Docker - Nivelando conhecimento - Parte 2

Docker Compose, porque um é pouco, dois é bom mas três já é demais

Hugo Posca

PagarMe

30 setembro 2016

Depois da apresentação anterior

Agora que você conhece o docker você começou a...

- “Buildar” suas próprias imagens
- “Conteinerizar” tudo
- Subir a sua aplicação utilizando o *docker* por linha de comando

Tudo certo!

Até que...

Você começou a subir uma aplicação que dependia de outros recursos, que estavam em outros contêineres...

```
docker pull ...
docker pull ...

docker build ...
docker build ...

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

Até que...

E com o agravante de que os contêineres têm que subir em ordem, com mais configurações e detalhes:

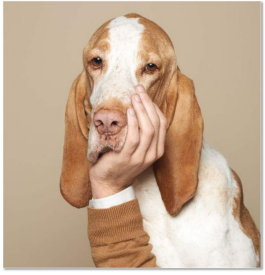
```
docker run -d --name redis redis:3.0.7-alpine
docker run -d --name mail schickling/mailcatcher
docker run -d --name database \
    -v 'pwd'/data:/var/lib/postgresql/data \
    postgres:9.5.4

docker run -d --name application \
    --link database --link redis --link mail \
    -p 5000:5000 \
    my_application:1.0
```

Até que...

Parabéns! Você acaba de ganhar:

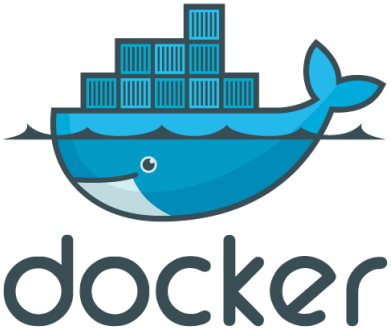
- Muitas coisas para digitar e lembrar
- Muitos detalhes pra esquecer e errar
- Matar e subir contêineres começou a ficar difícil

Enfim... Um processo suuuper divertido



Êêêeba!

Docker Compose



Origens

- Em **2013** a empresa Orchard iniciou um projeto chamado fig
- Em **2014** foram adquiridos pela Docker Inc.
- Nasce o *docker compose*

Propósito

Compose is a tool for defining and running multi-container Docker applications. With Compose, you use a Compose file to configure your application's services. Then, using a single command, you create and start all the services from your configuration.

Instalação

```
curl -L https://github.com/docker/compose/releases/download/\
1.8.0/docker-compose-$(uname -s)-$(uname -m) > \
/usr/local/bin/docker-compose

chmod +x /usr/local/bin/docker-compose
```

Mais opções na página de instalação

O segredo

O segredo está no arquivo ***docker-compose.yml***:

```
version: '2'
services:
  db:
    image: postgres:9.5.4
    volumes:
      - ./data:/var/lib/postgresql/data
  web:
    build: .
    command: bundle exec rails s -p 3000 -b '0.0.0.0'
    volumes:
      - .:/myapp
    ports:
      - "3000:3000"
    links:
      - db
```

O segredo

E agora, para subir tudo o que foi definido no ***docker-compose.yml***:

```
docker-compose up
```

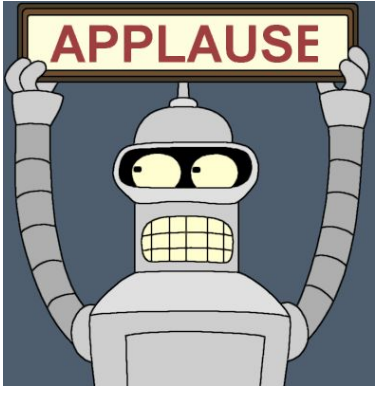
- Se as imagens não foram "buildadas", elas serão!
- Os contêineres de dependências subirão na ordem!
  1. Banco de dados
  2. Aplicação
- Depois de um único comando, tudo estará funcionando!

O segredo

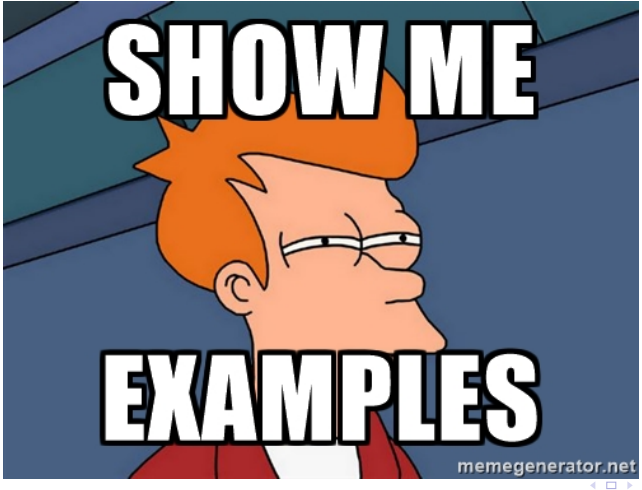
E agora, para subir tudo o que foi definido no ***docker-compose.yml***:

