玩转docker-composer基础篇

## 0.docker-composer安装

从github上下载docker-compose二进制文件安装

##下载最新版的docker-compose文件  
curl -L https://github.com/docker/compose/releases/download/1.16.1/docker-compose-`uname -s`-`uname -m` -o /usr/local/bin/docker-compose  
  
##添加可执行权限  
chmod +x /usr/local/bin/docker-compose  
  
##测试安装结果  
docker-compose --version

## 1.基本概念

* 使用Docker Compose不再需要使用shell脚本来启动容器
* Compose 通过一个配置文件来管理多个Docker容器，在配置文件中，所有的容器通过services来定义，然后使用docker-compose脚本来启动，停止和重启应用，和应用中的服务以及所有依赖服务的容器
* Compose 项目由 Python 编写，实现上调用了 Docker 服务提供的 API 来对容器进行管理
* 服务 (service)：一个应用的容器，实际上可以包括若干运行相同镜像的容器实例。
* 项目 (project)：由一组关联的应用容器组成的一个完整业务单元，在 docker-compose.yml 文件中定义。

## 2.基本命令的使用技巧

docker-compose –help 参数 可以查看调用的参数的格式

##容器相关操作的命令  
  
docker-compose --help

Define and run multi-container applications with Docker.  
  
Usage:  
 docker-compose [-f <arg>...] [options] [COMMAND] [ARGS...]  
 docker-compose -h|--help  
  
Options:  
 -f, --file FILE Specify an alternate compose file  
 (default: docker-compose.yml)  
 -p, --project-name NAME Specify an alternate project name  
 (default: directory name)  
 --verbose Show more output  
 --log-level LEVEL Set log level (DEBUG, INFO, WARNING, ERROR, CRITICAL)  
 --no-ansi Do not print ANSI control characters  
 -v, --version Print version and exit  
 -H, --host HOST Daemon socket to connect to  
  
 --tls Use TLS; implied by --tlsverify  
 --tlscacert CA\_PATH Trust certs signed only by this CA  
 --tlscert CLIENT\_CERT\_PATH Path to TLS certificate file  
 --tlskey TLS\_KEY\_PATH Path to TLS key file  
 --tlsverify Use TLS and verify the remote  
 --skip-hostname-check Don't check the daemon's hostname against the  
 name specified in the client certificate  
 --project-directory PATH Specify an alternate working directory  
 (default: the path of the Compose file)  
 --compatibility If set, Compose will attempt to convert deploy  
 keys in v3 files to their non-Swarm equivalent  
  
Commands:  
 build Build or rebuild services  
 bundle Generate a Docker bundle from the Compose file  
 config Validate and view the Compose file  
 create Create services  
 down Stop and remove containers, networks, images, and volumes  
 events Receive real time events from containers  
 exec Execute a command in a running container  
 help Get help on a command  
 images List images  
 kill Kill containers  
 logs View output from containers  
 pause Pause services  
 port Print the public port for a port binding  
 ps List containers  
 pull Pull service images  
 push Push service images  
 restart Restart services  
 rm Remove stopped containers  
 run Run a one-off command  
 scale Set number of containers for a service  
 start Start services  
 stop Stop services  
 top Display the running processes  
 unpause Unpause services  
 up Create and start containers  
 version Show the Docker-Compose version information

## 3. docker-composer 常用的命令

* ps：列出所有运行容器

docker-compose ps

* logs：查看服务日志输出

docker-compose logs eureka

* build：构建或者重新构建服务

docker-compose build .  
  
