## Docker介绍

Docker 是 Docker.Inc 公司开源的一个基于 LXC技术之上构建的Container容器引擎， 源代码托管在 GitHub 上, 基于Go语言并遵从Apache2.0协议开源。

     Docker是通过内核虚拟化技术（namespaces及cgroups等）来提供容器的资源隔离与安全保障等。由于Docker通过操作系统层的虚拟化实现隔离，所以Docker容器在运行时，不需要类似虚拟机（VM）额外的操作系统开销，提高资源利用率。优点如下：

面向开发：简化环境配置

面向测试：多版本测试

面向运维：环境一致

面向架构：自动化扩容（微服务）

## 制作镜像

### 安装配置Docker

# 安装

Docker yum install -y docker

vim /etc/sysconfig/docker#新增如下参数：

OPTIONS='--registry-mirror=https://mirror.ccs.tencentyun.com'

# 设置开机启动

systemctl enable docker.service

#重启docker服务：

systemctl restart docker

### 制作基础镜像

拉取 centos官方基础镜像docker pull centos

查看当前镜像docker images

运行并进入镜像：

docker run -ti docker.io/centos:latest /bin/bash

此时，终端已经进入了镜像里面，现在我们可以根据自己的需求安装额外的组件，比如我这次需要用到crontab任务计划服务、进程守护supervisor等，那么直接在这个终端开始操作：

[root@0d7f7b8769d9 /]# yum install -y epel-release crontabs

[root@0d7f7b8769d9 /]# yum install -y python-pip

[root@0d7f7b8769d9 /]# pip install --upgrade pip

[root@0d7f7b8769d9 /]# pip install supervisor

Ps：上面的PS提示符中的 0d7f7b8769d9 就是本次启动的 CONTAINER ID ,在下面的commit步骤即将用到。完成必要组件安装之后，按下 Ctrl +D 退出系统，接着使用 docker commit 命令创建新镜像，比如命名为 nginx-proxy-base，版本latest：

docker commit 0d7f7b8769d9 centos/nginx-proxy-base:latest

执行完成后，可以使用 docker images 查看刚创建的镜像：

到此，我们就创建了一个自定义的Docker基础镜像（Ps：基础镜像类似一个VM虚拟机的快照，方便后续步骤都可以从这个基础上重新制作。）

Ps：这里展示的是进入Docker里面通过手工部署的方式，其实我们还可以通过DockerFile来完成上述所有操作，可以极大的减小Docker镜像的体积。

### 编写Dockerfile自动化构建可ssh登陆的镜像

Dockerfile是由一系列命令和参数构成的脚本，这些命令应用于基 础镜 像并最终创建一个新的镜像。

接下来就利用之前我们pull下来的centos镜像来制作一个可以ssh登陆的镜像吧

这里需要创建三个文件 Dockerfile run.sh sources163.txt，处于同级目录。

dockerfile如下内容如下：

FROM centos:7.5.1804

#设置基础镜像镜像

MAINTAINER haya #指定作者

COPY sources163.txt /etc/apt/sources.list

#更换国内源

RUN apt-get -y update

RUN apt-get install -y openssh-server --force-yes

#安装ssh服务
RUN mkdir -p /root/.ssh

#创建ssh目录
ADD run.sh /run.sh

#将run.sh添加到容器内

RUN chmod 755 /run.sh

EXPOSE 22

#为暴漏容器运行时的监听端口给外部

EXPOSE 80

EXPOSE 443

CMD ["/run.sh"]

#容器启动时要运行的命令

run.sh文件内容：

#!/bin/bash

service ssh start

### 构建镜像

命令为：docker build -t="[name]:[tag]" ./ ，比如：

docker build -t="centos/nginx-proxy:v1" ./

build之后，再执行docker images就可以看到刚刚创建的镜像：

接着，可以下测试镜像是否能正常运行，命令语法大致如下：

docker run -v [宿主目录]:[镜像目录]-ti -p [宿主端口]:[镜像端口] 镜像名称:版本

若加上 -d 参数，docker将会后台运行，这里我们想看下刚刚创建的镜像是否正常， 所以采用前台运行模式，命令如下：

docker run -v /data/docker:/data/wwwlogs -ti -p 80:80 centos/nginx-proxy:v1

## 三、有仓库

### 拉取私有仓库registry

docker pull registry

### 拉起仓库

docker run -d -p 5000:5000-v /data/images:/tmp/registry docker.io/registry

### 推送镜像

第一步查看镜像列表时，拿到需要推送的镜像的ID，比如 f2ed91429b31

①、先打tag，语法如下：

docker tag [image id] [仓库地址]/[命名空间]/[镜像名称]:[版本]

