Docker 内部培训

涂飞平

2014-05-06

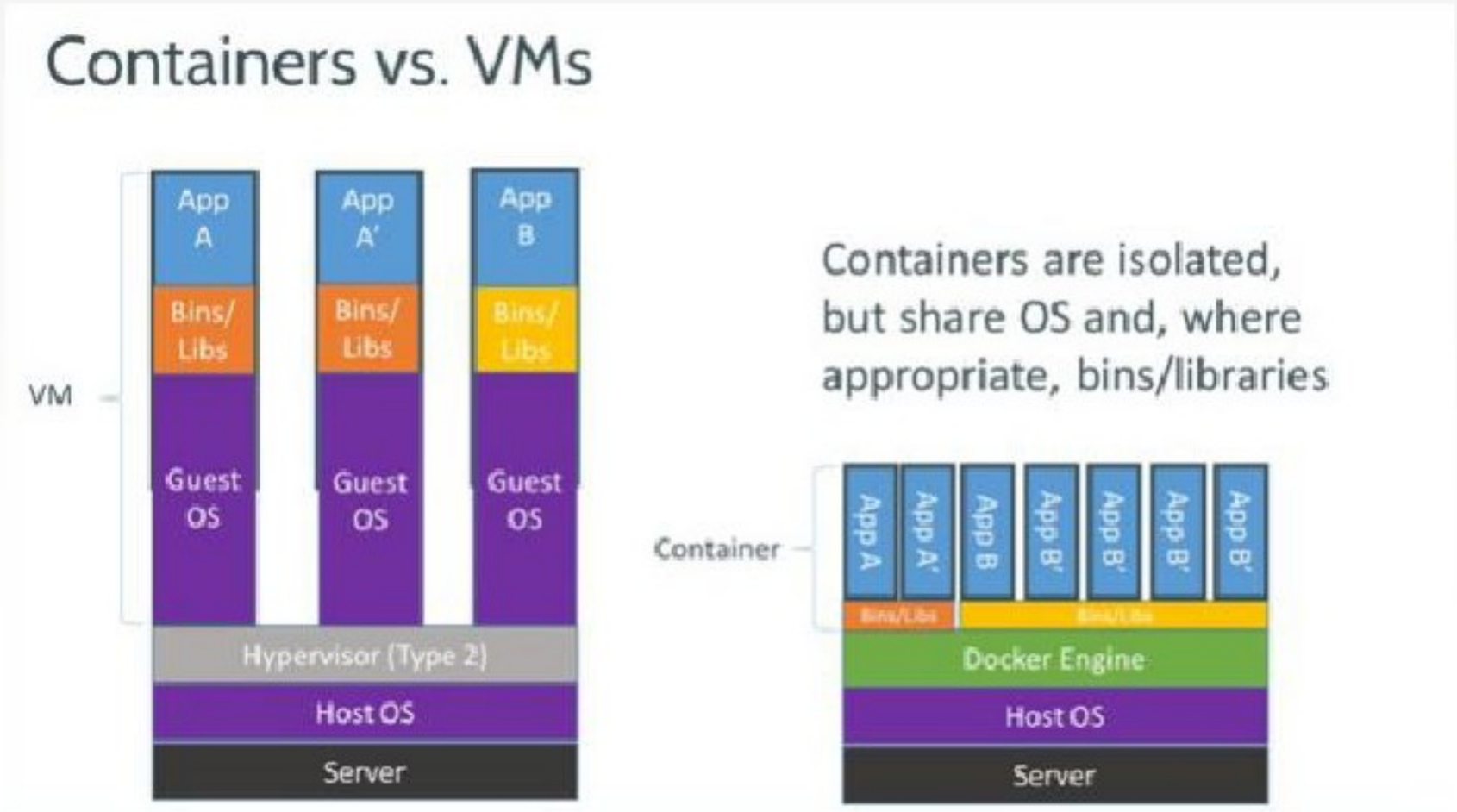
一、什么是Docker？

- Docker 是一个开源的应用容器引擎，让开发者可以打包他们的应用以及依赖包到一个可移植的容器中，然后发布到任何流行的 Linux 机器上，也可以实现虚拟化。容器是完全使用沙箱机制，相互之间不会有任何接口（类似 iPhone 的 app）。几乎没有性能开销,可以很容易地在机器和数据中心中运行。最重要的是,他们不依赖于任何语言、框架或包装系统。（摘自百度）
- Docker 是 PaaS 提供商 dotCloud 开源的一个基于 LXC 的高级容器引擎，源代码托管在 Github 上, 基于go语言
- 更多信息请参考 http://baike.baidu.com/view/11854949.htm?fr=aladdin

二、为什么要用Docker？

2.1、性能上比VM好

一台牛X的服务器，VMWare中启动10个虚拟机，基本上就不能再做其他事情了，大部分CPU和内存资源都耗费在VM内的OS了。而Docker将APP直接放在OS上面，省了大量内存和CPU资源占用，但在效果（沙箱）上与虚拟机完全一致。



2.2、重复使用、隔离应用

快速生成相同的环境，这个在系统测试和开发中的时候经常使用。

两个在公司实际使用的场景

1、根据特定的开发需求，生成一个标准的开发和测试环境，比如机构库的开发，我们需要的环境包括：JDK + Postgresql + tomcat + Nginx + ...