##或者  
docker-compose build -f xx.yml

* rm：删除指定服务的容器

docker-compose rm eureka

* start：启动指定服务已存在的容器

docker-compose stop eureka

* stop：停止已经处于运行状态的容器，但不删除它

docker-compose stop eureka

* restart:重启项目中的服务

docker-compose restart eureka

* up：构建、启动容器

docker-compose up -d

* down：停用移除所有容器以及网络相关

docker-compose down

## 4.docker-compose.yml 属性

* version：指定 docker-compose.yml 文件的写法格式

version: '3'

* services：多个容器集合

version: '3'  
services:  
 php-fpm:  
 java:  
 go:

* build：配置构建时，Compose 会利用它自动构建镜像，该值可以是一个路径，也可以是一个对象，用于指定 Dockerfile 参数

version: '3'  
services:  
 php-fpm:  
 build:  
 context: ./php-fpm ##目录  
 dockerfile: Dockerfile ##配置文件  
 java:  
 build:  
 context: ./java  
 go:  
 build:  
 context: ./go

* command：覆盖容器启动后默认执行的命令

version: '3'  
services:  
 php-fpm:  
 build:  
 context: ./php-fpm ##目录  
 dockerfile: Dockerfile ##配置文件  
 java:  
 build:  
 context: ./java  
 command: [bundle,exec,thin,-p,3000]  
 go:  
 build:  
 context: ./go

* environment：环境变量配置，可以用数组或字典两种方式

environment:  
 RACK\_ENV: development  
 SHOW: 'ture'  
-------------------------  
environment:  
 - RACK\_ENV=development  
 - SHOW=ture

version: '3'  
services:  
 php-fpm:  
 build:  
 context: ./php-fpm ##目录  
 dockerfile: Dockerfile ##配置文件  
 java:  
 build:  
 context: ./java  
 command: [bundle,exec,thin,-p,3000]  
 go:  
 build:  
 context: ./go  
 environment:  
 - MYSQL\_ROOT\_PASSWORD=${MYSQL\_ROOT\_PASSWORD}  
 - TZ=${WORKSPACE\_TIMEZONE}

* expose：暴露端口，只将端口暴露给连接的服务，而不暴露给主机

version: '3'  
services:  
 php-fpm:  
 build:  
 context: ./php-fpm ##目录  
 dockerfile: Dockerfile ##配置文件  
 expose:  
 - "3000"  
 - "8000"  
 java:  
 build:  
 context: ./java  
 command: [bundle,exec,thin,-p,3000]  
 go:  
 build:  
 context: ./go  
 environment:  
 - MYSQL\_ROOT\_PASSWORD=${MYSQL\_ROOT\_PASSWORD}  
 - TZ=${WORKSPACE\_TIMEZONE}

* image：指定服务所使用的镜像

version: '3'  
services:  
 nginx:  
 image: nginx  
 php-fpm:  
 build:  
 context: ./php-fpm ##目录  
 dockerfile: Dockerfile ##配置文件  
 expose:  
 - "3000"  
 - "8000"  
 java:  
 build:  
 context: ./java  
 command: [bundle,exec,thin,-p,3000]  
 go:  
 build:  
 context: ./go  
 environment:  
 - MYSQL\_ROOT\_PASSWORD=${MYSQL\_ROOT\_PASSWORD}  
 - TZ=${WORKSPACE\_TIMEZONE}

* network\_mode：设置网络模式

network\_mode: "bridge"  
network\_mode: "host"  
network\_mode: "none"  
network\_mode: "service:[service name]"  
network\_mode: "container:[container name/id]"

version: '3'  
networks:  
 os\_bridge:  
 driver: bridge  
services:  
 nginx:  
 image: nginx  
 php-fpm:  
 build:  
 context: ./php-fpm ##目录  
 dockerfile: Dockerfile ##配置文件  
 expose:  
 - "3000"  
 - "8000"  
 networks:  
 - os\_bridge  
  
 java:  
 build:  
 context: ./java  
 command: [bundle,exec,thin,-p,3000]  
 go:  
 build:  
 context: ./go  
 environment:  
 - MYSQL\_ROOT\_PASSWORD=${MYSQL\_ROOT\_PASSWORD}  
 - TZ=${WORKSPACE\_TIMEZONE}

