1、docker安装  
  
debian7安装docker  
  
参考地址：http://www.webmaster.me/server/installing-docker-on-debian-  
  
wheezy-in-60-seconds.html  
  
    echo deb http://get.docker.io/ubuntu docker main | sudo   
  
tee/etc/apt/sources.list.d/docker.list    
    sudo apt-key adv --keyserver keyserver.ubuntu.com --recv-keys   
  
36A1D7869245C8950F966E92D8576A8BA88D21E9    
    sudo apt-get update    
    sudo apt-get install -y lxc-docker   
  
#四行命令，Docker就安装好了。下面创建一个ubuntu虚拟系统：  
  
    docker pull ubuntu #此处是从官网拉取名为ubuntu的image，也可手动在  
  
https://index.docker.io上搜索想要的镜像。    
    docker run -i -t ubuntu /bin/bash #创建一个容器，-t是临时终端。   
  
ubuntu12.04、windows、macOS安装docker  
  
参考docker中文文档http://www.widuu.com/docker/  
  
2、docker使用过程实践  
  
2.1 在测试机启动容器，安装ssh  
  
    docker run -i -t ubuntu /bin/bash #此方式运行的容器，退出后容器就会  
  
关闭。    
    apt-get install openssh-server #安装ssh    
    #需要修改/etc/sshd/sshd\_config文件中内容    
    PermitRootLogin yes    
    UsePAM no   
  
2.2 启动ssh，容器以后台方式运行  
  
    docker run -d -p 50001:22 <容器id> /usr/sbin/sshd-D    
    #容器id可通过 docker ps-a查看，最上面的为最新的。   
  
2.3 通过ssh连接到容器安装软件  
  
    ssh root@127.0.0.1-p 50001    
    #连上后想装什么就装什么，可使用exit退出容器，但后台还会运行。   
  
2.4 服务安装完成后，停止容器。  
  
    docker stop <容器id> #停止运行的容器   
  
2.5 把容器提交生成最新的镜像  
  
    docker commit <容器id> debian02 #把这个容器提交生成新的debian02镜像(  
  
该镜像是原始镜像与容器的整合)   
  
2.6 打包镜像  
  
    docker save debian02 >/root/debian02.tar #debian02镜像打包   
  
2.7 在另外的机器上导入镜像  
  
    docker load < debian02.tar #导入镜像    
    docker images #查看存在的镜像   
  
2.8 启动容器  
  
    docker run -h="redis-test" --name redis-test -d -p 51000:22 -  
  
p51001:3306 -p 51003:6379 -p 51004:6381 -p 51005:80 -p 51006:8000 -p   
  
51007:8888 debian02 /etc/rc.local    
    #此处是我测试机器启动命令，指定主机名与端口映射。    
    #启动后，后面又装了程序，开机自启动命令可放在/etc/rc.local文件中。    
    docker容器迁移简单方便，可以任意的拷贝部署，以后再也不怕新部署环境了  
  
