Dockerfile` in the current directory. It also says to run the command `python app.py` inside the image, forward the exposed port 5000 on the container to port 5000 on the host machine, connect up the Redis service, and mount the current directory inside the container so we can work on code without having to rebuild the image.
- `redis`, which uses the public image redis (https://registry.hub.docker.com/_/redis/), which gets pulled from the Docker Hub registry.

Build and run your app with Compose

Now, when you run `docker-compose up`, Compose will pull a Redis image, build an image for your code, and start everything up:

```
$ docker-compose up
Pulling image redis...
Building web...
Starting composetest_redis_1...
Starting composetest_web_1...
redis_1 | [8] 02 Jan 18:43:35.576 # Server started, Redis version 2.8.3
web_1   |  * Running on http://0.0.0.0:5000/
```

The web app should now be listening on port 5000 on your Docker daemon host (if you're using Boot2docker, `boot2docker ip` will tell you its address).

If you want to run your services in the background, you can pass the `-d` flag (for daemon mode) to `docker-compose up` and use `docker-compose ps` to see what is currently running:

```
$ docker-compose up -d
Starting composetest_redis_1...
Starting composetest_web_1...
$ docker-compose ps
    Name                 Command            State       Ports
-------------------------------------------------------------------
composetest_redis_1   /usr/local/bin/run         Up
composetest_web_1     /bin/sh -c python app.py   Up      5000->5000/tcp
```

The `docker-compose run` command allows you to run one-off commands for your services. For example, to see what environment variables are available to the `web` service:

```
$ docker-compose run web env
```

See `docker-compose --help` to see other available commands.

If you started Compose with `docker-compose up -d`, you'll probably want to stop your services once you've finished with them:

```
$ docker-compose stop
```

At this point, you have seen the basics of how Compose works.

- Next, try the quick start guide for Django (/compose/django/), Rails (/compose/rails/), or Wordpress (/compose/wordpress/).
- See the reference guides for complete details on the commands (/compose/cli/), the configuration file (/compose/yml/) and environment variables (/compose/env/).

Release Notes

Version 1.2.0 (April 7, 2015)

For complete information on this release, see the 1.2.0 Milestone project page (https://github.com/docker/compose/wiki/1.2.0-Milestone-Project-Page). In addition to bug fixes and refinements, this release adds the following:

- The `extends` keyword, which adds the ability to extend services by sharing common configurations. For details, see PR #972 (https://github.com/docker/compose/pull/1088).
- Better integration with Swarm. Swarm will now schedule inter-dependent containers on the same host. For details, see PR #972 (https://github.com/docker/compose/pull/972).

Getting help

Docker Compose is still in its infancy and under active development. If you need help, would like to contribute, or simply want to talk about the project with like-minded individuals, we have a number of open channels for communication.

- To report bugs or file feature requests: please use the issue tracker on Github (https://github.com/docker/compose/issues).
- To talk about the project with people in real time: please join the `#docker-compose` channel on IRC.
- To contribute code or documentation changes: please submit a pull request on Github (https://github.com/docker/compose/pulls).

For more information and resources, please visit the Getting Help project page (https://docs.docker.com/project/get-help/).

Governance
(https://www.docker.com/community/governance/)
Forums
(http://forums.docker.com)
IRC
(http://botbot.me/freenode/docker)
GitHub
(https://github.com/docker/docker)
Stackoverflow
(http://stackoverflow.com/search?
q=docker)
Swag
(http://www.cafepress.com/docker)

Partner Solutions

Find a Partner
(https://www.docker.com/partners/find/)
Partner Program
(https://www.docker.com/partners/program/)
Learn More
(https://www.docker.com/partners/learn/)

Online Tutorial
(http://www.docker.com/tryit/)
How To Buy
(https://www.docker.com/resources/howtobuy/)
Status
(http://status.docker.com)
Security
(https://www.docker.com/resources/security/)

Press
(https://www.docker.com/company/press/)
Careers
(https://www.docker.com/company/careers/)
Contact
(https://www.docker.com/company/contact/)

(http://twitter.com/docker)
Google+
(https://plus.google.com/u/0/108146856671
Facebook
(https://www.facebook.com/docker.run)
YouTube
(http://www.youtube.com/user/dockerrun)

(https://www.linkedin.com/company/docke
GitHub
(https://github.com/docker/)
Reddit
(http://www.reddit.com/r/docker)
AngelList
(https://angel.co/docker-
inc-1)