Docker 微服务教程

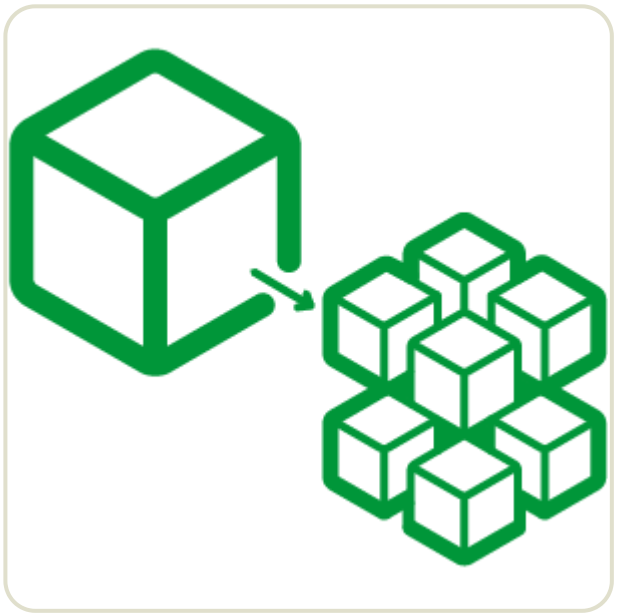
作者： 阮一峰

日期： 2018年2月13日

Docker 是一个容器工具，提供虚拟环境。很多人认为，它改变了我们对软件的认识。

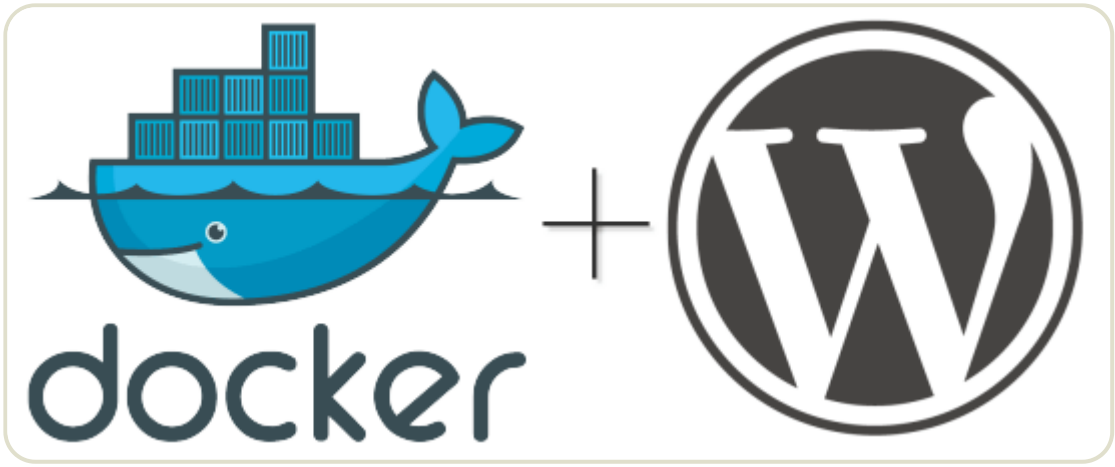
站在 Docker 的角度，软件就是容器的组合：业务逻辑容器、数据库容器、储存容器、队列容器.....Docker 使得软件可以拆分成若干个标准化容器，然后像搭积木一样组合起来。

这正是微服务（microservices）的思想：软件把任务外包出去，让各种外部服务完成这些任务，软件本身只是底层服务的调度中心和组装层。



微服务很适合用 Docker 容器实现，每个容器承载一个服务。一台计算机同时运行多个容器，从而就能很轻松地模拟出复杂的微服务架构。

上一篇教程介绍了 Docker 的概念和基本用法，本文接着往下介绍，如何在一台计算机上实现多个服务，让它们互相配合，组合出一个应用程序。



我选择的示例软件是 WordPress。它是一个常用软件，全世界用户据说超过几千万。同时它又非常简单，只要两个容器就够了（业务容器 + 数据库容器），很适合教学。而且，这种"业务 + 数据库"的容器架构，具有通用性，许多应用程序都可以复用。

