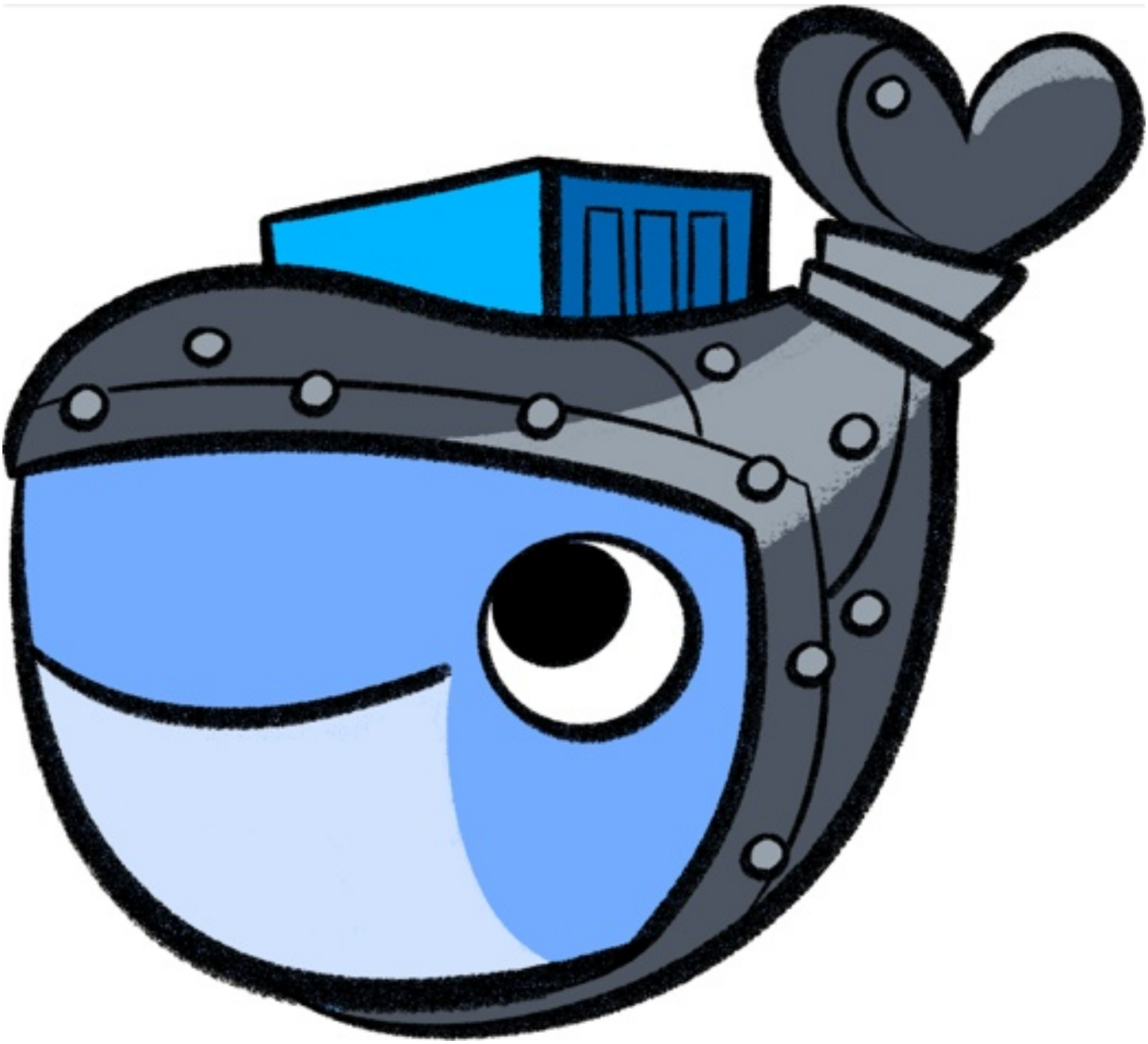


Docker部署 - Django+MySQL+uWSGI+Nginx

GL0oMY
逃离舒适

前言

本文章仅作为个人第一个Django项目部署流程的总结，也存在太多不优的地方(例如代码放在主机上，违背了Docker的集装箱原则、未实现持续部署)，也不建议作为一篇 Docker或项目部署指导。

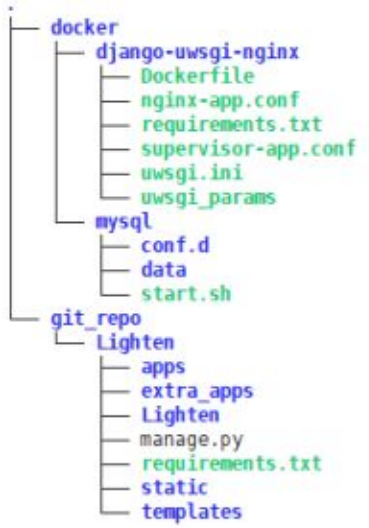
有纰漏或待优化的过程恳请读者指出，感谢

部署环境

阿里云ECS Ubuntu16.04 64位

宿主机目录结构

/root下:



