# Dockerfile、镜像、容器快速入门

### 1.1、Dockerfile书写示例

Dockerfile可以用来生成Docker镜像，它明确的定义了Image的生成过程。虽然直接修改容器也可以提交生成镜像，但是这种方式生成的镜像对使用者是透明的，很难进行二次修改。最佳实践只建议使用Dockerfile生成镜像，开发者、使用者都需要明确的知道镜像的生成过程。

以下有两个示例。

示例一为Ubuntu 14.04之上的一个Nginx+uWSGI+Django项目Dockerfile示例。常用的Dockerfile有以下6部分组成：基础镜像信息、基础镜像、维护者信息、镜像操作指令、指定容器需要暴露的端口、指定容器需要使用的持久化存储、容器启动指令。

源代码、Dockerfile的SVN：

<https://192.168.5.246/svn/cloud/GalaxyManager/branches/0.1/DJANGO-DEMO>

|  |
| --- |
| # 1. 基础镜像信息.  # VERSION: V1.0.0.0  # Author: JunneYang  # 2. 基础镜像.  # FROM ubuntu:14.04  # 3. 维护者信息.  # MAINTAINER JunneYang <597092663@qq.com>  # 4. 镜像操作指令.  # RUN apt-get update -y  # RUN apt-get install python-pip -y  # RUN pip install --upgrade pip  # RUN pip install Django==1.8.3  # RUN apt-get install python-dev libmysqlclient-dev -y  # RUN pip install MySQL-python==1.2.5  # RUN pip install uwsgi  # RUN apt-get install nginx -y  FROM 10.5.24.46:80/nscloud/django:1.0.0.0  RUN mkdir -p /opt/applications  COPY ./applications/ /opt/applications/  RUN find /opt/applications/ -type d -name ".svn" | xargs rm -rf  RUN chmod 755 /opt/applications/run.sh  # 4. 指定容器需要暴露的端口.  EXPOSE 80 443  # 5. 指定容器需要使用的持久化存储.  VOLUME ["/opt/applications/logs", "/opt/applications/template/sharedfiles"]  # 6. 容器启动指令: 如果为LongTime Service，不能起为后台进程.  CMD ["/bin/bash", "-c", "env && cd /opt/applications/ && ./run.sh"] |

示例二为Mun模板。源码、Dockerfile等SVN：

<https://192.168.5.246/svn/cloud/Mun>

更多Dockerfile资料：

<https://yeasy.gitbooks.io/docker_practice/content/dockerfile/index.html>

<http://www.toxingwang.com/cloud/docker/3032.html>

<https://discuss.csphere.cn/t/dockerfile/1655>

<http://os.51cto.com/art/201507/485007.htm>

<http://www.blogjava.net/yongboy/archive/2013/12/16/407643.html>

<http://blog.tankywoo.com/docker/2014/05/08/docker-2-dockerfile.html>

<http://blog.csdn.net/qinyushuang/article/details/43342553>

<https://hub.docker.com/r/konradkleine/docker-registry-frontend/~/dockerfile/>

### 1.2、使用Dockerfile构建Docker镜像

使用如下命令，构建Docker镜像。

|  |
| --- |
| # docker build -t 10.5.24.46:80/nscloud/django:1.0.0.1 .  -t 表示该镜像的名称、标签  . 指定Dockerfile当前位置 |

使用如下命令查看Image：

|  |
| --- |
| # docker images |

### 1.3、使用镜像运行容器

使用如下命令，启动运行容器：

|  |
| --- |
| # docker run --name=django-demo -d -it -p 80:80 -p 443:443 \  -e MYSQL\_DATABASE="mysite" \  -e MYSQL\_USER="mysite" \  -e MYSQL\_PASSWORD="Admin@123" \  -e MYSQL\_HOST="10.5.24.31" \  -e MYSQL\_PORT="31111" \  -v /opt/fileserver/django-demo/logs:/opt/applications/logs \  -v /opt/fileserver/django-demo/files/sharedfiles:/opt/applications/template/sharedfiles \  10.5.24.46:80/nscloud/django:1.0.0.1 |

--name：指定容器名称，可以不指定

-d：容器后台运行

-it：容器交互式运行

-p：使用主机端口映射访问容器服务

-e：容器启动参数（环境变量），环境变量传递到镜像脚本中使用

-v：指定容器挂载的外部持久化存储路径

10.5.24.46:80/nscloud/django:1.0.0.1：容器启动镜像

使用如下命令，查看容器是否运行：

|  |
| --- |
| # docker ps |

使用如下命令，查看容器控制台输出：

|  |
| --- |
| # docker logs #容器ID# |

可以访问：<http://xx.xx.xx.xx/> 或者 <https://xx.xx.xx.xx/> 查看容器服务是否正常。

使用如下命令停止容器：

|  |
| --- |
| # docker stop #容器ID# |

更多命令，使用 docker –help 获取。

可以查看 /opt/fileserver/django-demo/logs、/opt/fileserver/django-demo/files/sharedfiles是否存在内容，检验持久化存储是否有效。

更多docker run，参考：

<http://www.tuicool.com/articles/uUBVJr>