为了加深读者理解，本文采用三种方法，演示如何架设 WordPress 网站。

- 方法 A：自建 WordPress 容器
- 方法 B：采用官方的 WordPress 容器
- 方法 C：采用 Docker Compose 工具

一、预备工作：image 仓库的镜像网址

本教程需要从仓库下载 image 文件，但是国内访问 Docker 的官方仓库很慢，还经常断线，所以要把仓库网址改成国内的镜像站。这里推荐使用官方镜像 registry.docker-cn.com 。下面是我的 Debian 系统的默认仓库修改方法，其他系统的修改方法参考官方文档。

打开 /etc/default/docker 文件（需要 sudo 权限），在文件的底部加上一行。

```
DOCKER_OPTS="--registry-mirror=https://registry.docker-cn.com"
```

然后，重启 Docker 服务。

```
$ sudo service docker restart
```

现在就会自动从镜像仓库下载 image 文件了。

二、方法 A：自建 WordPress 容器

前面说过，本文会用三种方法演示 WordPress 的安装。第一种方法就是自建 WordPress 容器。

2.1 官方 的 PHP image

首先，新建一个工作目录，并进入该目录。

```
$ mkdir docker-demo && cd docker-demo
```

然后，执行下面的命令。

```
$ docker container run \
  --rm \
  --name wordpress \
  --volume "$PWD/":/var/www/html \
  php:5.6-apache
```

上面的命令基于 `php` 的 image 文件新建一个容器，并且运行该容器。 `php` 的标签是 `5.6-apache` ，说明装的是 PHP 5.6，并且自带 Apache 服务器。该命令的三个参数含义如下。

- `--rm`：停止运行后，自动删除容器文件。
- `--name wordpress`：容器的名字叫做wordpress。
- `--volume "$PWD/":/var/www/html`：将当前目录（`$PWD`）映射到容器的`/var/www/html`（`Apache` 对外访问的默认目录）。因此，当前目录的任何修改，都会反映到容器里面，进而被外部访问到。

运行上面的命令以后，如果一切正常，命令行会提示容器对外的 IP 地址，请记下这个地址，我们要用它来访问容器。我分配到的 IP 地址是 172.17.0.2。

打开浏览器，访问 172.17.0.2，你会看到下面的提示。

```
Forbidden
You don't have permission to access / on this server.
```

这是因为容器的 `/var/www/html` 目录（也就是本机的 `docker-demo` 目录）下面什么也没有，无法提供可以访问的内容。

请在本机的 `docker-demo` 目录下面，添加一个最简单的 PHP 文件 `index.php` 。

```
<?php
phpinfo();
?>
```

保存以后，浏览器刷新 `172.17.0.2` ，应该就会看到熟悉的 `phpinfo` 页面了。

2.2 拷贝 WordPress 安装包

既然本地的 `docker-demo` 目录可以映射到容器里面，那么把 WordPress 安装包拷贝到 `docker-demo` 目录下，不就可以通过容器访问到 WordPress 的安装界面了吗？

首先，在 `docker-demo` 目录下，执行下面的命令，抓取并解压 WordPress 安装包。

```
$ wget https://cn.wordpress.org/wordpress-4.9.4-zh_CN.tar.gz
$ tar -xvf wordpress-4.9.4-zh_CN.tar.gz
```

解压以后，WordPress 的安装文件会在 `docker-demo/wordpress` 目录下。

这时浏览器访问 `http://172.17.0.2/wordpress` ，就能看到 WordPress 的安装提示了。

2.3 官方的 MySQL 容器

WordPress 必须有数据库才能安装，所以必须新建 MySQL 容器。

打开一个新的命令行窗口，执行下面的命令。

```
$ docker container run \
  -d \
  --rm \
  --name wordpressdb \
  --env MYSQL_ROOT_PASSWORD=123456 \
  --env MYSQL_DATABASE=wordpress \
  mysql:5.7
```