②、然后push，语法如下：

docker push [仓库地址]/[命名空间]/[镜像名称]

执行过程如下所示：

docker tag f2ed91429b31 localhost:5000/centos/nginx-proxy:latest

docker push localhost:5000/centos/nginx-proxyThe push refers to a repository

完成后，执行docker images就可以看到刚刚提交的镜像了：

③、测试拉取：

现在可以在本机（本机可以先删除在拉取）或另找一台服务器进行docker pull拉取测试。

比如，先在宿主机上删除这个镜像：

docker rmi localhost:5000/centos/nginx-proxy

此时，docker images 列表已经消失，再执行 docker pull 就又回来了。

docker pull localhost:5000/centos/nginx-proxy

### 离线方案

当私有仓库无法使用时（比如存在网络限制），我们还可以将镜像保存为一个tar包，方便离线使用，使用也非常简单：

①、export / import 方案

使用 docker ps -a 查看当前正在运行的docker镜像列表，得到对应的 CONTAINER ID，执行如下语句可以将运行中的镜像导出到指定tar包：

docker export [CONTAINER ID]> centos-nginx-proxy-latest.tar

有了tar包之后，就可以使用 import 来导入：

cat centos-nginx-proxy-latest.tar | docker import- centos/nginx-proxy:v1

②、save / load 方案

使用 docker images 查看本地已有镜像列表，得到对应的IMAGE ID，然后执行如下语句可以将本地已存在镜像保存到指定tar包：

docker save [IMAGE ID]> centos-nginx-proxy-latest.tar

后面则可以使用 load 来加载tar包镜像：

docker load < centos-nginx-proxy-latest.tar

**两种方案的区别：**

•export 只能导出正在运行的镜像，而 save 可以直接导出本地镜像；

•export 导出的镜像文件一般会小于 save 保存的镜像（本文实践数据：相差38MB）；

•export 导出（import导入）是根据容器拿到的镜像，再导入时会丢失镜像所有的历史，所以无法进行回滚操作（docker tag，而save保存（load加载）的镜像，没有丢失镜像的历史，可以回滚到之前的层（layer）。

## 四、其他知识

### Docker容器互联

网上大量文章采用docker link进行连接，根据目前官方文档 —link是一个被遗弃，最终可能会被删除的命令。推荐使用docker networks进行容器连接，我们将启动两个容器，一个apache-php 一个mysql，然后用docker network对容器进行连通。

首先创建一个名为test-net的网络，默认连接方式bridge.

IMG_256

启动一个mysql容器，映射3306端口，连接到test-net网络，设置别名为db，设置mysql密码为password

IMG_257

启动一个apache-php容器，映射80端口，连接到test-net网络，设置别名为web

IMG_258

### Docker容器和宿主机文件传递

从容器里面拷文件到宿主机：

 docker cp 容器名：要拷贝的文件在容器里面的路径   要拷贝到宿主机的相应路径

从宿主机拷文件到容器里面：

docker cp 要拷贝的文件路径 容器名：要拷贝到容器里面对应的路径

### Docker文件挂载

Docker容器启动的时候，如果要挂载宿主机的一个目录，可以用-v参数指定。

譬如我要启动一个centos容器，宿主机的/test目录挂载到容器的/soft目录，可通过以下方式指定：

# docker run -it -v /test:/soft centos /bin/bash

### Docker容器进入的4种方式

使用docker attach(多个窗口易阻塞)

使用SSH(需开启sshd)

使用nsenter(需安装)

使用exec

docker exec -it 775c7c9ee1e1 /bin/bash