即Docker相关文件目录: /root/docker

webapp项目: /root/git_repo/Lighten (Lighten为项目名)

├── docker

| ├── django-uwsgi-nginx

| | ├── Dockerfile # 用于创建Django+uWSGI+Nginx镜像

| | ├── nginx-app.conf # Nginx配置文件

| | ├── requirements.txt # Django项目的python库依赖文件 从/Lighten中copy

| | ├── supervisor-app.conf # Supervisor配置文件

| | ├── uwsgi.ini # uWSGI配置

```
|   └── mysql
|   ├── conf.d # MySQL配置，包含my.cnf
|   ├── data # 空目录，将作为数据卷挂载到MySQL容器中
|   └── start.sh # 构建MySQL容器的脚本
└── git_repo
    └── Lighten # Django项目名
        ├── apps
        ├── extra_apps # 只包含xadmin
        ├── Lighten
        ├── manage.py # 入口程序
        ├── requirements.txt
        ├── static
        └── templates
```

从GitHub clone项目代码

```
cd /root/git_repo
```

然后clone

```
git clone https://github.com/yiyuhao/Lighten.git
```

clone完后，就不用先理会了，这个目录会挂载到容器中，到时候再进行相关操作

安装Docker

docker.com/get-docker

创建MySQL容器

选择了创建2个Docker容器，一个是MySQL，一个是Django+uWSGI+Nginx且安装Supervisor进行进程管理。

下面进行MySQL容器的创建。

目录：

```
—mysql
 ├——— conf.d
 |   └——— my.cnf
 ├——— data
 └——— start.sh
```

start.sh

```
#!/bin/bash

echo "create a mysql container.."
docker run -d --name mysql \
           -v $(pwd)/conf.d:/etc/mysql/conf.d \
           -v $(pwd)/data:/var/lib/mysql \
           -e MYSQL_ROOT_PASSWORD="my-secret-password" \
           -e MYSQL_DATABASE="lighten" \
           -p 3307:3306 \
       mysql:5.7.19 \
           --character-set-server=utf8 --collation-server=utf8_general_ci
```

简单说明：docker run 为运行容器的命令，若本地仓库不存在mysql:5.7.19的镜像则自动从DockerHub pull下来。

-v, --volume list Bind mount a volume

-e, --env list Set environment variables

-p, --publish list Publish a container's port(s) to the host

my.cnf

```
[client]
default-character-set=utf8
[mysqld]
character-set-server=utf8
performance_schema = OFF
[mysql]
no-auto-rehash
default-character-set=utf8
```

然后运行start.sh

```
/root/docker/mysql/start.sh
```

可以看到容器已成功运行

```
docker ps -a
```

现在看看mysql容器是否正确运行

```
docker exec -it mysql bash
```

docker run创建时，写入的环境变量MYSQL_DATABASE会由mysql镜像处理，创建database

创建Django+uWSGI+Nginx+Supervisor镜像并启动容器

由于该容器需要与MySQL容器互联，Docker通过两种方式为容器公开连接信息：

- 更新环境变量
- 更新/etc/hosts文件

对于第一种方式：**互联之后会在该容器生成mysql地址、端口、密码等信息作为环境变量供其使用**，这些信息的格式是固定的。

因此需要在Django项目中读取这些环境变量。

创建镜像之前先修改一下Django项目中settings.py:

/root/git_repo/Lighten/Lighten/settings.py

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.mysql',
        'NAME': os.environ.get('MYSQL_DATABASE_NAME'),
        'USER': 'root',
        'PASSWORD': os.environ.get('MYSQL_ENV_MYSQL_ROOT_PASSWORD'),
        'HOST': os.environ.get('MYSQL_PORT_3306_TCP_ADDR'),
        'OPTIONS': {
            "init_command": "SET foreign_key_checks=0;",
        }
    }
}
```

```
docker run -d --link mysql_container:mysql webapp
```

--link name:alias，其中name是要连接的容器名称，alias是这个连接的别名。

变量名中的'MYSQL'就是alias，这一点需要注意。

目录：

下面开始编写Dockerfile:

```
FROM ubuntu:16.04

MAINTAINER Dockerfiles

# 安装git、python、nginx、supervisor等，并清理缓存
RUN apt-get update && \
    apt-get upgrade -y && \
    apt-get install -y \
        git \
        python \
        python-dev \
        python-setuptools \
        python-pip \
        nginx \
        supervisor \
        libmysqlclient-dev && \
```

