docker介绍

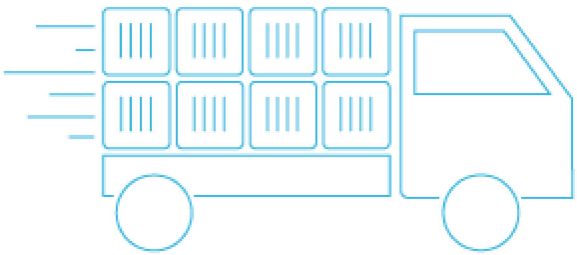
> docker是用google开源语言golang开发的一款类似vm这样的管理容器的开源软件，他的优势在于不像vm一样启动一个虚拟机需要等待漫长的时间，而对docker而言只需要简单的启动一个image就相当于运行一个独立的container。各个container是相互独立的。启停一个container也是相对方便和快速的

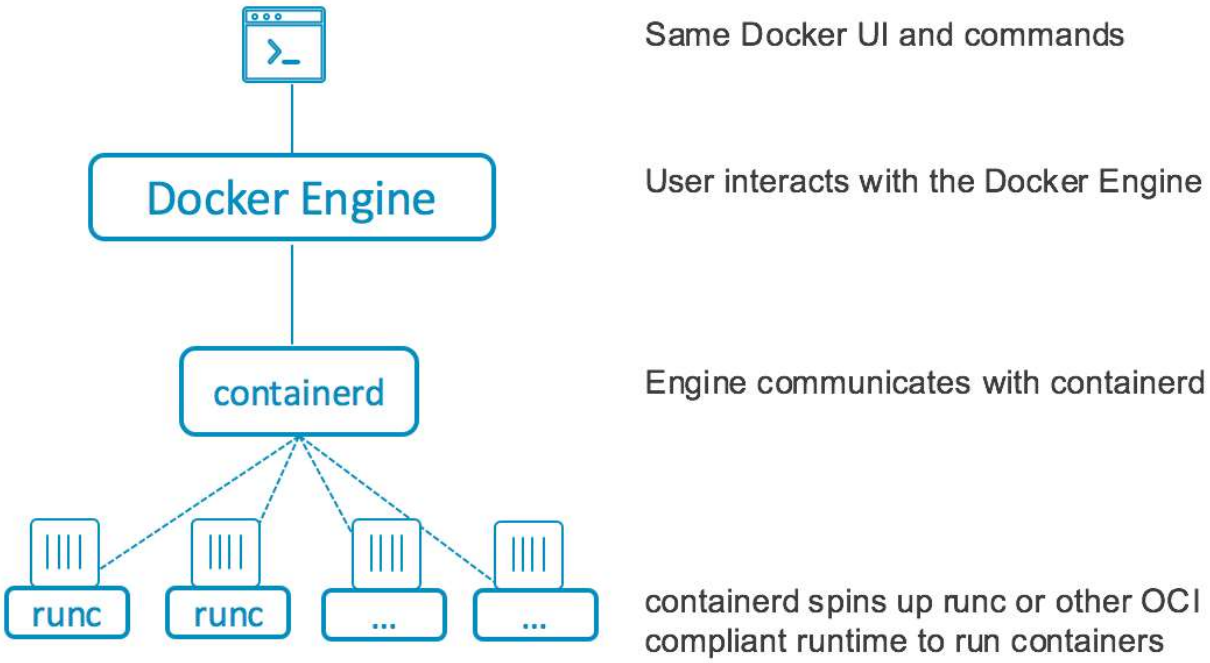
此笔记记录阅读*docker*官网及博客并实践操作所记

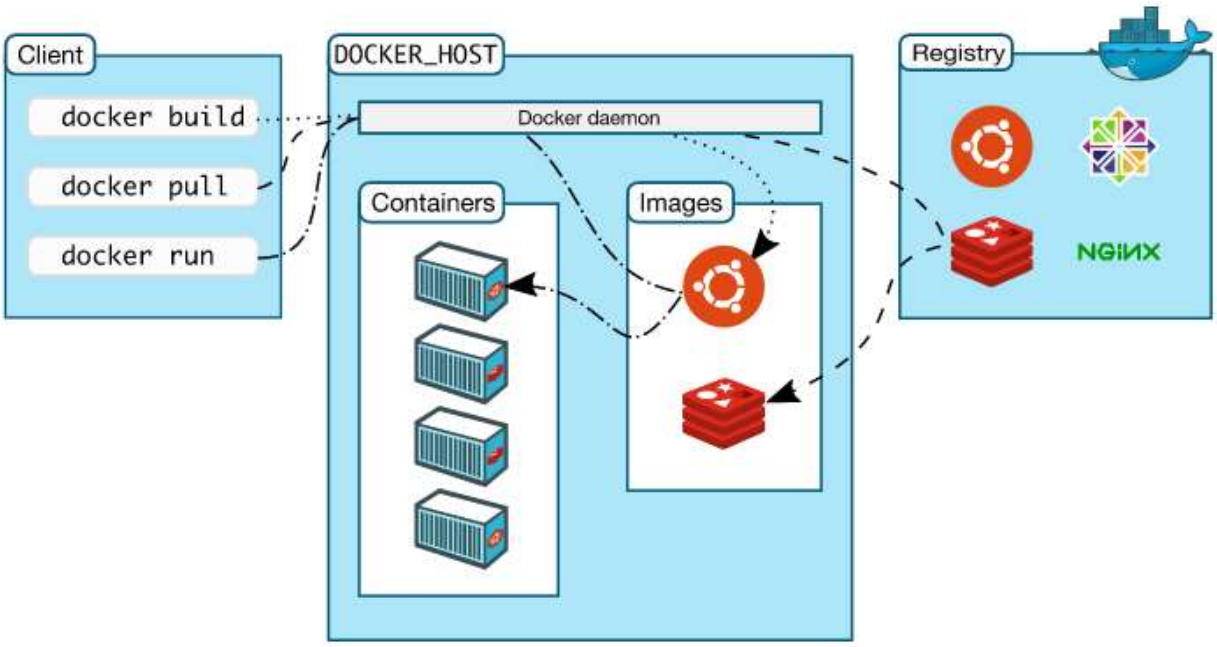
- docker是个什么东西
- docker和vm的区别
- docker实践讲解

1. 什么是docker

> dokcer的英文是叫码头工人，这个很有趣，然而它所做的一切也正像码头工人所干的事一样。docker是一个软件集合，帮你做了码头的活，所有的东西都用我的容器来打包，你可以在你的容器里面运行你自己的东西。