，一堆依赖装的想死有木有。   
  
3、关于docker容器的端口映射  
  
由于docker容器的IP地址每次启动都会变，所以不适用于手动添加端口映射(难道  
  
每次重启都来查看容器的IP么？)，所以需要每次启动容器时由docker程序自动添  
  
加NAT规则，前期尽可能的把需要映射的端口在创建容器时配置好，如下：  
  
    docker run -h="activemq" --name activemq -d -p 51000:22 -p   
  
51001:3306-p 51003:6379 -p 51004:6381 -p 51005:80-p 51006:8000 -p   
  
51007:8888 debian/base/etc/rc.local    
    #此处我把mysql,redis,nginx,ssh都进行了映射。   
  
后续对于docker容器的管理，记住容器的名称，如上述名称是activemq，则使用  
  
docker stop,start来控制容器进程。  
  
    docker stop activemq    
    docker start activemq   
  
当然，也可以不让docker每次启动容器修改容器的IP地址，参考如下：  
  
docker网络配置：http://www.open-open.com/lib/view/open1404896485747.html  
  
4、关于docker容器的多程序开机自动运行  
  
docker容器每次启动时，开机自启动的命令都要在启动容器前指定。如 docker   
  
run -I -t debian /bin/bash命令，只会运行/bin/bash程序，其它的程序都不会  
  
运行，对于要跑多个程序的容器特别纠结。  
  
多程序开机自动运行方法：  
  
可把前面所说的启动命令换成dockerrun -I -t debian /etc/rc.local，在容器中  
  
把所有需要开机自的启动命令放在/etc/rc.local中，就可以达到多程序开机自启  
  
动了。  
  
后台运行则是：docker run -d -p 50001:22 debian /etc/rc.local。注意：run  
  
命令是创建一个新的容器，如果要启动一个曾经运行过的容器，则用命令docker   
  
ps -a中找对应的容器ID，然后使用docker start <容器ID>即可。  
  
5、关于docker容器和镜像的关系  
  
无论容器里做什么操作，写文件，删文件。该容器的基本镜像都不会有任何改变。  
  
这是因为Docker从父镜像建立增量镜像，只存储每个容器的更改。因此，如果你有  
  
一个300MB的父镜像，如果你在容器中安装了50MB的额外应用或服务，你的容器只  
  
有50MB，父镜像还是300MB。  
  
但是可以使用Dockfile或commit命令来，把增量镜像和父镜像一起生成一个新的镜  
  
像。  
  
commit使用：  
  
    docker commit <容器id> <新镜像名称>   
  
Dockfile使用：  
  
    root@yangrong:/data# cat Dockerfile    
    FROMubuntu/testa #这是基础镜像    
    CMD["/root/start.sh"] #这是启动命令    
    root@yangrong:/data# docker build -t <新镜像名> ./   
  
关于Dockfile更多参数参考地址：  
  
http://www.tuicool.com/articles/FRvAbe  
  
http://www.colorscode.net/2014/01/04/howto-build-image-with-automatic-  
  
startup-ssh-service-from-dockerfile/  
  
6、docker参数详解  
  
    docker    
    useage of docker    
    -D 默认false 允许调试模式(debugmode)    
    -H 默认是unix:///var/run/docker.sock tcp://[host[:port]]来绑定 或者  
  
unix://[/path/to/socket]来使用(二进制文件的时候)，当主机ip host=  
  
[0.0.0.0],(端口)port=[4243] 或者 path=[/var/run/docker.sock]是缺省值，做  
  
为默认值来使用    
    -api-enable-cors 默认flase 允许CORS header远程api    
    -b 默认是空，附加在已存在的网桥上，如果是用'none'参数，就禁用了容器  
  
的网络    
    -bip 默认是空，使用提供的CIDR（ClasslessInter-Domain Routing-无类型  
  
域间选路）标记地址动态创建网桥(dcoker0),和-b参数冲突    
    -d 默认false 允许进程模式(daemonmode)    
    -dns 默认是空，使docker使用指定的DNS服务器    
    -g 默认是"/var/lib/docker":作为docker使用的根路径    
    -icc 默认true，允许inter-container来通信    
    -ip 默认"0.0.0.0"：绑定容器端口的默认Ip地址    
    -iptables 默认true 禁用docker添加iptables规则    
    -mtu 默认1500 : 设置容器网络传输的最大单元(mtu)    
    -p 默认是/var/run/docker.pid进程pid使用的文件路径    
    -r 默认是true 重启之前运行的容器    
    -s 默认是空 ，这个是docker运行是使用一个指定的存储驱动器    
    -v 默认false 打印版本信息和退出   
  
7、docker run命令详解  
  
    Usage: docker run [OPTIONS] IMAGE[:TAG] [COMMAND] [ARG...]    
    Run a command in a new container    
    -a=map[]: 附加标准输入、输出或者错误输出    
    -c=0: 共享CPU格式（相对重要）    
    -cidfile="": 将容器的ID标识写入文件    
    -d=false: 分离模式，在后台运行容器，并且打印出容器ID    
    -e=[]:设置环境变量    
    -h="": 容器的主机名称    
    -i=false: 保持输入流开放即使没有附加输入流    
    -privileged=false: 给容器扩展的权限    
    -m="": 内存限制 (格式:<number><optional unit>, unit单位 = b, k, m or   
  
g)    
    -n=true: 允许镜像使用网络    
    -p=[]: 匹配镜像内的网络端口号    
    -rm=false:当容器退出时自动删除容器 (不能跟 -d一起使用)    
    -t=false: 分配一个伪造的终端输入    
    -u="": 用户名或者ID    
    -dns=[]: 自定义容器的DNS服务器    
    -v=[]: 创建一个挂载绑定：[host-dir]:[container-dir]:[rw|ro].如果容器  
  