当我们新来一个开发或者测试人员，我们根据对应的环境，生成一个属于这个开发/测试人员的虚拟机（暂且如此称呼吧），编译部署代码，测试系统都隔离在自身负责的范围。

2、对于系统平台部分，数据库和平台部分，安装在一个环境中，然后我们开发的不同模块都连接到这个虚拟机中公开的接口上，这样，这个环境对我们开发产生积极的影响，核心部分隔离了，应用开发不用再繁琐的部署平台的其他部分了，及保证了核心代码安全，又可以让开发人员专注自身业务而不用管平台底层部分。

三、安装Docker

- 这里仅仅针对Ubuntu做说明，其他的Linux都差不多

  aptitude install lxc-docker就可以安装docker了，从名字可以看出docker与lxc的关系，至于lxc是什么，可以自行Google，lxc是docker的底层实现，是一种linux的内核虚拟化的工具集合

- 安装完毕后，使用命令 docker --version查看版本并确定是否安装成功

```
root@230Server:~# docker --version
Docker version 0.10.0, build dc9c28f
```

四、Docker的基本元素

- Image

提供一个快速部署的模板，这是与lxc最大的区别，可以基于一个Image快速部署多个相同的容器，docker images命令可以查看目前系统有哪些镜像

```
root@230Server:~# docker images
REPOSITORY                    TAG                 IMAGE ID            CREATED             VIRTUAL SIZE
memcached                     latest              c9a8f438ef4d        13 days ago         842 MB
irp-web-ui                    latest              f8c769bc6c4d        2 weeks ago         856.8 MB
msgserver                     latest              2b1bcdbc7092        3 weeks ago         837.5 MB
```

- Container

容器，可以简单认为是虚拟机了（虽然不准确），它是我们工作实体，每个Container相当于一个完整的Linux系统，docker ps命令可以查看目前系统有哪些容器

五、如何创建/获得Docker Image

- 5.1、通过网络获取
  可以到http://index.docker.io网站查找是否有自己需要的 docker image，如果有合适的，直接使用 docker pull命令就可以获取到。
- 5.2、自己构建

如果没有合适的，这个时候我们可以通过编写Dockerfile文件，然后通过 docker build命令来创建自己image。

```
#IRP develope base image
FROM irp5-base:latest
MAINTAINER Sunny <tufeiping@gmail.com>

#install vim
RUN apt-get install -y vim

USER root

EXPOSE 8080
EXPOSE 5432
EXPOSE 22

ENTRYPOINT /usr/local/run.py
```

六、Dockerfile简介

- 详细请参考
  http://docs.docker.io/reference/builder/
- 基本指令（比较重要的指令）

  FROM RUN ADD CMD/ENTRYPOINT USER EXPOSE
- 样例
- 生成Image
  docker build -t msgserver .

```
#IRP develope base image
FROM irp5-base:latest
MAINTAINER Sunny <tufeiping@gmail.com>
#install vim
RUN apt-get install -y vim
USER root


EXPOSE 8080
EXPOSE 5432
EXPOSE 22


ENTRYPOINT /usr/local/run.py
```

七、生成，使用Container

- 如果把Image比作Java中的类（Class），那Container就是Java中的对象（Object）了，真正做实事的就是Container
  在Docker中，只要使用docker run运行一个image，就会生成一个container（没错，每次运行都生成一个新的container），所以一般都是运行一次后，记录ContainerID，然后在需要开始或者停止的时候使用docker start/stop命令来启动或者停止
- 示例
  docker run -d -p 11211:11211 -p 23044:22 memcached

  docker ps可以查看已经启动的container信息

八、现状和建议

- 公司内部的服务都已经部署好的，都已经写好脚本了，所有的container都有静态映射ssh端口，可以进入后进行细微的配置调整（docker在公司内部目前主要作为paas平台在应用）

