2021.3.12 Docker 编排与部署工具 Compose

、Docker Composet 介绍

Docker Compose 的前身是 Fig，它是一个**定义及运行多个 Docker 容器的工具**。可以使用 YAML 文件来配置应用程序的服务。然后，使用单个命令，您可以创建并启动配置中的所有服务。**Docker Compose 会通过解析容器间的依赖关系按先后顺序启动所定义的容器**（link，网络容器-net-from 或数据容器-volume-from）。帮助实现复杂多容器关联的环境。

Compose 是 Docker 的服务编排工具，主要用来构建基于 Docker 的复杂应用，Compose 通过一个配置文件来管理多个 Docker 容器，非常适合组合使用多个容器进行开发的场景。

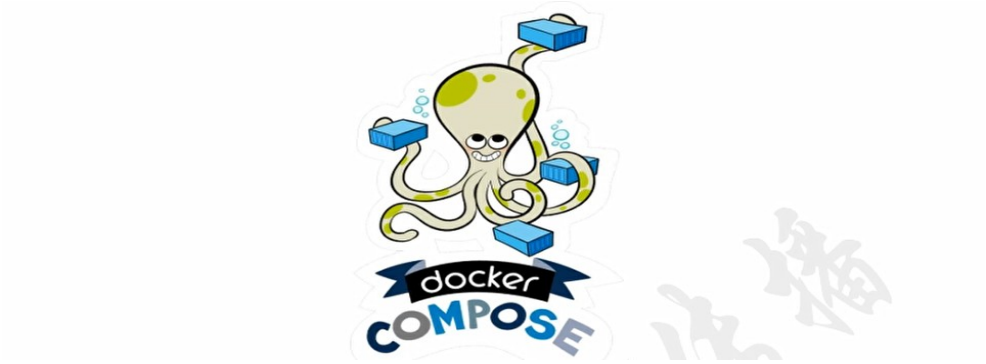
Compose 适用于所有环境：生产，开发，测试以及 CI（持续集成）工作流程。使用 Compose 基本上是一个三步过程：

1、使用 **Dockerfile** 定义应用程序的环境，以便在任何地方进行复制。

2、在 **docker-compose.yml** 中定义组成应用程序的服务，以便它们可以在隔离的环境中一起运行。

3、运行 **docker-compose** 开始并运行整个应用程序。

可以参考 docker-compose 官方说明详细了解：https://docs.docker.com/compose/overview/



docker-compose 是用来做 docker 的多容器控制，有了 docker-compose 你可以把所有繁复的 docker 操作全都用一条命令自动化完成。从上图可以看到，这位 compose 非常开心的把 N 多个容器抓在一起，根据自己的心情来编排部署。

Docker 对于运维或开发者来说，Docker 最大的优点在于它提供了一种全新的发布机制。这种发布机制，指的是我们使用 Docker 镜像作为统一的软件制品载体，使用 Docker 容器提供独立的软件运行上下文环境，使用 Docker Hub 提供镜像统一协作，最重要的是该机制使用 Dockerfile 定义容器内部行为和容器关键属性来支撑软件运行。

Dockerfile 作为整个机制的核心。在 Dockerfile 中不但能够定义使用者在容器中需要进行的操作，而且能够定义容器中运行软件需要的配置，于是软件开发和运维终于能够在一个配置文件上达成统一。运维人员使用同一个 Dockerfile 能在不同的场合下"重现"与开发煮环境中一模一样的运行单元（Docker 容器）出来。

二、Compose 的优点

先来了解一下我们平时是怎么样使用 docker 的？把它进行拆分一下：

1、docker search 镜像，是不是先查找一个镜像；

2、docker run -itd 镜像名称，然后在运行这个镜像；

3、然后如果你要在运行第二个镜像、第三个镜像等等，你是不是又要 docker search、docker run 运行。

上面"docker run -id 镜像名称"这只是最小的动作，如果你要映射硬盘，设置 nat 网络或者映射端口等等。就要做更多的 docker 操作，这显然是非常没有效率的，况且如果你要大规模部署，是不是觉得就很麻烦了。