* ports：对外暴露的端口定义，和 expose 对应

version: '3'  
services:  
 php-fpm:  
 build:  
 context: ./php-fpm ##目录  
 dockerfile: Dockerfile ##配置文件  
 expose:  
 - "3000"  
 - "8000"  
 ports:  
 - "59501:9501"  
 java:  
 build:  
 context: ./java  
 command: [bundle,exec,thin,-p,3000]  
 go:  
 build:  
 context: ./go  
 environment:  
 - MYSQL\_ROOT\_PASSWORD=${MYSQL\_ROOT\_PASSWORD}  
 - TZ=${WORKSPACE\_TIMEZONE}

* volumes：卷挂载路径

version: '3'  
services:  
 php-fpm:  
 build:  
 context: ./php-fpm ##目录  
 dockerfile: Dockerfile ##配置文件  
 expose:  
 - "3000"  
 - "8000"  
 ports:  
 - "59501:9501"  
 volumes:  
 - ./php-fpm/php${PHP\_VERSION}.ini:/usr/local/etc/php/php.ini  
 - ./php-fpm/mod:/usr/local/etc/php/conf.d/mod/  
 - ./php-fpm/conf.d:/usr/local/etc/php-fpm.d/cfg.d/  
  
 java:  
 build:  
 context: ./java  
 command: [bundle,exec,thin,-p,3000]  
 go:  
 build:  
 context: ./go  
 environment:  
 - MYSQL\_ROOT\_PASSWORD=${MYSQL\_ROOT\_PASSWORD}  
 - TZ=${WORKSPACE\_TIMEZONE}

* links：将指定容器连接到当前连接，可以设置别名，避免ip方式导致的容器重启动态改变的无法连接情况

version: '3'  
services:  
 php-fpm:  
 build:  
 context: ./php-fpm ##目录  
 dockerfile: Dockerfile ##配置文件  
 expose:  
 - "3000"  
 - "8000"  
 ports:  
 - "59501:9501"  
 volumes:  
 - ./php-fpm/php${PHP\_VERSION}.ini:/usr/local/etc/php/php.ini  
 - ./php-fpm/mod:/usr/local/etc/php/conf.d/mod/  
 - ./php-fpm/conf.d:/usr/local/etc/php-fpm.d/cfg.d/  
 links:  
 - mongo:db ##mongo是其他容器的服务名,php-fpm服务用db访问到mongo的容器的服务  
 - redis  
 java:  
 build:  
 context: ./java  
 command: [bundle,exec,thin,-p,3000]  
 go:  
 build:  
 context: ./go  
 environment:  
 - MYSQL\_ROOT\_PASSWORD=${MYSQL\_ROOT\_PASSWORD}  
 - TZ=${WORKSPACE\_TIMEZONE}

## 5.出现的问题

Q:当docker-compose.yml配置发生变化时，如何去更新容器? A:可使用 docker-compose up 命令更新配置

Q:为什么yml配置总是报错呢？ A:yml配置是有格式的，YAML中允许表示三种格式，分别是常量值，对象和数组。格式如下：

#即表示url属性值；  
url: http://www.wolfcode.cn   
#即表示server.host属性的值；  
server:  
 host: http://www.wolfcode.cn   
#数组，即表示server为[a,b,c]  
server:  
 - 120.168.117.21  
 - 120.168.117.22  
 - 120.168.117.23  
#常量  
pi: 3.14 #定义一个数值3.14  
hasChild: true #定义一个boolean值  
name: '你好YAML' #定义一个字符串

基本格式要求

1，YAML大小写敏感；

2，使用缩进代表层级关系；

3，缩进只能使用空格，不能使用TAB，不要求空格个数，只需要相同层级左对齐（一般2个或4个空格）

## 6.参考文章

[docker官方](https://docs.docker.com/)

[docker-composer官方](https://docs.docker.com/compose/) [docker-composer文件配置参数](https://docs.docker.com/compose/compose-file/)