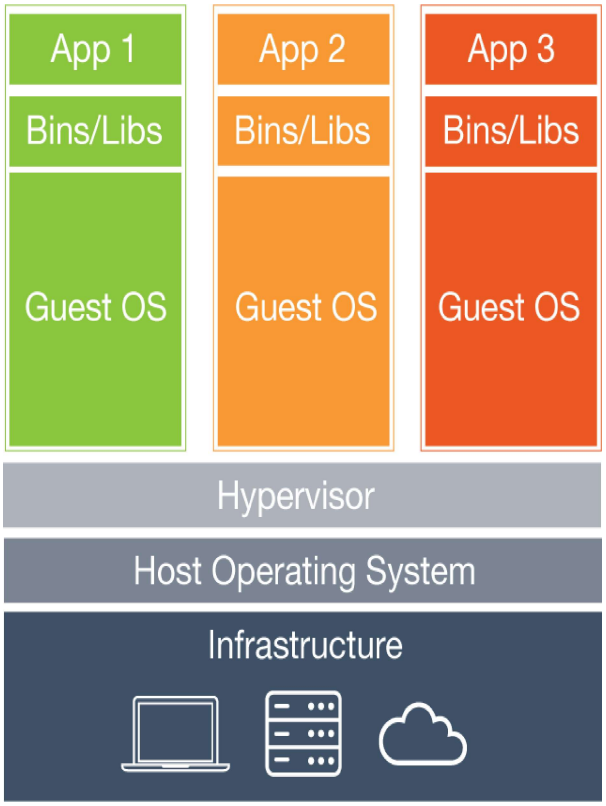


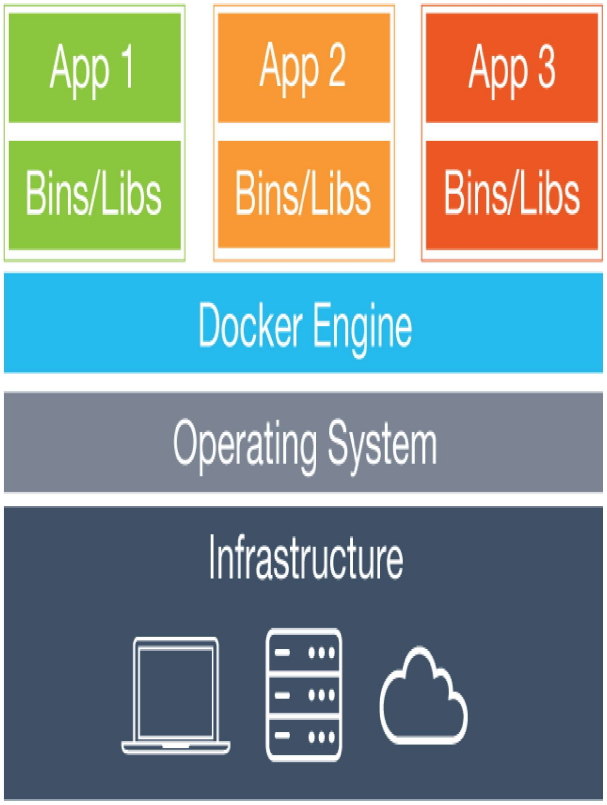




2. 和通常我们说的vm的区别

docker的容器就类似我们vm的安装的iso镜像，只是vm和docker已经把环境的各种依赖的工作都准备好了。docker非常方便的廉价的运行多个container。 container实际是是独立于操作系统上的一个独立的进程，这个进程实现了资源和环境的隔离。它拥有自己的ipc、network、user及pid等





3. docker实践

- 安装环境
- docker组成部分的一些概念
- 网络
- 构建属于自己image的
- docker生态

3.1 环境准备

- 下载安装centos7.2镜像虚拟机
- 安装docker (https://docs.docker.com/engine/installation/linux/centos/)
- 启动docker `systemctl start docker`

> docker版本现在到了1.11了，需要linux 3.10的内核，所以下载的centos7来操作

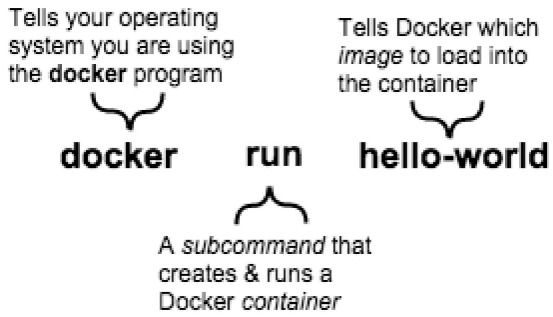
服务启动脚本在 `/usr/lib/systemd/system` 目录下

3.2 Concept

- Understand images & container (https://docs.docker.com/linux/step_two/)
- Docker Volume (http://cloud.51cto.com/art/201501/463143.htm)

image和container区别是image就相当于一个编译好的软件，container就好比这个软件的实例，实例可以有多个。image是静态的，container是一个image的运行态。

volume是一个container的独立于UFS系统的一个存储单元，通常用于挂载宿主的一个目录



3.3 Network

- container网络配置 (https://docs.docker.com/engine/userguide/networking/dockernetworks/)
- container独立ip配置 (http://www.cnblogs.com/feisky/p/4063162.html)