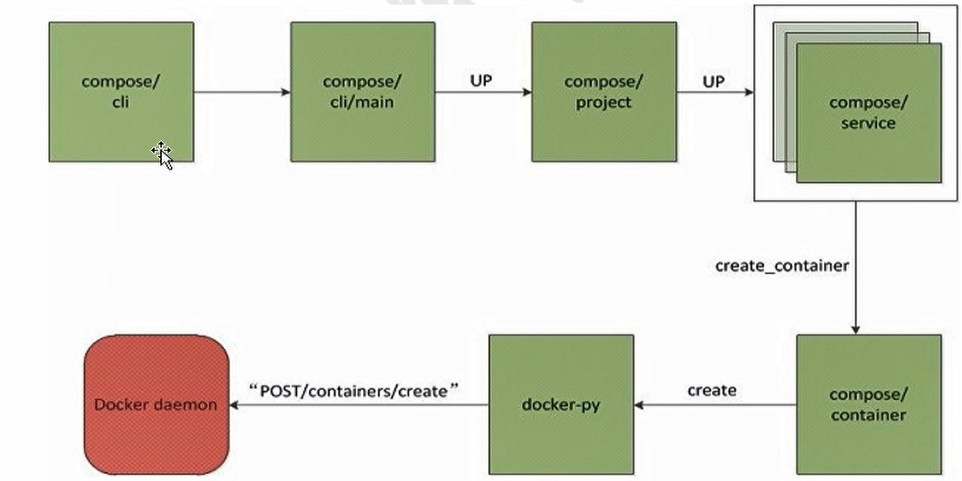
但是我们写在 docker-compose 里面就很好了。你只需要写好后只运行一句：docker-compose up -d

三、编排和部署

编排，即 orchestration，它根据被部署的对象之间的耦合关系，以及被部署对象环境的依赖，制定部署流程中各个动作的执行顺序，部署过程所需要的依赖文件的存储位置和获取方式，以及如何验证部署成功。这些信息都会在编排工具中以指定的格式（比如配置文件或者特定的代码）来要求运维人员定义并保存起来，从而保证这个流程能够随时在全新的环境中可靠有序地重现出来。

部署，即 deployment，它是按照编排所指定的内容和流程，在目标机器上执行编排指定环境初始化，存放指定的依赖和文件，运行指定的部署动作，最终按照编排中的规则来确认联署成功。

这么来解释吧，**编排**是一个指挥家，他的大脑里存储了整个乐曲的演奏流程，对于每一个小节每一段音乐的演奏方式、开始、结束他都了然于胸；**部署**就是整个乐队，他们严格按照指挥家的意图用乐器来完成乐谱的执行，在需要时开始演奏，又在适当的时机停止演奏。最终，两者通过协作就能把每一位演奏者独立的演奏通过组合、重叠、衔接来形成高品位的交响乐。



首先，用户执行的 docker-compose up 指令调用了命令行中的启动方法。功能很简单明了，一个 docker-compose.yml 定义了一个 docker-compose 的 project，docker-compose 操作提供的命令行参数则作为这个 project 的启动参数交由 project 模块去处理。

其次，如果当前宿主机已经存在与该应用对应的容器，docker-compose 将进行行为逻辑判断。如果用户指定可以重新启动已有服务，docker-compose 就会执行 service 模块的容器重启方法，否则就将直接启动已有容器。这两种操作的区别在于前者会停止旧的容器，创建启动新的容器，并把旧容器移除掉。在这个过程中创建容器的各项定义参数都是从

docker-compose up 指令和 docker-compose.yml 中传入的。

接下来，启动容器的方法也很简洁，这个方法中完成了一个 Docker 容器启动所需的主要参数的封装，并在 container 模块执行启动。该方法所支持的参数我想大多数朋友过是有所了解的。

最后，container 模块会调用 docker-py 客户端执行向 Docker daemon 发起创建容器的 POST 请求，再往后就是 Docker 处理的范畴了，相信看过我这篇文章 Docker：架构拆解请的朋友就明白了。

