Docker 容器编排利器：Docker-Compose

作者：liumapp
原文链接：https://ld246.com/article/1524101853730
来源网站：链滴

一个项目发布到Docker中，我们往往需要先编写Dockerfile，在这个里面操作Image的安装，Contai
er的加载等操作，但像Spring Cloud这种一个项目下N个子服务（每个子服务独占一个进程）的大型
目，难道每一个子服务下都去编写Dockerfile吗？而且这里面还牵涉到子服务与子服务之间的通讯在D
cker下如何去配置的问题。这种情况下，我们使用Docker-Compose是最适合不过的了。

1. 前言

首先上Github上的项目展示案例 liumapp/docker-compose-demo

接下来，我们简单介绍一下docker-compose的定义：

Docker Compose是Docker三剑客的最后一个，第一个是Machine，用来在不同平台下快速安装Doc
er的，第二个是Swarm，帮助Docker在集群中运转，第三个便是Docker Compose，用来帮助用户
行容器组（请注意，不是单独的容器）。

2. 使用

那么我们如何来使用Docker Compose呢?

在案例 liumapp/docker-compose-demo中，我们使用Spring Cloud项目集群为例，来介绍如何通
Docker Compose将一个Spring Cloud项目群部署到Docker中。

2.1操作步骤

2.1.1 安装Image

第一步跟Docker Compose的关系不大，安装Image，在案例中我们使用了docker-maven-plugin
一个插件来操作，然后编写了一个build-image.sh的脚本，一键完成所有项目的Image安装。

docker-maven-plugin的配置如下：

```
<plugin>
  <groupId>com.spotify</groupId>
  <artifactId>docker-maven-plugin</artifactId>
  <version>1.0.0</version>
  <configuration>
    <imageName>liumapp/${project.artifactId}:${project.version}</imageName>
    <baseImage>java:8</baseImage>
    <entryPoint>["java", "-jar", "/${project.build.finalName}.jar"]</entryPoint>
    <resources>
      <resource>
        <targetPath>/</targetPath>
        <directory>${project.build.directory}</directory>
        <include>${project.build.finalName}.jar</include>
      </resource>
    </resources>
  </configuration>
</plugin>
```

build-image.sh的内容就不粘贴了，基本就是分别进入各个子服务的目录下，再依次执行

```
mvn clean package docker:build
```

来完成Image的安装。

2.1.2 编写docker-compose.yml

docker-compose.yml应该位于项目的根目录下，且在命名上不能有偏差。

在编写docker-compose.yml时，我们要注意，要定义的容器组位于services下，每一个services的I一般跟子项目的artifactId保持一致，并在其下再去定义所采用的Image、启动失败时是否重启、容器称、主机名称、及监听和开放的端口号、依赖的服务等等内容。

具体对应的参数项为：

```
image: ${您采用的Image}
restart: ${是否在失败时重启，是的话为always}
container_name: ${运行时的容器名称}
hostname: ${配置网络的主机名称，可用于容器组内的通讯}
ports:
  - "1234:1234" ${监听和开放的端口号}
depends_on:
  - docker-compose-eureka ${依赖的服务}
```

那么案例中的docker-compose.yml的内容就不粘贴了，比较多，大家请去项目中查阅。

2.1.3 启动容器组

安装好Image后，我们可以使用：

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

来启动，-d表示以守护进程的形式运行。

2.1.4 停止容器组

想要关停容器组，我们可以使用：

```
docker-compose down
```

来停止。

2.1.5 删除Image

案例中我们使用了rm-images.sh脚本来删除所安装的Image，其内容为:

```
docker rmi liumapp/docker-compose-config:v1.0.0
docker rmi liumapp/docker-compose-eureka:v1.0.0
docker rmi liumapp/docker-compose-gateway:v1.0.0
docker rmi liumapp/demo-api-a:v1.0.0
```

```
docker rmi liumapp/demo-api-b:v1.0.0
```