```
RUN pip install -i https://pypi.doubanio.com/simple/ uwsgi

# 环境变量
ENV MYSQL_DATABASE_NAME lighten
ENV EMAIL_HOST_USER my-email@email.com
ENV EMAIL_HOST_PASSWORD my-secret-email-password

# nginx、supervisor配置
RUN echo "daemon off;" >> /etc/nginx/nginx.conf
COPY nginx-app.conf /etc/nginx/sites-available/default
COPY supervisor-app.conf /etc/supervisor/conf.d/

# 安装项目所需python第三方库
COPY requirements.txt /home/docker/code/Lighten/
RUN pip install -i https://pypi.doubanio.com/simple/ \
                -r /home/docker/code/Lighten/requirements.txt

# uwsgi.ini 及 uwsgi_params
COPY . /home/docker/code/

EXPOSE 80
CMD ["supervisord", "-n"]
```

pip安装使用了国内的镜像源，提高下载速度 pypi.doubanio.com/simpl...

环境变量是提供给Django项目，例如邮箱配置，数据库名

因为项目使用的的是python2.7版本，apt-get install python就好，如果是python3则需要修改Dockerfile

Nginx配置: nginx-app.conf

```
upstream django {
    server unix:/home/docker/code/app.sock; # for a file socket
}

server {
    listen      80 default_server;

    server_name X.X.X.X;
```

```
    client_max_body_size 75M;

    location /media  {
            alias /home/docker/code/Lighten/media;
    }

    location /static {
        alias /home/docker/code/Lighten/static;
    }

    location / {
        uwsgi_pass  django;
        include     /home/docker/code/uwsgi_params; # the uwsgi_params file
    }
}
```

uwsgi.ini

```
[uwsgi]
ini = :base

socket = %dapp.sock
master = true
processes = 4

enable-threads = true

[dev]
ini = :base
socket = :8001


[local]
ini = :base
http = :8000

[base]
chdir = %dLighten/
module=Lighten.wsgi:application
chmod-socket=666
```

supervisor-app.conf

```
[program:app-uwsgi]
command = /usr/local/bin/uwsgi --ini /home/docker/code/uwsgi.ini

[program:nginx-app]
command = /usr/sbin/nginx
```

然后通过docker build命令来创建镜像

```
docker build -t lighten /root/docker/django-uwsgi-nginx
```

然后需要等待一段时间...

创建完成，docker images来查看一下本地的镜像

根据创建好的镜像启动一个容器，挂载app目录，链接mysql容器

```
    -v /root/git_repo/Lighten/:/home/docker/code/Lighten
    --name webapp-lighten
    -p 80:80
lighten
bash
```

(备注: 这里换行只是为了排版)

-v, 是挂载一个宿主机目录Lighten，作为数据卷(也可以在Dockerfile中，声明VOLUME来配置)

通过--link参数即可连接mysql容器，如之前所述

启动后我们进入容器检查一下

```
docker exec -it webapp-lighten bash
```

下发命令env查看环境变量

可以看到，存在连接mysql容器后提供的部分环境变量(例如图中蓝框)，也有通过Dockerfile写入镜像的环境变量(例如图中红框)

接下来需要做的是Django db migrate

紧接着就可以通过supervisor启动项目了~

```
supervisord -n
```

部署流程到这里就完全完成，现在可以成功地访问网站了~

持续部署

(还不会Jenkins之类，先挖个坑。)

更新: 由于代码会有更改，所以决定暂时写个脚本来完成

start.sh

```
# 镜像不存在时创建镜像
if ! docker images | grep lighten; then
    echo 'The docker image does not exist,'
    echo 'Now creating image <lighten>...'
    docker build -t lighten $(pwd)
fi

# 镜像存在时，检查容器是否存在
if docker ps -a | grep -i lighten; then
    # 容器存在时则删除容器
    echo 'The docker container <lighten> already exist, deleting it...'
    docker rm -f webapp-lighten
    # 启动容器
    docker run -itd \
               --link mysql:mysql \
               -v /root/git_repo/Lighten/:/home/docker/code/Lighten \
               --name webapp-lighten \
               -p 80:80 \
           lighten
else
    # 第一次创建容器时需要更新数据库
    docker run -itd \
               --link mysql:mysql \
               -v /root/git_repo/Lighten/:/home/docker/code/Lighten \
               --name webapp-lighten \
               -p 80:80 \
           lighten \
           sh -c 'python /home/docker/code/Lighten/manage.py migrate && supervi
fi
```

源码

此次部署所编写的Dockerfile、start.sh及部分配置放在了GitHub:

github.com/yiyuhao/Dock...

编辑于 2017-09-24