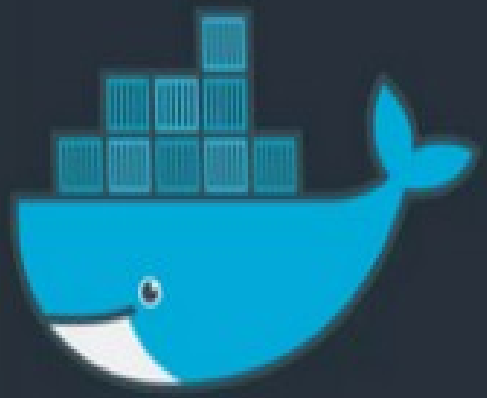
Docker Compose



Did you know?

There are 3,00,000 Docker Compose files in GitHub today

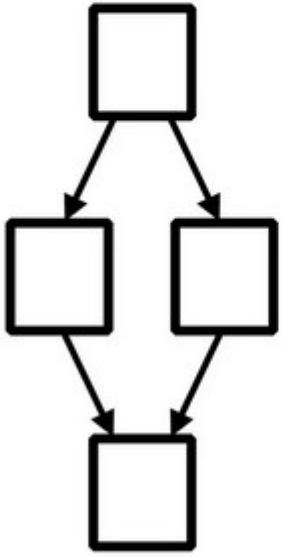
Multi-container apps are a hassle.

- Build images from Dockerfiles
- Pull images from the Hub or a private registry
- Configure and create containers
- Start and stop containers
- Stream their logs

Multi-container apps are a hassle.

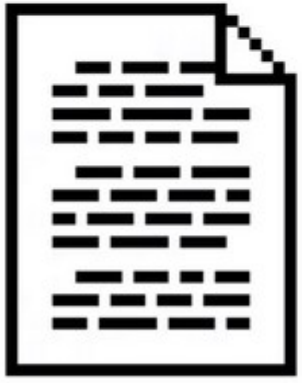
```
$ docker pull redis:latest

$ docker build -t web .

$ docker run -d --name=db redis:latest redis-server
--appendonly yes

$ docker run -d --name=web --link db:db -p
5000:5000 -v `pwd`:/code web python app.py
```

Multi-container apps are a hassle.



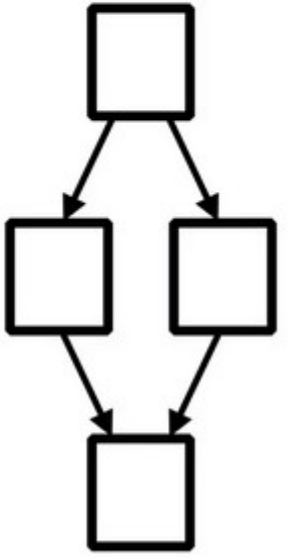
```
$ docker pull ...
$ docker pull ...
$ docker build ...
$ docker build ...

$ docker run ...
$ docker run ...
$ docker run ...
$ docker run ...
```

Docker Compose:
Get an app running in one command.



Text file

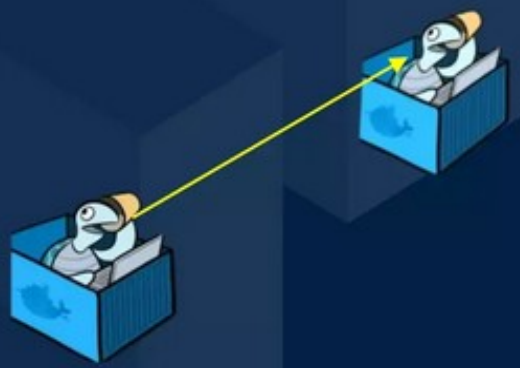
$ docker up



Building Blocks of Docker Compose

Services

Volumes



Networking

Docker Compose: Multi Container Applications


```
containers:
  web:
    build: .
    command: python app.py
    ports:
    - "5000:5000"
    volumes:
    - .:/code
    environment:
    - PYTHONUNBUFFERED=1
  redis:
    image: redis:latest
    command: redis-server --appendonly yes
```

Native Docker API

The same API used by **other Docker commands**.

The same API used by **other Docker tools**.

The same API **exposed by Docker Swarm**.

Multi-container apps on multi-host clusters.

Built into the Docker client

- New command: `docker up`
- Enhanced commands: `docker build`, `pull`, `run`, `start`, `stop`, `kill`, `rm`...

app.py

```
from flask import Flask
from redis import Redis

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.\n' % redis.get('hits')

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

requirements.txt

```
flask
redis
```

Dockerfile

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

group.yml

```
name: counter

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

Enhanced syntax which refers to group.yml

Same as before:

```
docker rm mycontainer
```

New form:

```
docker rm :web
```

Equivalent to:

```
docker rm [name of the web container]
```

(if the web container exists)

Enhanced syntax which refers to group.yml

New form:

```
docker rm :
```

Equivalent to:

```
docker rm [all containers defined in group.yml]
```

Enhanced syntax which refers to group.yml

List all running containers defined in group.yml:

```
docker ps :
```

Manually rebuild the web image:

```
docker build :web
```

Manually re-pull the redis image:

```
docker pull :redis
```

Docker Compose is a 3 Steps Process

Define your app's environment with a Dockerfile

Define the services that make up your app in Docker Compose file

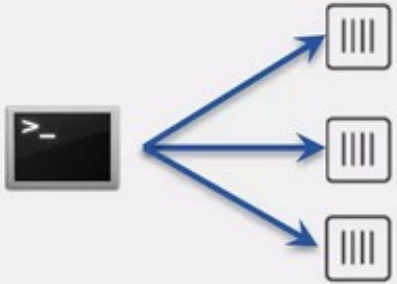
Run the CLI:

$ docker-compose up

Docker Compose: Multi Container Applications

Without Compose

- Build and run one container at a time
- Manually connect containers together
- Must be careful with dependencies and start up order



With Compose

- Define multi container app in compose.yml file
- Single command to deploy entire app
- Handles container dependencies
- Works with Docker Swarm, Networking, Volumes, Universal Control Plane