为了能够说明 compose 如何实现上述编排与部署的原理，下面和大家分享一个通过 compose 来编排部署 LNMP 服务来更好的理解它。

五、Compose 应用案例

（一）安装 docker-ce

```
[root@localhost ~]# cd /etc/yum.repos.d/
[root@localhost ~]# 下载一个阿里云的 yum 源
Wget -O /etc/yum.repos.d/CentOS-Base.repo http://mirrors.aliyun.com/repo/Centos-7.repo

[root@localhost ~]# yum -y install yum-utils device-mapper-persistent-data lvm2  //安装依赖包
[root@localhost ~]#
yum-config-manager  --add-repo  http://mirrors.aliyun.com/docker-ce/linux/centos/docker-ce.repo

[root@localhost ~]# ls /etc/yum.repos.d/
backup  Centos-aliyun.repo  CentOS-Media.repo  docker-ce.repo

[root@localhost ~]# yum -y install docker-ce
[root@localhost ~]# systemctl start docker
[root@localhost ~]# systemctl enable docker
[root@localhost ~]# docker --version   //或者 docker  version   //查看 docker 版本信息
Docker version 20.10.5, build 55c4c88
```

方法一：阿里云镜像加速（推荐）

```
[root@localhost ~]# sudo tee /etc/docker/daemon.json <<-'EOF'
{
  "registry-mirrors": ["https://317k7ifs.mirror.aliyuncs.com"]
}
EOF
[root@localhost ~]# sudo systemctl daemon-reload
[root@localhost ~]# sudo systemctl restart docker
[root@localhost ~]# docker info      //查看 docker 的详细信息
 Registry Mirrors:   //镜像加速
  https://317k7ifs.mirror.aliyuncs.com/
```

```
Live Restore Enabled: false
```

方法二：http://get.daocloud.io/镜像加速

```
curl -sSL https://get.daocloud.io/daotools/set_mirror.sh | sh -s http://f1361db2.m.daocloud.io
```

（二）安装 docker-composek

下载最新版本安装，下载时间可能比较长

```
[root@localhost ~]# rz     //docker-compose
[root@localhost ~]# mv docker-compose /usr/local/bin/
[root@localhost ~]# chmod +x /usr/local/bin/docker-compose
[root@localhost ~]# docker-compose --version
docker-compose version 1.21.1, build 5a3f1a3
```

（三）docker-compose 用法

docker-compose [-f <arg>...] [options] [COMMAND][ARGS...]