上面的命令会基于 MySQL 的 image 文件（5.7版本）新建一个容器。该命令的五个命令行参数的含义如下。

- `-d`：容器启动后，在后台运行。
- `--rm`：容器终止运行后，自动删除容器文件。
- `--name wordpressdb`：容器的名字叫做`wordpressdb`
- `--env MYSQL_ROOT_PASSWORD=123456`：向容器进程传入一个环境变量`MYSQL_ROOT_PASSWORD`，该变量会被用作 MySQL 的根密码。
- `--env MYSQL_DATABASE=wordpress`：向容器进程传入一个环境变量`MYSQL_DATABASE`，容器里面的 MySQL 会根据该变量创建一个同名数据库（本例是`WordPress`）。

运行上面的命令以后，正常情况下，命令行会显示一行字符串，这是容器的 ID，表示已经新建成功了。

这时，使用下面的命令查看正在运行的容器，你应该看到 `wordpress` 和 `wordpressdb` 两个容器正在运行。

```
$ docker container ls
```

其中， `wordpressdb` 是后台运行的，前台看不见它的输出，必须使用下面的命令查看。

```
$ docker container logs wordpressdb
```

2.4 定制 PHP 容器

现在 WordPress 容器和 MySQL 容器都已经有了。接下来，要把 WordPress 容器连接到 MySQL 容器了。但是，PHP 的官方 image 不带有 `mysql` 扩展，必须自己新建 image 文

件。

首先，停掉 WordPress 容器。

```
$ docker container stop wordpress
```

停掉以后，由于 `--rm` 参数的作用，该容器文件会被自动删除。

然后，在 `docker-demo` 目录里面，新建一个 `Dockerfile` 文件，写入下面的内容。

```
FROM php:5.6-apache
RUN docker-php-ext-install mysqli
CMD apache2-foreground
```

上面代码的意思，就是在原来 PHP 的 image 基础上，安装 `mysqli` 的扩展。然后，启动 Apache。

基于这个 Dockerfile 文件，新建一个名为 `phpwithmysql` 的 image 文件。

```
$ docker build -t phpwithmysql .
```

2.5 Wordpress 容器连接 MySQL

现在基于 phpwithmysql image，重新新建一个 WordPress 容器。

```
$ docker container run \
  --rm \
  --name wordpress \
  --volume "$PWD/":/var/www/html \
  --link wordpressdb:mysql \
  phpwithmysql
```

跟上一次相比，上面的命令多了一个参数 `--link wordpressdb:mysql` ，表示 WordPress 容器要连到 `wordpressdb` 容器，冒号表示该容器的别名是 `mysql` 。

这时还要改一下 `wordpress` 目录的权限，让容器可以将配置信息写入这个目录（容器内部写入的 `/var/www/html` 目录，会映射到这个目录）。

```
$ chmod -R 777 wordpress
```

接着，回到浏览器的 `http://172.17.0.2/wordpress` 页面，点击"现在就开始！"按钮，开始安装。

WordPress 提示要输入数据库参数。输入的参数如下。

- 数据库名：`wordpress`
- 用户名：`root`
- 密码：`123456`
- 数据库主机：`mysql`
- 表前缀：`wp_`（不变）

点击"下一步"按钮，如果 Wordpress 连接数据库成功，就会出现下面的页面，这就表示可以安装了。

至此，自建 WordPress 容器的演示完毕，可以把正在运行的两个容器关闭了（容器文件会自动删除）。

```
$ docker container stop wordpress wordpressdb
```

三、方法 B：Wordpress 官方镜像

上一部分的自建 WordPress 容器，还是挺麻烦的。其实不用这么麻烦，Docker 已经提供了官方 WordPress image，直接用那个就可以了。有了上一部分的基础，下面的操作就很容易理解了。

3.1 基本用法

首先，新建并启动 MySQL 容器。

```
$ docker container run \
  -d \
  --rm \
  --name wordpressdb \
  --env MYSQL_ROOT_PASSWORD=123456 \
  --env MYSQL_DATABASE=wordpress \
  mysql:5.7
```

