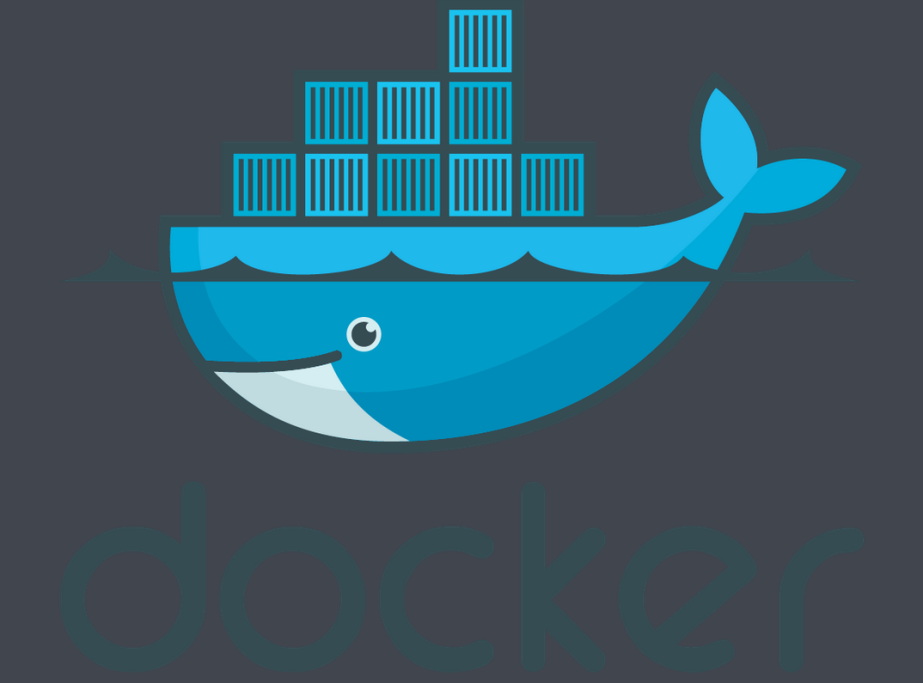
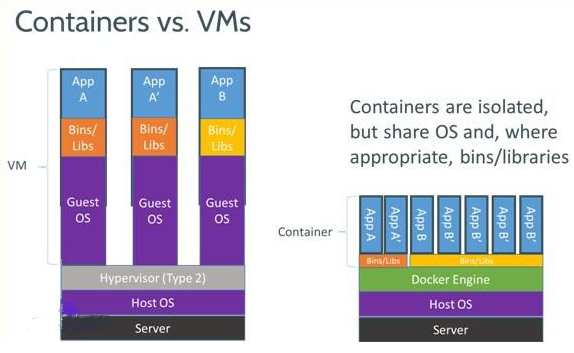
# 概念简介

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。

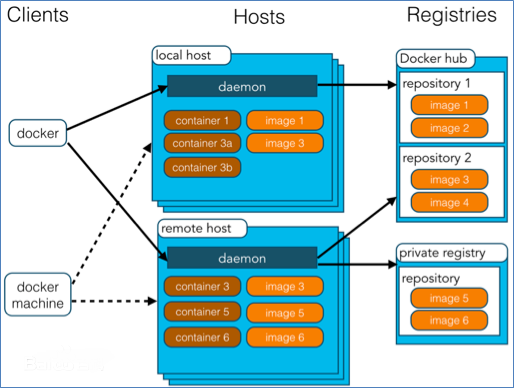


docker设想是交付运行环境如同海运，OS如同一个货轮，每一个在OS基础上的软件都如同一个集装箱，用户可以通过标准化手段自由组装运行环境，同时 集装箱的内容可以由用户自定义，也可以由专业人员制造。这样，交付一个软件，就是一系列标准化组件的集合的交付，如同乐高积木，用户只需要选择合适的积木 组合，并且在最顶端署上自己的名字(最后个标准化组件是用户的app)。这也就是基于docker的PaaS产品的原型。



Docker 使用客户端-服务器 (C/S) 架构模式，使用远程API来管理和创建Docker容器。Docker 容器通过 Docker 镜像来创建。容器与镜像的关系类似于面向对象编程中的对象与类。

|  |  |
| --- | --- |
| **Docker** | **面向对象** |
| 容器 | 对象 |
| 镜像 | 类 |



# 安装Docker私服

## 安装Docker

参考：<https://docs.docker.com/engine/installation/linux/centos/#/install-with-yum>

### 查看内核要在3.10以上

uname -r

### 添加配置文件

tee /etc/yum.repos.d/docker.repo <<-'EOF'