docker-compose 常用选项：

·--verbose 输出更多调试信息。

·--version 打印版本并退出。

·-f，-file FILE 使用特定的 compose 模板文件，默认为 docker-compose.yml

·-p，--project-name NAME 指定项目名称，默认使用目录名称。

docker-compose 常用命令：（和 docker 的命令差不多）

·build 构建或重建服务

·kill 杀掉容器

·logs 显示容器的输出内容

·port 打印绑定的开放端口

·ps 显示容器

·pull 拉取服务镜像

·restart 重启服务

·rm 删除停止的容器

·run 运行一个一次性命令

·scale 设置服务的容器数目

·exec 切换到容器内

·start 开启服务

·stop 停止服务

·up 创建并启动容器

（四）Yaml 简介

YAML 是一种标记语言，可读性很强。类似于 xML 数据描述语言，语法比 XML 简单的

多。YAML 数据结构通过缩进来表示，连续的项目通过诚号来表示，键值对用冒号分割，数组用括号括起来，hash 用花括号括起来。

YAML 文件格式注意事项：

在缩进中空白字符的数目并不是非常重要，只要相同阶层的元素左侧对齐就可以了（不过不能使用 TAB 字符）；

·通常开头缩进 2 个空格；

·字符的后面缩进 1 个空格，比如冒号、逗号、横杆；

·支持#注释；

·允许在文件中加入选择性的空行，以增加可读性；

docker-compose 中 YAML 常用的字段：

字段	描述
build dockerfile context	指定dockerfile文件名 构建镜像上下文路径
image	指定镜像
command	执行命令，覆盖默认命令
container_name	指定容器名称，由于容器名称是唯一的，如果指定自定义名称，则无法scale
deploy	指定部署和运行服务相关配置，只能在Swarm模式使用
environment	添加环境变量
networks	加入网络，引用顶级networks下条目
ports	暴露端口，与-p相同，但端口不能低于60
volumes	挂载宿主机路径或命名卷。如果是命名卷在顶级volumes定义卷名称
restart	重启策略，默认no，always|on-failure|unless-stopped
hostname	容器主机名


```
[root@localhost ~]# rz  //compose_lnmp-190606.zip
[root@localhost ~]# rz  //wordpress-4.9.4-zh_CN.tar.gz
[root@localhost ~]# rz  //centos-7-x86_64.tar.gz
[root@localhost ~]# unzip compose_lnmp-190606.zip
[root@localhost ~]# cd compose_lnmp/
[root@localhost compose_lnmp]# ls
docker-compose.yml  mysql  nginx  php  wwwroot
[root@hostalocal compose_lnmp]# cat docker-compose.yml
version: '3'
services:                          //指定多个服务 nginx、php、mysql
 nginx:
  hostname: nginx
  build:                          //构建容器
   context: ./nginx               //指定镜像产生的工作目录
   dockerfile: Dockerfile
  ports:                          //指定端口
```

```
    - 80:80
  networks:                    //指定网络
    - lnmp
  volumes:                     //数据卷
    - ./wwwroot:/usr/local/nginx/html      //将/wwwroot 与/usr/local/nginx/html 做关联

 php:
  hostname: php
  build:
    context: ./php
    dockerfile: Dockerfile
  ports:
    - 9000:9000
  networks:
    - lnmp
  volumes:
    - ./wwwroot:/usr/local/nginx/html

 mysql:
  hostname: mysql
  image: mysql:5.6                //指定官方仓库里的镜像
  ports:
    - 3306:3306
  networks:
    - lnmp
  volumes:
    - ./mysql/conf:/etc/mysql/conf.d
    - ./mysql/data:/var/lib/mysql
  command: --character-set-server=utf8
  environment:                    //定义环境变量
    MYSQL_ROOT_PASSWORD: 123456
    MYSQL_DATABASE: wordpress
    MYSQL_USER: user
    MYSQL_PASSWORD: user123

networks:
 lnmp:                        //定义网络 lnmp
```

Compose 内的文件都是镜像构建的主工作目录：

```
[root@hostalocal compose_lnmp]# ls
docker-compose.yml  mysql  nginx  php  wwwroot
```

注意：nginx/nginx.conf 需要修改一个 IP 地址

wwwroot 的作用：

```
[root@hostalocal compose_lnmp]# ls wwwroot/
index.html  index.php
用来存放网页的
```

·测试环节

```
[root@hostalocal ~]# cat centos-7-x86_64.tar.gz  | docker import - centos:7  #导入一个系统镜像
sha256:0242445816cd4ca1e4ca3d797c47b4bd95cab1b367da7ebfe512d9b6f542285c
[root@hostalocal ~]# docker images
REPOSITORY       TAG       IMAGE ID       CREATED              SIZE
centos           7         0242445816cd   About a minute ago   589MB
```

启动 compose

```
[root@hostalocal ~]# cd compose_lnmp/
[root@hostalocal compose_lnmp]# docker-compose  up -d  #启动 compose
[root@hostalocal compose_lnmp]# docker-compose  ls  #查看镜像和容器
```

·解压 wordpress-4.9.4-zh_CN.tar.gz

```
[ root@localhost ~]# tar xf wordpress-4.7.4-zh_CN.tar.gz
[ root@localhost ~]#  mv wordpress  compose_lnmp/wwwroot/
```