然后，基于官方的 WordPress image，新建并启动 WordPress 容器。

```
$ docker container run \
  -d \
  --rm \
  --name wordpress \
  --env WORDPRESS_DB_PASSWORD=123456 \
  --link wordpressdb:mysql \
  wordpress
```

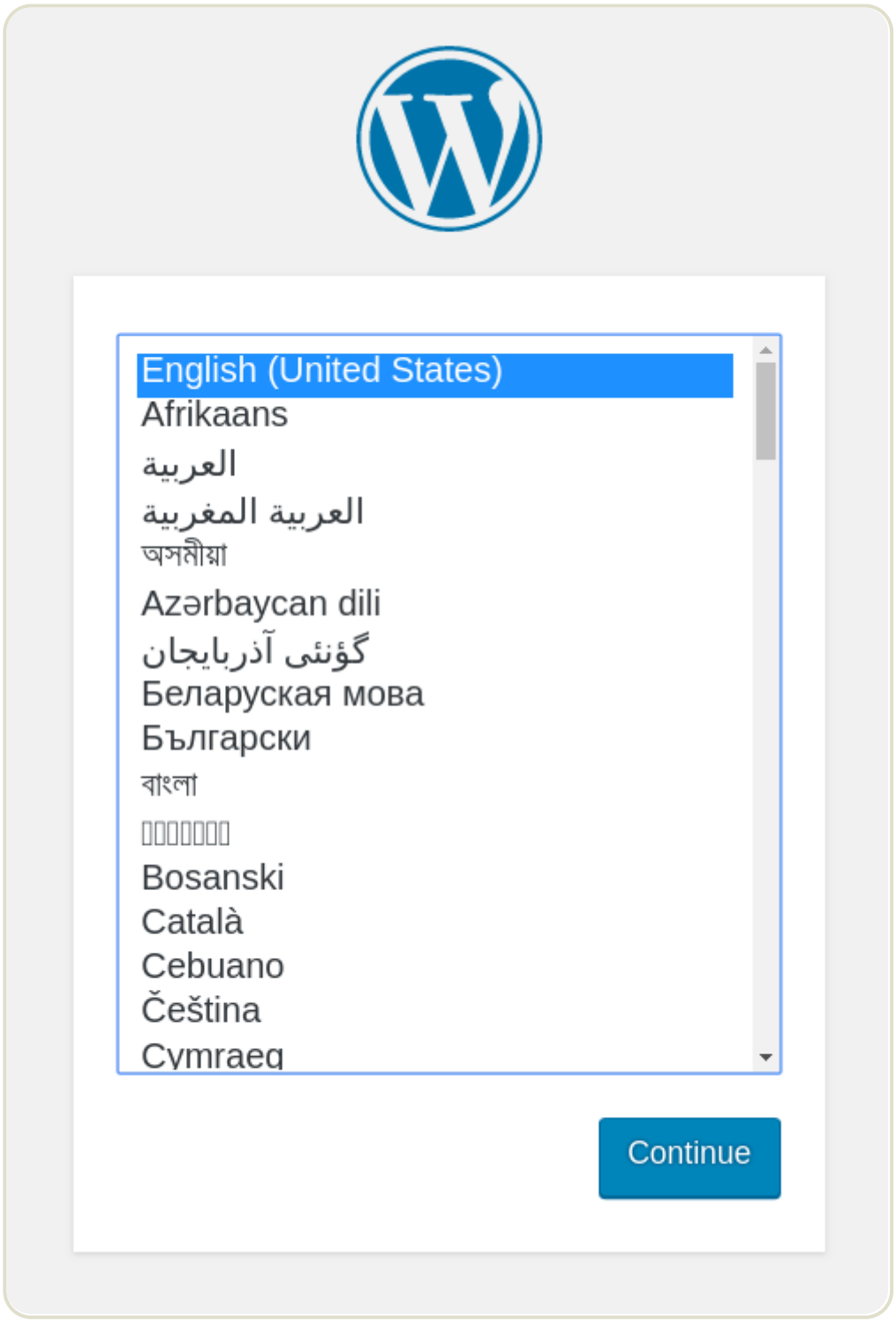
上面命令中，各个参数的含义前面都解释过了，其中环境变量 `WORDPRESS_DB_PASSWORD` 是 MySQL 容器的根密码。

