#DevOps (/tags/DevOps) #Docker (/tags/Docker) Posted on 2018-04-03

Docker 构建镜像入门，以 Python Flask 为例

> 实验环境： Ubuntu 16.04 LTS
>
> 本文介绍如何自行构建一个封装了 python flask 的 docker 镜像，并且运行它的一个容器。

什么是Docker

Docker是一个操作系统容器管理工具，通过将应用打包到操作系统容器里面，从而让你能轻松管理和部署应用。

容器 vs 虚拟机

容器可能不如虚拟机一样为人所熟知，但是它们是另外的一种提供操作系统虚拟化的方法。然而，他们与标准的虚拟机有很大的差异。

标准的虚拟机通常包含一个完整的操作系统，OS软件包，最后包含一两个应用。它是通过一个向虚拟机提供了硬件虚拟化的Hypervisor来实现的，允许单个服务器运行很多独立的被当做虚拟游客（virtual guest）的操作系统。

而容器与虚拟机的类似之处在于它们允许单个服务器运行多个操作环境（operating environment），然而这些环境不却是完整的操作系统。容器通常只包含必要的OS软件包和应用。他们通常不包含一个完整的操作系统或者硬件虚拟化。这也意味着比之虚拟机，容器的额外开销（overhead）更小。

> 容器和虚拟机通常被视为不能共生的技术，然而这通常是一个误解。虚拟机面向物理服务器，提供可以能与其他虚拟机一起共享这些物理资源的，功能完善的操作环境。容器通常是用来通过对单一主机的一个进程进行隔离，来保证被隔离的进程无法与处于同一个系统的其他进程进行互动。实际上，比起完全的虚拟机，容器与BSD的Jail，chroot的进程更加类似。

Docker 自身并不是一个容器的运行时环境。 Docker 提供的是一种容器管理，打包和部署的方法。尽管这种类型的功能已经某一种程度地存在于虚拟机中，但在传统上，它们并不是为了绝大多数的容器方案而生的，而那些已经存在的，却又不如 Docker 一样容易使用且功能完善。

安装 Docker

```
sudo apt install docker
sudo apt install docker.io
```

安装成功后，可以使用 `docker ps` 命令来检测是否正确安装：

```
longj@longj-Daydream:~$ docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS
```

有时候会出现下面报错：

```
ubuntu@VM-100-106-ubuntu:~$ docker ps
Cannot connect to the Docker daemon. Is the docker daemon running on this host?
```

这个时候只需要在命令前面加上 `sudo` 即可。

Docker 镜像

首先需要分清楚 docker image (镜像) 和 docker container (容器) 的区别：一个**docker container 是一个特定 docker image 的运行实例。** 我们先要在宿主机 (host) 上获取到一个 docker image， 然后再通过特定的 docker 命令来构建并运行一个 docker container。可以将 docker image 类比成你平时安装 Ubuntu 虚拟机时首先需要下载的 Ubuntu 镜像，将 docker container 类比成安装 Ubuntu 虚拟机的实例，显然，通过同一个镜像，你是可以安装多个互不影响的虚拟机的。

其中获得一个 docker image 的方法有两个：

1. 在 Dockhub 上直接 pull 下别人已经上传好的镜像。 `docker pull [OPTIONS] NAME[:TAG|@DIGEST]`
2. 自己写 Dockerfile 去构建一个定制的镜像。

现在我们的目标是要拿到一个 Python Flask 的镜像并且进行部署运行一个实例。

构建自定义的 Python Flask 应用镜像

为了Docker化一个 Python Flask 应用镜像，我们将要构建我们自己的Docker镜像，这意味要创建一个Dockerfile。

在绝大多数的虚拟机环境中，假如你想要创建一个机器的镜像，你需要首先创建一个虚拟机，然后安装好操作系统，然后安装好应用程序，最后将其转化成一个模板或者镜像。然而，对于Docker来说，这些步骤都可以通过 Dockerfile 进行自动化。一个 Dockerfile 是一个可以向 Docker 提供构建指令的方式。我们将要创建一个可以用来部署一个最简单的 Flask 应用镜像的 Dockerfile 。