```
docker-compose up
```

- Se as imagens não foram "buildadas", elas serão!
- Os contêineres de dependências subirão na ordem!
  1. Banco de dados
  2. Aplicação
- Depois de um único comando, tudo estará funcionando!



...


SHOW ME
EXAMPLES
memegenerator.net

#1 - Subindo um container que depende de outros

```
docker run -d --name redis ...
docker run -d --name mail ...
docker run -d --name database ...

docker run -d --name application \
    --link database \
    --link redis \
    --link mail \
    my_application:1.0
```

#1 - Subindo um container que depende de outros

No seu **docker-compose.yml**:

```
version: '2'
services:
  redis:
    image: redis
  mail:
    image: schickling/mailcatcher
  database:
    image: postgres:9.5.4
    ...
  application:
    build: .
    ...
    links:
    - database
    - redis
    - mail
```

#1 - Subindo um container que depende de outros

No seu **docker-compose.yml**:

```
version: '2'
services:
  redis:
    image: redis
  mail:
    image: schickling/mailcatcher
  database:
    image: postgres:9.5.4
    ...
  application:
    build: .
    ...
    links:
    - database
    - redis
    - mail
```

E agora, basta executar:

```
docker-compose up
```

- Se as imagens não foram baixadas ou “buildadas”, elas serão!
- Os contêineres subirão na ordem:
  1. Redis, mail, database
  2. Aplicação

Se quiser deixar tudo rodando em background, sem travar o seu terminal:

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

#2 - Montando volumes

```
docker run -d --name database \
  -v `pwd`/data:/var/lib/postgresql/data \
  postgres:9.5.4

docker run -d --name application \
  ...
  -v `pwd`/:/code
  my_application:1.0
```

#2 - Montando volumes

```
docker run -d --name database \
  -v 'pwd'/data:/var/lib/postgresql/data \
  postgres:9.5.4

docker run -d --name application \
  ...
  -v 'pwd'/:/code
  my_application:1.0
```

No seu **docker-compose.yml**:

```
version: '2'
services:
  ...
  database:
    image: postgres:9.5.4
    volumes:
      - ./data:/var/lib/postgresql/data
  application:
    build: .
    ...
    volumes:
      - ./:/code
    links:
      - database
    ...
```

#2.1 - Montando volumes

No docker existe o conceito de *volumes nomeados*:

```
version: '2'
services:
  ...
  database:
    image: postgres:9.5.4
    volumes:
      - datavolume:/var/lib/postgresql/data
    ...
```

Dessa forma *datavolume* será um volume gerenciado pelo próprio docker e não um diretório explícito.
Experimente:

```
docker volume ls
```

#3 - Liberando portas

```
docker run -d \
  -p 8300-8302:8300-8302 \
  -p 8301-8302:8301-8302/udp \
  -p 8400:8400 \
  -p 8500:8500 \
  -p 8600:8600 \
  -p 8600:8600/udp \
  consul:latest
```

#3 - Liberando portas

```
docker run -d \
  -p 8300-8302:8300-8302 \
  -p 8301-8302:8301-8302/udp \
  -p 8400:8400 \
  -p 8500:8500 \
  -p 8600:8600 \
  -p 8600:8600/udp \
  consul:latest
```

No seu **docker-compose.yml**:

```
version: '2'
services:
  consul:
    image: consul:latest
    ports:
    - 8300-8302:8300-8302
    - 8301-8302:8301-8302/udp
    - 8400:8400
    - 8500:8500
    - 8600:8600
    - 8600:8600/udp
```

#4 - “Buildando” outro Dockerfile, com outro contexto

```
docker build \
  -f Dockerfile-alternativo \
  src/
```

#4 - “Buildando” outro Dockerfile, com outro contexto

```
docker build \
  -f Dockerfile-alternativo \
  src/
```

No seu **docker-compose.yml**:

```
version: '2'
services:
  application:
    build:
      context: src/
      dockerfile: Dockerfile-alternativo
```

#5 - Mas eu preciso escalar...

"Minha aplicação tem que subir vários containers idênticos"

```
docker-compose scale application=5
```

#5 - Mas eu preciso escalar...

"Minha aplicação tem que subir vários containers idênticos"

```
docker-compose scale application=5
```

Comandos úteis

```
docker-compose build # Constroi as imagens que estao
                     # definidas no seu docker-compose.yml
docker-compose build --no-cache # Constroi as imagens,
                                # ignorando construcoes anteriores
docker-compose up [servico] # Sobe todos os servicos ou
                            # o servico especificado
docker-compose stop     # Para os containers que estao executando
docker-compose start    # (Re)inicia containers parados
docker-compose rm [-f]  # Remove containers parados
```

Onde usar?

- Em ambientes de desenvolvimento
- No seu workflow de integração contínua

```
docker-compose up
./ci-tests.sh
docker-compose down
```

Em produção... *o buraco é mais embaixo*.

Referências

- Docker Compose
- Instalando
- Docker Compose File Reference
- Quickstart Django
- Quickstart Rails

Pensamento da noite