目录丢失，docker会创建一个新的卷    
    -volumes-from="": 挂载容器所有的卷    
    -entrypoint="": 覆盖镜像设置默认的入口点    
    -w="": 工作目录内的容器    
    -lxc-conf=[]: 添加自定义-lxc-conf="lxc.cgroup.cpuset.cpus = 0,1"   
    -sig-proxy=true: 代理接收所有进程信号(even in non-tty mode)    
    -expose=[]: 让你主机没有开放的端口    
    -link="": 连接到另一个容器(name:alias)    
    -name="": 分配容器的名称，如果没有指定就会随机生成一个    
    -P=false: Publish all exposed ports to thehost interfaces 公布所有显  
  
示的端口主机接口   
  
8、docker常用命令总结  
  
    docker pull <镜像名:tag> #从官网拉取镜像    
    docker search <镜像名> #搜索在线可用镜像名   
  
8.1查询容器、镜像、日志  
  
    docker top <container> #显示容器内运行的进程    
    docker images #查询所有的镜像，默认是最近创建的排在最上。    
    docker ps #查看正在运行的容器    
    docker ps -l #查看最后退出的容器的ID    
    docker ps -a #查看所有的容器，包括退出的。    
    docker logs {容器ID|容器名称} #查询某个容器的所有操作记录。    
    docker logs -f {容器ID|容器名称} #实时查看容易的操作记录。   
  
8.2删除容器与镜像  
  
    docker rm$(docker ps -a -q) #删除所有容器    
    docker rm <容器名or ID> #删除单个容器    
    docker rmi <ID> #删除单个镜像    
    docker rmi$(docker images | grep none | awk '{print $3}' | sort -r)   
  
#删除所有镜像   
  
8.3启动停止容器  
  
    docker stop <容器名or ID> #停止某个容器    
    docker start <容器名or ID> #启动某个容器    
    docker kill <容器名or ID> #杀掉某个容器   
  
8.4容器迁器  
  
    docker export <CONTAINER ID> > /home/export.tar #导出    
    cat /home/export.tar | sudo docker import - busybox-1-export:latest   
  
# 导入export.tar文件    
    docker save debian> /home/save.tar #将debian容器打包    
    docker load< /home/save.tar #在另一台服务器上加载打包文件   
  
save和export的对比参考地址：  
  
http://www.fanli7.net/a/bianchengyuyan/C\_\_/20140423/452256.html  
  
8.5运行一个新容器  
  
    #运行一个新容器，同时为它命名、端口映射。以debian02镜像为例    
    docker run -h="redis-test" --name redis-test -d -p 51000:22 -  
  
p51001:3306 -p 51003:6379 -p 51004:6381 -p 51005:80 -p 51006:8000 -p   
  
51007:8888 debian02 /etc/rc.local    
       
    #从container中拷贝文件，当container已经关闭后，在里面的文件还可以拷  
  
贝出来。    
    sudo docker cp 7bb0e258aefe:/etc/debian\_version . #把容器中  
  
的/etc/debian\_version拷贝到当前目录下。   
  
8.6 docker Dockfile镜像制作  
  
    root@yangrong:/data# cat Dockerfile    
    FROM ubuntu/testa #这是基础镜像    
    CMD ["/root/start.sh"] #这是启动命令    
    root@yangrong:/data# docker build -t <新镜像名> ./ #生成新的镜像   
  
Dockfile更多参数参考：  
  
http://www.tuicool.com/articles/FRvAbe  
  
http://www.colorscode.net/2014/01/04/howto-build-image-with-automatic-  
  
startup-ssh-service-from-dockerfile/