[dockerrepo]

name=Docker Repository

baseurl=https://yum.dockerproject.org/repo/main/centos/7/

enabled=1

gpgcheck=1

gpgkey=https://yum.dockerproject.org/gpg

EOF

### 安装docker

yum install docker-engine

（安装docker时，默认的安装位置是/var/lib/docker）

### 启动

service docker start

### 查看version

docker version

### 校验安装是否正常

docker run hello-world

## 安装docker compose

（下边安装harbor时会用到 docker compose）

参考<https://docs.docker.com/compose/install/>

### 下载docker-compose

curl -L https://github.com/docker/compose/releases/download/1.8.0/docker-compose-`uname -s`-`uname -m` > /usr/local/bin/docker-compose

### 修改权限

chmod +x /usr/local/bin/docker-compose

### 查看版本

docker-compose --version

## 安装harbor

Harbor是VMware公司最近开源的企业级Docker Registry项目, 其目标是帮助用户迅速搭建一个企业级的Docker registry服务。它以Docker公司开源的registry为基础，提供了管理UI, 基于角色的访问控制(Role Based Access Control)，AD/LDAP集成、以及审计日志(Audit logging) 等企业用户需求的功能，同时还原生支持中文。Harbor的每个组件都是以Docker容器的形式构建的，使用Docker Compose来对它进行部署。

### 下载harbor

git clone <https://github.com/vmware/harbor>

### 修改配置文件vim darbor/ Deploy/ harbor.cfg

hostname = reg.mydomain.com修改harbor所在服务器ip

### 在harbor/Deploy目录下运行命令，更新配置

./prepare

### 修改/lib/systemd/system/docker.service文件，添加私服IP

在ExecStart所在的行末尾添加--insecure-registry 172.16.6.151

### 重启docker daemon 和 service

(命令：systemctl daemon-reload 和 systemctl restart docker.service)

### 在harbor/Deloy目录运行build和启动命令（harbor是运行在这个命令生成的镜像基础上的）（此命令运行时间较长）

docker-compose up -d

### 查看容器状态，均为up状态表示正常

docker-compose ps

### 打开浏览器，登录<http://172.16.6.151/>

默认用户名和密码设置在配置文件darbor/ Deploy/ harbor.cfg中。

### 常规测试

（1）从docker hub上下载hello-world镜像

docker pull hello-world

（2）给镜像打标签，以便上传到私服，其中library是harbor默认提供的项目

docker tag hello-world 172.16.6.151/library/hello-world

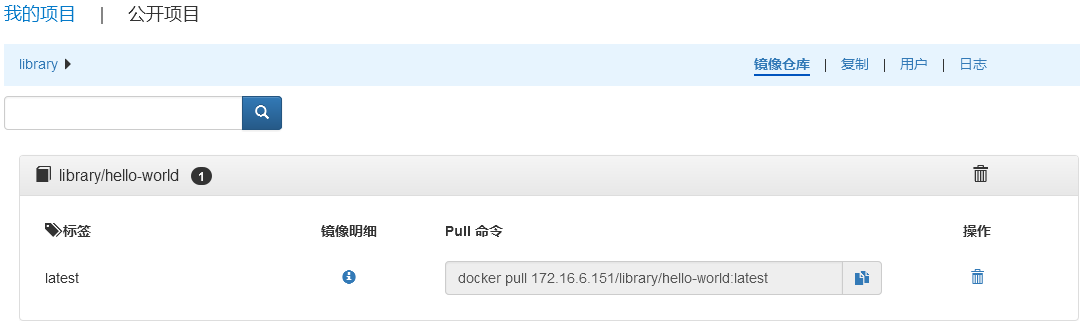
（3）登录镜像私服上传镜像，执行以下指令，再输入用户名和密码

docker login 172.16.6.151

docker push 172.16.6.151/library/hello-world

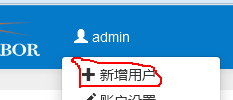
（4）在harbor界面上查看，可看到已成功push到私有镜像仓库

（5）仓库在服务器上的位置是：/data/registry/docker/registry/v2/repositories/



## 新增用户

用户指导链接：https://github.com/vmware/harbor/blob/master/docs/user\_guide.md



# 构建编译相关部分

## 编译机安装Docker

### 编译机6.112安装Docker参考的 二、（一），报错 无法访问镜像

[Errno 14] HTTP Error 404 - Not Found

Trying other mirror.