上面命令指定 `wordpress` 容器在后台运行，导致前台看不见输出，使用下面的命令查出 `wordpress` 容器的 IP 地址。

```
$ docker container inspect wordpress
```

上面命令运行以后，会输出很多内容，找到 `IPAddress` 字段即可。我的机器返回的 IP 地址是 `172.17.0.3` 。

浏览器访问 `172.17.0.3` ，就会看到 WordPress 的安装提示。



3.2 WordPress 容器的定制

到了上一步，官方 WordPress 容器的安装就已经成功了。但是，这种方法有两个很不方便的地方。

- 每次新建容器，返回的 IP 地址不能保证相同，导致要更换 IP 地址访问 WordPress。
- WordPress 安装在容器里面，本地无法修改文件。

解决这两个问题很容易，只要新建容器的时候，加两个命令行参数就可以了。

先把刚才启动的 WordPress 容器终止（容器文件会自动删除）。

```
$ docker container stop wordpress
```

然后，使用下面的命令新建并启动 WordPress 容器。

```
$ docker container run \
 -d \
 -p 127.0.0.2:8080:80 \
 --rm \
 --name wordpress \
 --env WORDPRESS_DB_PASSWORD=123456 \
 --link wordpressdb:mysql \
 --volume "$PWD/wordpress":/var/www/html \
 wordpress
```

上面的命令跟前面相比，命令行参数只多出了两个。

- `-p 127.0.0.2:8080:80`：将容器的 80 端口映射到127.0.0.2的8080端口。
- `--volume "$PWD/wordpress":/var/www/html`：将容器的/var/www/html目录映射到当前目录的wordpress子目录。

浏览器访问 127.0.0.2:8080:80 就能看到 WordPress 的安装提示了。而且，你在 wordpress 子目录下的每次修改，都会反映到容器里面。

最后，终止这两个容器（容器文件会自动删除）。

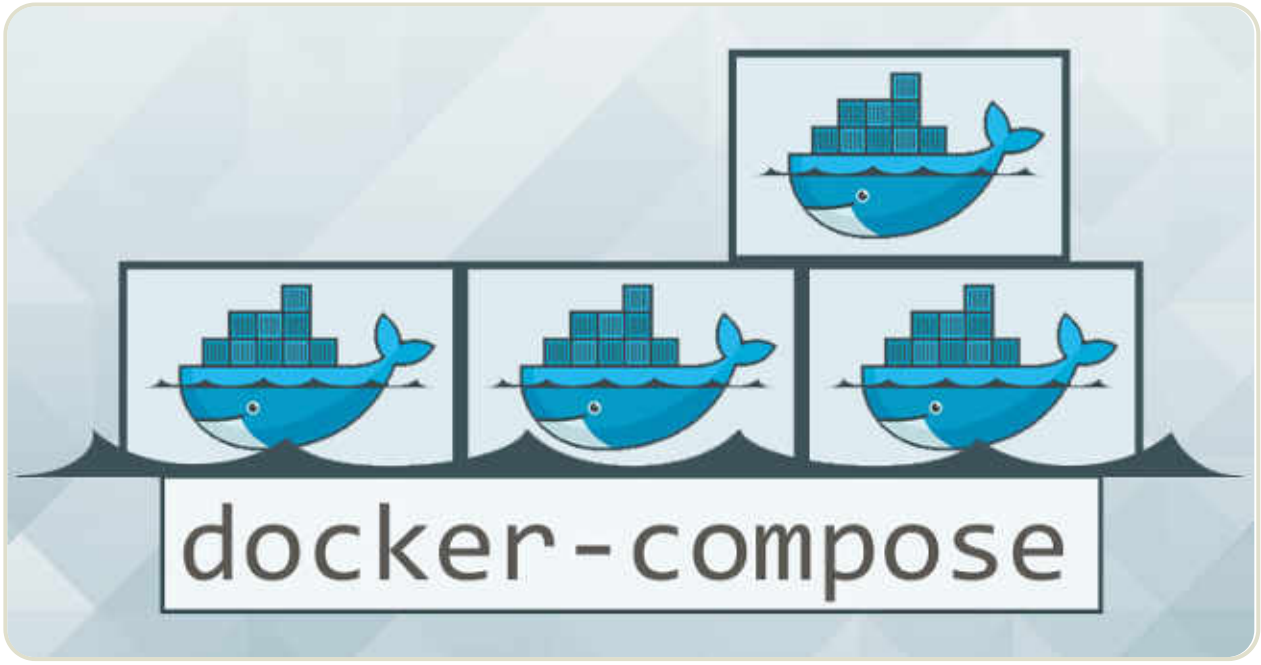
```
$ docker container stop wordpress wordpressdb
```