docker启动的容器的网络默认是桥接虚拟的docker0网卡，它桥接到主机，与主机共享网络，而且默认的ip地址段是虚拟的docker0网卡得出来以172.17.0开头的。虽然容器可以通过桥接访问到外部主机，但是外部主机想要访问我docker里面的容器是不可到达的，只能访问到docker的宿主机,这样在多个容器需要与其它机器通信的时候，我需要个容器的ip地址不同，端口相同。后台到go群里头问了下相关解决方案，之前看了下snat和dnat原理，以及听建议宿主机配多个ip,然后用dnat映射到容器里的ip，后来建议容器启动的时候用–net=host，程序listen的时候配置宿主ip即可。好了，赶紧动手试一下^_^

实践操作了一天，最终此方案还是不行，还是要想方法通过-p来映射宿主机的不同的ip地址段到不同的容器，可以设置多个。

步骤:

1. 在宿主机上创建多个需要的ip地址
2. 创建的容器以-p ip:port:port方式运行，ip是宿主机上的ip地址

3.3.1 主机配置多个ip

```
cd /etc/sysconfig/network-script
cp ifcfg-eno ifcfg-eno:0
```

example:

```
DEVICE="eno16777736:0"
BOOTPROTO=static
ONBOOT=yes
TYPE="Ethernet"
IPADDR=10.10.2.70
NETMASK=255.255.255.0
GATEWAY=10.10.2.1
```

说明

DEVICE字段的名称需要和外面网卡的文件名称一致(去掉ifcfg-),BOOTPROTO改成静态的，IPADDR改成需要的地址即可，NETMASK是子网掩码，GATEWAY是网关地址。实践操作的系统是centos7以上的，里面看到的内容可能不同，直接用这个替换掉就可

3.3.2 启动一个容器

```
docker run -it --name test1 imageid
docker run -it -p 10.10.2.62:6379:6379 -v /home/yibin/DockerData/qq:/data --name conta
docker run -it -p 10.10.2.70:6379:6379 -v /home/yibin/DockerData/wx:/data --name conta
```