### 于是改用rpm包方式安装

下载rpm安装包

docker-engine-1.12.1-1.el7.centos.x86\_64.rpm

安装报错缺少依赖的包

然后根据报错安装了各个依赖包：

rpm –ivh docker-engine-selinux-1.12.1-1.el7.centos.noarch.rpm

yum install device-mapper-libs

rpm -ivh docker-engine-1.12.1-1.el7.centos.x86\_64.rpm

yum install libsystemd\*

最后安装

rpm -ivh docker-engine-1.12.1-1.el7.centos.x86\_64.rpm

成功！！

### 配置

配置/lib/systemd/system/docker.service，在ExecStart所在的行末尾添加--insecure-registry=172.16.6.151

### 测试

运行构建脚本，卡在这里（因为安装完成，配置修改过后 需要重启；解决方法：把脚本中调用安装业务脚本的部分注释掉）

|  |
| --- |
| Plugin configuration has changed. Restart RabbitMQ for changes to take effect.  The following plugins have been enabled:  rabbitmq\_tracing  Plugin configuration has changed. Restart RabbitMQ for changes to take effect.  Redirecting to /bin/systemctl restart crond.service  cat: install: No such file or directory |

**命令：docker images**

查看本地存在哪些image文件

（实际存放images和container的地方就是/var/lib/docker/devicemapper/devicemapper目录下的metadata和data两个文件

）

**命令：docker rmi 仓库名:TAG 、 docker rmi (IMAGE ID)**

删除本地没用的image文件（编译机上需要定期删除历史image文件，不然占空间会越来越大）

例如：

docker rmi 172.16.6.151/mustang/uas-c:20160905

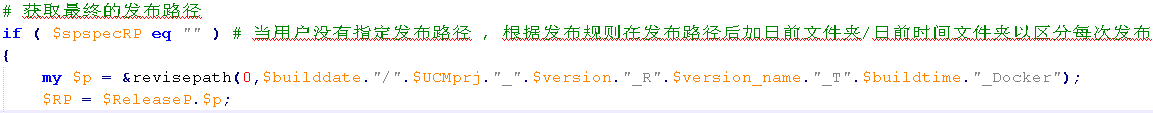
docker rmi c54a2cc56cbb

## 编译脚本修改

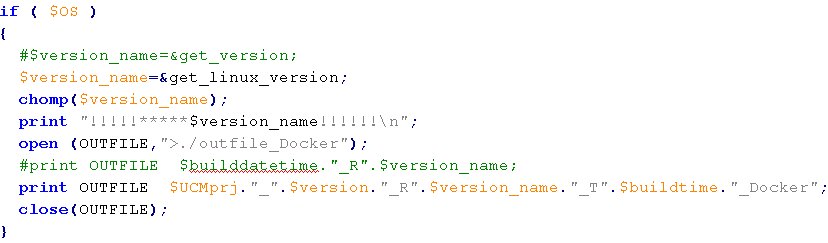
为了image TAG和99目录保持一致，增加Mustang\10-common\build\buildscript\_docker目录

**修改了sp.pl**

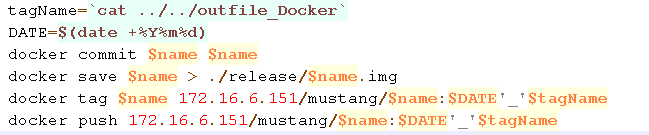
1. 发布到0.99的目录，默认发到date目录下（而不是date-SP目录）



1. 发布目录名称，写入临时文件outfile\_Docker中，方便docker打包脚本读取，打TAG时和172.16.0.99目录名称保持一致，方便查看对应的版本。



**修改install.Sh**



**以后编译dockerfile命令为：**

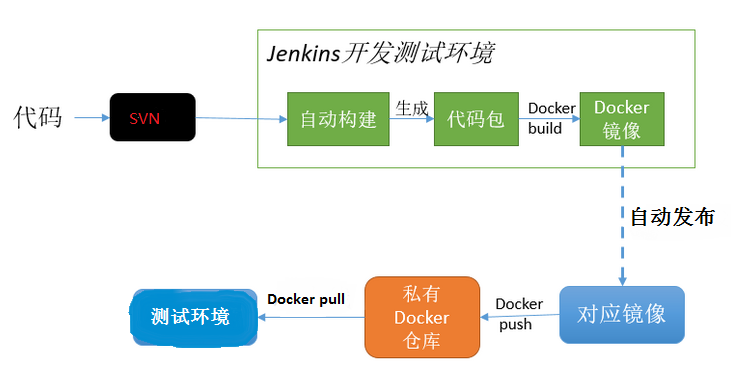
cd 10-common/build/buildscript\_docker

perl sp.pl daily\_64bit dockerfile

# 总结

首先了解Docker是做什么的，然后规划在目前工作流程中如何修改添加，最后和开发人员商讨实施细节。

Docker在构建发布流程中的位置，如下图所示



安装过程中遇到的错误，根据报错信息提示，查找安装对应的文件。