Debug

Docker Compose—简化复杂应用的利器

2015年5月17日 by debugo · 5条评论

Compose是用于定义和运行复杂Docker应用的工具。你可以在一个文件中定义一个多容器的应用，然后使用一条命令来启动你的应用，然后所有相关的操作都会被自动完成。

1. 安装Docker和Compose

```
# 当前最新的Docker是1.6.2，Compose为1.2.0
curl -s https://get.docker.io/ubuntu/ | sudo sh
sudo apt-get update
sudo apt-get install lxc-docker
# 参考http://docs.docker.com/compose/install/#install-compose
curl -L https://github.com/docker/compose/releases/download/1.2.0/docker-compose-`uname -s`-`
chmod +x /usr/local/bin/docker-compose
### 上面这个方法真的慢出翔，可以通过Python pip安装。
apt-get install python-pip python-dev
pip install -U docker-compose
```

这样compose就安装好了，查看一下compose的版本信息：

```
chmod +x /usr/local/bin/docker-compose
docker-compose -version
docker-compose 1.2.0
```

2. 使用Compose

使用Compose只需要简单的三个步骤：
首先，使用Dockerfile来定义你的应用环境：

```
FROM python:2.7
ADD ./code
WORKDIR /code
RUN pip install -r requirements.txt
```

其中，requirements.txt中的内容包括：

```
flask
redis
```

再用Python写一个简单的app.py

```
from flask importFlaskfrom redis importRedisimport os
app =Flask(__name__)
redis =Redis(host='redis', port=6379)@app.route('/')def hello():
    redis.incr('hits')return'Hello World! I have been seen %s times.'% redis.get('hits')if __
    app.run(host="0.0.0.0", debug=True)
```

第二步，用一个compose.yaml来定义你的应用服务，他们可以把不同的服务生成不同的容器中组成你的应用。

```
web:
  build:.
  command: python app.py
  ports:
         - "5000:5000"
  volumes:
         - .:/code
  links:
         - redis
redis:
  image: redis
```

第三步，执行docker-compose up来启动你的应用，它会根据compose.yaml的设置来pull/run这俩个容器，然后再启动。

```
Creating myapp_redis_1...
Creating myapp_web_1...
Building web...
Step 0 : FROM python:2.7
2.7: Pulling from python
...
Status: Downloaded newer image for python:2.7
 ---> d833e0b23482
Step 1 : ADD . /code
 ---> 1c04b1b15808
Removing intermediate container 9dab91b4410d
Step 2 : WORKDIR /code
 ---> Running in f495a62feac9
 ---> ffea89a7b090
Attaching to myapp_redis_1, myapp_web_1
......
redis_1 | [1] 17 May 10:42:38.147 * The server is now ready to accept connections on port 637
web_1   |  * Running on http://0.0.0.0:5000/ (Press CTRL+C to quit)
web_1   |  * Restarting with stat
```

3. Yaml文件参考

在上面的yaml文件中，我们可以看到compose文件的基本结构。首先是定义一个服务名，下面是yaml服务中的一些选项条目：
image:镜像的ID
build:直接从pwd的Dockerfile来build，而非通过image选项来pull
links：连接到那些容器。每个占一行，格式为SERVICE[:ALIAS],例如 – db[:database]
external_links：连接到该compose.yaml文件之外的容器中，比如是提供共享或者通用服务的容器服务。格式同links
command：替换默认的command命令
ports:导出端口。格式可以是：

```
ports:-"3000"-"8000:8000"-"127.0.0.1:8001:8001"
```

expose：导出端口，但不映射到宿主机的端口上。它仅对links的容器开放。格式直接指定端口号即可。
volumes：加载路径作为卷，可以指定只读模式：

```
volumes:-/var/lib/mysql
 - cache/:/tmp/cache
```

```
3  -~/configs:/etc/configs/:ro
```

volumes_from：加载其他容器或者服务的所有卷

```
1 environment:- RACK_ENV=development
2   - SESSION_SECRET
```

env_file：从一个文件中导入环境变量，文件的格式为RACK_ENV=development
extends:扩展另一个服务，可以覆盖其中的一些选项。一个sample如下：

```
1  common.yml
2  webapp:
3    build:./webapp
4    environment:- DEBUG=false- SEND_EMAILS=false
5  development.yml
6  web:extends:
7      file: common.yml
8      service: webapp
9    ports:-"8000:8000"
10   links:- db
11   environment:- DEBUG=true
12 db:
13   image: postgres
```

net：容器的网络模式，可以为”bridge”, “none”, “container:[name or id]”, “host”中的一个。
dns：可以设置一个或多个自定义的DNS地址。
dns_search:可以设置一个或多个DNS的扫描域。
其他的working_dir, entrypoint, user, hostname, domainname, mem_limit, privileged, restart, stdin_open, tty, cpu_shares，和docker run命令是一样的，这些命令都是单行的命令。例如：

```
1  cpu_shares:73
2  working_dir:/code
3  entrypoint: /code/entrypoint.sh
4  user: postgresql
5  hostname: foo
6  domainname: foo.com
7  mem_limit:1000000000
8  privileged:true
9  restart: always
10 stdin_open:true
11 tty:true
```

4. docker-compose常用命令

在第二节中的docker-compose up，这两个容器都是在前台运行的。我们可以指定-d命令以daemon的方式启动容器。除此之外，docker-compose还支持下面参数：
--verbose：输出详细信息
-f 制定一个非docker-compose.yml命名的yaml文件
-p 设置一个项目名称（默认是directory名）
docker-compose的动作包括：
build：构建服务
kill -s SIGINT：给服务发送特定的信号。
logs：输出日志
port：输出绑定的端口
ps：输出运行的容器

pull：pull服务的image
rm：删除停止的容器
run: 运行某个服务，例如docker-compose run web python manage.py shell
start：运行某个服务中存在的容器。
stop:停止某个服务中存在的容器。
up：create + run + attach容器到服务。
scale：设置服务运行的容器数量。例如：docker-compose scale web=2 worker=3
参考:

Compose Document

Posted in Ops, Virtualization.

← 在Windows下构建Docker开发环境
MongoDB的权限管理 →

5条评论 最新 最早 最热

z-kidy

volumes:
-.:/code

死活不成功是什么问题

3月26日 回复 顶 转发

db浆糊

volumes:
- .:/code
-后有个空格

5月24日 回复 顶 转发

compose

links 不起作用,不写hosts,是为什么呢

8月15日 回复 顶 转发

YMwGH

万 部 A 片高清 国产 日韩 hTTp://uVU.cc/inRx

10月4日 回复 顶 转发

jqYLi

这个更刺j激，准备c好手纸哦 A 片。。 hTTp://T.CN/RcDykDn

10月29日 回复 顶 转发

社交帐号登录: 微信 微博 QQ 人人 更多»