Web 访问 192.168.200.12/index.html

Web 访问 192.168.200.12

PHP Version 5.6.39

System	Linux php 3.10.0-1062.el7.x86_64 #1 SMP Wed Aug 7 18:08:02 UTC 2019 x86_64
Build Date	Mar 13 2021 06:10:21
Configure Command	'./configure' '--prefix=/usr/local/php' '--with-config-file-path=/usr/local/php/et mysqli' '--with-openssl' '--with-zlib' '--with-curl' '--with-gd' '--with-jpeg-dir' '--v '--enable-fpm' '--enable-zip' '--enable-mbstring'
Server API	FPM/FastCGI
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/usr/local/php/etc
Loaded Configuration File	/usr/local/php/etc/php-fpm.conf
Scan this dir for additional .ini files	(none)
Additional .ini files parsed	(none)
PHP API	20131106
PHP Extension	20131226
Zend Extension	220131226
Zend Extension Build	API220131226,NTS
PHP Extension Build	API20131226,NTS
Debug Build	no
Thread Safety	disabled
Zend Signal Handling	disabled
Zend Memory Manager	enabled
Zend Multibyte Support	provided by mbstring
IPv6 Support	enabled

Web 访问 192.168.200.12/wordpress（可以使用自定义的密码和用户登入）


欢迎使用WordPress。在开始前，我们需要您数据库的一些信息。请准备好如下信息。
1. 数据库名
2. 数据库用户名
3. 数据库密码
4. 数据库主机
5. 数据表前缀（table prefix，特别是当您要在一个数据库中安装多个WordPress时）
我们会使用这些信息来创建一个wp-config.php文件。 如果自动创建未能成功，不用担心，您要做的只是将数据库信息填入配置文件。您也可以在文本编辑器中打开wp-config-sample.php，填入您的信息，并将其另存为wp-config.php。 需要更多帮助？看这里。
绝大多数时候，您的网站服务提供商会给您这些信息。如果您没有这些信息，在继续之前您将需要联系他们。如果您准备好了...
现在就开始！


用户名或电子邮件地址
admin
密码
记住我的登录信息
登录
忘记密码?
显示选项
帮助
WordPress 5.7现已可用！请现在更新。
仪表盘
欢迎使用WordPress!
我们准备了几个链接供您开始:
不再显示
开始使用
自定义您的站点
或更换主题
接下来
撰写您的第一篇博文
添加"关于"页面
查看站点
更多操作
管理边栏小工具和菜单
打开/关闭评论功能
了解更多新手上路知识

PS：如果访问报 Access denined，手动进入 nginx 容器添加读权限。

docker-compose 解决的问题局限在“编排”二字，甚至连“部署”范畴都涉足甚少，而在一个能够服务于大众的云平台中，编排与部署也仅仅是其中的一个组成部分而已。来一起分析一下它的局限制会有哪些：

- docker-compse 是面向单宿主机部署的，这是一种部署能力的欠缺。在更多的场合下，管理员需要面对大量物理服务器（或者虚拟机），这时如果要实现基于 docker-compose 的容器自动化编排与部署，管理员就得借助成熟的自动化运维工具（ansible、puppet、chef、saltstack）来负责管理多个目标主机，将 docker-compose 所需的所有资源（配置文件、用户代码）交给目标主机，然后在目标主机上执行 docker-compose 指令。
- 同样网络和存储也比较棘手，Docker 不能提供跨宿主机的网络，完全面向 Docker daemon 的 docker-compose 当然也不支持。这意味着管理员必须部署一套类似于 OpenvSwich 的独立网络工具，而且管理员还需要完成集成工作。当好不容易把容器编排都安排妥当之后，又会发现容器还处在内网环境中，于是负载均衡、服务发现等一堆问题就面临而来了，这些问题很快能消耗掉工程师所有的耐心。

那么，是否有一种能够提供完善的面向服务器集群的 Docker 编排和部署方案呢？Docker 官方给出的答案是 Compose 同 Machine 和 Swarm 联动，其实还有大家近期经常听到了 kubernetes（k8s）。

那么，是否有一种能够提供完善的面向服务器集群的 Docker 编排和部署方案呢？Docker 官方给出的答案是 Compose 同 Machine 和 Swarm 联动，其实还有大家近期经常听到了 kubernetes（k8s）。