- 如果构建新的image和container，需要系统地学习docker的使用

- 建议学习docker前，先学习lxc的使用和概念，有时间，可以参考docker的源代码，能更深入了解docker是怎样整合其他工具和技术的

```
root@230Server:~# docker images
REPOSITORY                    TAG                 IMAGE ID            CREATED             VIRTUAL SIZE
memcached                     latest              c9a8f438ef4d        13 days ago         842 MB
irp-web-ui                    latest              f8c769bc6c4d        2 weeks ago         856.8 MB
msgserver                     latest              2b1bcdbc7092        3 weeks ago         837.5 MB
ubuntu                        saucy               25593492b938        3 weeks ago         192.7 MB
caas                          latest              b37331ebe958        3 weeks ago         1.263 GB
mlpla                         latest              9f02938c327b        4 weeks ago         1.685 GB
standard                      vm                  28935279b42f        4 weeks ago         990.5 MB
lfx                           latest              28935279b42f        4 weeks ago         990.5 MB
irp5-base-dspace              latest              7dfda351211c        4 weeks ago         1.147 GB
nexus                         latest              3e6145cb4166        4 weeks ago         4.06 GB
dspace-platform               latest              34f394fd4a42        4 weeks ago         1.087 GB
irp5-base                     latest              6ebed094f231        4 weeks ago         990.5 MB
<none>                        <none>              181a1b708c96        4 weeks ago         811.3 MB
irpbase                       latest              40a6e3746f42        4 weeks ago         808 MB
<none>                        <none>              f721eafaa7fb        4 weeks ago         794.8 MB
<none>                        <none>              ec981800d2b2        4 weeks ago         794.8 MB
<none>                        <none>              1ec84bd98ebb        4 weeks ago         794.8 MB
<none>                        <none>              3cc67c9e6527        4 weeks ago         794.8 MB
<none>                        <none>              f37dfe6c1ccc        4 weeks ago         239.2 MB
<none>                        <none>              35d4418a4945        4 weeks ago         239.2 MB
<none>                        <none>              12c283d09e0a        4 weeks ago         207.7 MB
besn0847/hdfs                 latest              4ffac023a148        5 weeks ago         789.9 MB
besn0847/hdfs                 base                4faa80d67388        5 weeks ago         789.9 MB
sequenceiq/hadoop-docker      latest              dc8d5b074208        5 weeks ago         839.3 MB
sequenceiq/hadoop-docker      1.0                 76b806d47413        5 weeks ago         839.2 MB
ubuntu                        precise             9cd978db300e        3 months ago        204.4 MB
192.168.0.230:8081/ubuntu     latest              8dbd9e392a96        12 months ago       128 MB
localhost:49153/ubuntu        latest              8dbd9e392a96        12 months ago       128 MB
```

```
root@230Server:~# docker ps
CONTAINER ID        IMAGE                              COMMAND                CREATED             STATUS              PORTS

7bdc1a301cce        sequenceiq/hadoop-docker:latest    /etc/bootstrap.sh -d   22 hours ago        Up 22 hours         0.0.0.0:8130->8030/tcp, 0.0.0.0:8131->8031/tcp, 0.0.0.0:81
0.0.0.0:8142->8042/tcp, 0.0.0.0:8188->8088/tcp, 0.0.0.0:23050->22/tcp, 0.0.0.0:49707->49707/tcp, 0.0.0.0:50010->50010/tcp, 0.0.0.0:50020->50020/tcp, 0.0.0.0:50070->50070/tcp,
archimedes
6dc8262f71e1        memcached:latest                   /bin/sh -c /usr/loca   13 days ago         Up 16 hours         5432/tcp, 8080/tcp, 0.0.0.0:11211->11211/tcp, 0.0.0.0:2304

vinci
ada4bd333d14        irp-web-ui:latest                  /bin/sh -c /usr/loca   2 weeks ago         Up 22 hours         5432/tcp, 0.0.0.0:8043->8080/tcp, 0.0.0.0:23043->22/tcp

re
b7c397f749b3        msgserver:latest                   /bin/sh -c /usr/loca   3 weeks ago         Up 22 hours         5432/tcp, 0.0.0.0:5555->5555/tcp, 0.0.0.0:5556->5556/tcp,

ttani
dd50e891d9d3        caas:latest                        "/bin/sh -c /usr/loc   3 weeks ago         Up 22 hours         5432/tcp, 0.0.0.0:8038->8080/tcp, 0.0.0.0:23038->22/tcp

ewton
dbccd5e02fcf        mlpla:latest                       /bin/sh -c '/opt/tom   4 weeks ago         Up 22 hours         5432/tcp, 0.0.0.0:8037->8080/tcp, 0.0.0.0:23037->22/tcp

bardeen
25884164e20e        lfx:latest                         /bin/sh -c /usr/loca   4 weeks ago         Up 22 hours         5432/tcp, 0.0.0.0:8041->8080/tcp, 0.0.0.0:23041->22/tcp

mpson
db6af81a5cea        nexus:latest                       /bin/sh -c /run.py /   4 weeks ago         Up 22 hours         5432/tcp, 0.0.0.0:6332->8080/tcp, 0.0.0.0:23022->22/tcp

d_engelbart
0315a9257435        dspace-platform:latest             /bin/sh -c /usr/loca   4 weeks ago         Up 22 hours         5432/tcp, 0.0.0.0:8040->8080/tcp, 0.0.0.0:23040->22/tcp
```

谢谢大家