–net参数是容器指定网络连接方式，默认的话就是桥接

安装net-tools工具
yum install net-tools -y

3.4 搭建游戏服务器的基础image

1. 新建一个容器
2. 安装gcc g++ vim net-tools gcc-c++等工具
3. 源码安装redis及yum安装mysql
4. build一个新的image

3.4.1 centos7下面安装yum 安装mysql

1. 只需在/etc/yum.repos.d/目录下添加以下文件mysql-community.repo文件，内容如下:

- 5.6对应的配置

```
# Enable to use MySQL 5.6
[mysql56-community]
name=MySQL 5.6 Community Server
baseurl=http://repo.mysql.com/yum/mysql-5.6-community/el/5/$basearch/
enabled=1
gpgcheck=1
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-mysql
```

- 5.7对应的配置

```
# Note: MySQL 5.7 is currently in development. For use at your own risk.
# Please read with sub pages: https://dev.mysql.com/doc/relnotes/mysql/5.7/en/
[mysql57-community-dmr]
name=MySQL 5.7 Community Server Development Milestone Release
baseurl=http://repo.mysql.com/yum/mysql-5.7-community/el/6/$basearch/
enabled=1
gpgcheck=1
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-mysql
```

2. 最后执行`yum install mysql-community-server`即可

3.4.2 容器里面不能用**systemctl**

因为systemctl是系统启动的时候的一个进程，容器启动的时候并没有启动它，所以不能用它来启动.

docker run -itd –privileged=true -p 10.10.2.62:3306:3306 -p 10.10.2.62:6379:6379 –name test1.com

refer to (https://forums.docker.com/t/systemctl-status-is-not-working-in-my-docker-container/9075)

3.4.3 通过**Dockerfile**新建一个**image**

```
FROM fgame:base

MAINTAINER tofindme 849397833@qq.com

ENTRYPOINT ["/usr/sbin/init"]

expose 3306 6379
```

指定初始进程为init

```
FROM fgame:base
MAINTAINER tofindme 849397833@qq.com
ENV container docker
RUN (cd /lib/systemd/system/sysinit.target.wants/; for i in *; do [ $i == systemd-tmpf
rm -f /lib/systemd/system/multi-user.target.wants/*;\
rm -f /etc/systemd/system/*.wants/*;\
rm -f /lib/systemd/system/local-fs.target.wants/*; \
rm -f /lib/systemd/system/sockets.target.wants/*udev*; \
rm -f /lib/systemd/system/sockets.target.wants/*initctl*; \
rm -f /lib/systemd/system/basic.target.wants/*;\
rm -f /lib/systemd/system/anaconda.target.wants/*;
VOLUME [ "/sys/fs/cgroup" ]
CMD ["/usr/sbin/init"]
expose 3306 6379
```

```
docker build -t fgame:v1.1 .

docker run -itd --privileged=true -p 10.10.2.62:6379:6379 -p 10.10.2.62:3306:3306 -p 1
```

需要**expose**的端口
9000 登录服
9004 逻辑服
3306 数据库
6379 redis
7602 网页

在listen的时候本地端口就行，以及游戏逻辑服务下发给ip地址

3.5 docker生态

围绕docker有着自己的生态，从单机docker到集群管理的docker三剑客(docker swarm compose)，以及docker官网有docker cloud、docker Hub等

docker隔离原理说明 (http://www.tuicool.com/articles/jeEZ7rV)

参考:
- 网卡配置配置ip (http://www.2cto.com/os/201306/223532.html)
- snat和dnat说明 (http://www.cnblogs.com/iceocean/articles/1616305.html)
- 进入容器说明 (http://blog.csdn.net/u010397369/article/details/41045251)
- 静态ip (http://xiaorui.cc/2015/05/19/%E8%A7%A3%E5%86%B3docker%E7%BB%91%E5%AE%9A%E5%88%86%E9%85%8D%E9%9D%99%E6%80%81%E5%A4%96%E7%BD%91ip%E7%9A%84%E9%97%AE%E9%A2%98/)

3.4 路由规则

PREROUTE&POSTROUTE 说明 (http://gaodi2002.blog.163.com/blog/static/232076820070211513268 3/)
POSTROUTE 是指源地址转换，PREROUTE是指目标地址转换
这样就好理解了，我局域网需要访问外部主机，则需要把我局域网内的主机转换成可以与别人通信的地址。若局域网外的主机回包给局域网内的主机，则需要把目标地址转换一下，知道局域网外的主机发给是的局域网内哪台主机