Step 1 理解应用

项目内容和结构如下：

```
Web
  ____ app.py
  ____ requirements.txt
  ____ Dockerfile
```

app.py :

```
from flask import Flask
app = Flask(__name__)
@app.route('/')
def hello_world():
    return 'Hello World!'
if __name__ == '__main__':
    app.run(debug=True,host='0.0.0.0')
```

requirements.txt:

```
Flask>=0.10
```

Step 2 为你的应用写一个 Dockerfile

1. 用 `From` 来继承一个 Docker 镜像

Dockerfile 的第一条指令是FROM指令。这用来将一个已经存在的 Docker 镜像指定为基础镜像。这基本上为我们提供了继承另一个Docker镜像的方法。在我们这个场景中，我们可以通过指定 `ubuntu:16.04` 使用 Ubuntu 16.04镜像。

```
FROM ubuntu:16.04
MAINTAINER longjj <blablabla@gmail.com>
ENV LANG C
```

除了FROM指令之外，还包含了一个MAINTAINER指令，其是用来显示 Dockerfile 的作者的。然后，使用 `ENV LANG C` 来解决后面安装 python3 时出现的 locale 问题。

2. 安装 Python 3 开发环境（亦可以选择 Python 2.7）

```
RUN apt-get update \
  && apt-get install -y python3-pip python3-dev locales\
  && pip3 install --upgrade pip
```

3. 将当前目录里面的代码文件拷贝到镜像里面，并且安装 Python 依赖

```
COPY . /app
WORKDIR /app
RUN pip3 install -r requirements.txt
```

4. 设置 `EntryPoint` 去告诉 Docker 在这个 docker container 中运行 `python3` 的命令，执行 `app.py`

```
ENTRYPOINT ["python3"]
CMD ["app.py"]
```

5. 总体的 Dockerfile 如下

```
## Dockerfile that generates an instance of www.longjj.com
FROM ubuntu:16.04
LABEL maintainer="longjj"
ENV LANG C

RUN apt-get update \
  && apt-get install -y python3-pip python3-dev locales\
  && pip3 install --upgrade pip

COPY . /app
WORKDIR /app
RUN pip3 install -r requirements.txt
ENTRYPOINT ["python3"]
CMD ["app.py"]
```

Step 3 构建你的 Docker Image

在 Dockerfile 所在目录下运行下面的命令。

```
$ docker build -t helloworldapp:latest .
```

Step 4 使用你构建的镜像运行一个 Docker Container

```
$ docker run -d -p 5000:5000 helloworldapp
```

`-d` （detach，脱离）标志是用来告诉 Docker 在后台运行容器；另一个标志是 `-p`，这个标志能让用户来将一个端口从主机机器映射到容器中的一个端口。

我们使用的 Flask 应用默认暴露了 5000 端口来提供HTTP服务。默认情况下，与Docker容器内部绑定的端口并没有与主机系统绑定。为了让外部的系统访问容器内部暴露的端口，这些端口必须通过使用 `-p` 标志从主机端口映射到容器端口。假如我们想要端口从主机的 8080 端口，映射到容器中的 5000 端口，我们可以通过使用这种语法 `-p 8080:80` 。

从上面的命令中，看起来我们的容器已经启动成功了。我们可以通过运行执行 `docker ps` 来验证。

```
ubuntu@VM-100-106-ubuntu:~$ sudo docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED
STATUS              PORTS                                       NAMES
de0b737586e6        tiny-hippo          "python3 app.py"         5 days ago
Up 5 days           0.0.0.0:5000->5000/tcp                      hippo
```

在浏览器中访问 http://localhost:5000 (http://localhost:5000) , 你现在应该能够拿到 Python Flask 服务器返回的信息。

参考文献

1. Getting started with Docker by Dockerizing this Blog (http://bencane.com/2015/12/01/getting-started-with-docker-by-dockerizing-this-blog/#conversation)

2. Dockerize Simple Python Flask App (https://medium.com/@ikod/dockerize-simple-python-flask-app-62461efbe58e)