四、方法 C：Docker Compose 工具

上面的方法 B 已经挺简单了，但是必须自己分别启动两个容器，启动的时候，还要在命令行提供容器之间的连接信息。因此，Docker 提供了一种更简单的方法，来管理多个容器的联

动。

4.1 Docker Compose 简介



Compose 是 Docker 公司推出的一个工具软件，可以管理多个 Docker 容器组成一个应用。你需要定义一个 YAML 格式的配置文件 `docker-compose.yml` ，写好多个容器之间的调用关系。然后，只要一个命令，就能同时启动/关闭这些容器。

```
# 启动所有服务
$ docker-compose up
# 关闭所有服务
$ docker-compose stop
```

4.2 Docker Compose 的安装

Mac 和 Windows 在安装 docker 的时候，会一起安装 docker compose。Linux 系统下的安装参考官方文档。

安装完成后，运行下面的命令。

```
$ docker-compose --version
```

4.3 WordPress 示例

在 `docker-demo` 目录下，新建 `docker-compose.yml` 文件，写入下面的内容。

```
mysql:
    image: mysql:5.7
```

```
    environment:
      - MYSQL_ROOT_PASSWORD=123456
      - MYSQL_DATABASE=wordpress
web:
    image: wordpress
    links:
      - mysql
    environment:
      - WORDPRESS_DB_PASSWORD=123456
    ports:
      - "127.0.0.3:8080:80"
    working_dir: /var/www/html
    volumes:
      - wordpress:/var/www/html
```

上面代码中，两个顶层标签表示有两个容器 `mysql` 和 `web` 。每个容器的具体设置，前面都已经讲解过了，还是挺容易理解的。

启动两个容器。

```
$ docker-compose up
```

浏览器访问 http://127.0.0.3:8080，应该就能看到 WordPress 的安装界面。

现在关闭两个容器。

```
$ docker-compose stop
```

关闭以后，这两个容器文件还是存在的，写在里面的数据不会丢失。下次启动的时候，还可以复用。下面的命令可以把这两个容器文件删除（容器必须已经停止运行）。

```
$ docker-compose rm
```

五、参考链接

- How to Manually Build Docker Containers for WordPress, by Aleksander Koko
- How to Use the Official Docker WordPress Image, by Aleksander Koko
- Deploying WordPress with Docker, by Aleksander Koko

（完）

文档信息

相关文章

- **2019.03.16:** 普通人的网页配色方案

  > 网页需要配色。一种好看、易用、符合心意的配色，是很不容易的，尤其在没有设计师时。

- **2019.03.06:** 敏捷开发入门教程

  > 敏捷开发（agile development）是非常流行的软件开发方法。据统计，2018年90%的软件开发采用敏捷开发。

- **2019.02.18:** ORM 实例教程

  > 一、概述 面向对象编程和关系型数据库，都是目前最流行的技术，但是它们的模型是不一样的。

- **2019.02.07:** 找回密码的功能设计

  > 所有需要登录的网站，都会提供"找回密码"的功能，防止用户忘记密码。

广告（购买广告位）