第十章　配置 docker 静态 IP 地址-配置 docker 私有仓库

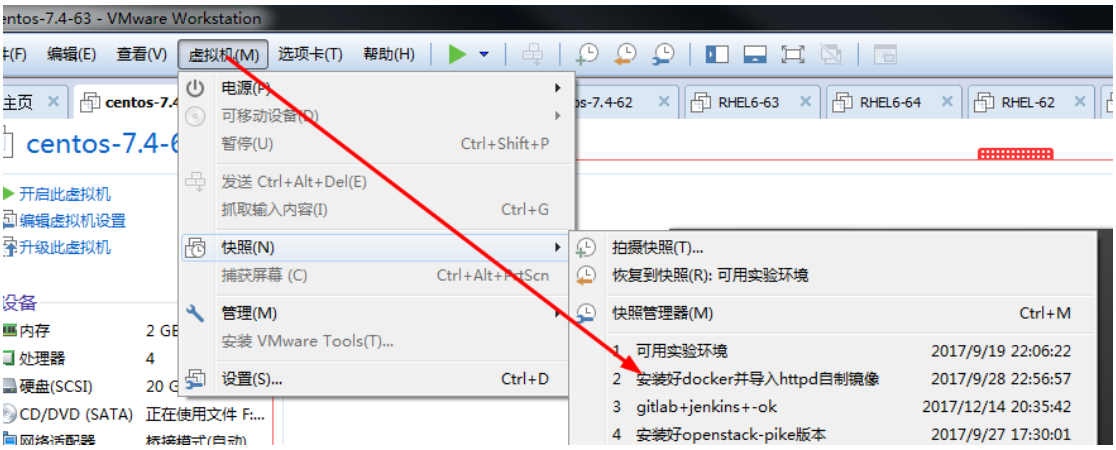
本节所讲内容：

10.1　创建 docker 静态化 IP

10.2　创建 docker 私有化仓库

10.3　使用阿里云私有仓库存储自己的 docker 镜像

实验环境：　一个还原到之前安装了 docker 的虚拟机快照：



10.1　创建 docker 静态化 IP

10.1.1　Docker 的 4 种网络模式

1、Docker 有以下 4 种网络模式：

host 模式，使用--net=host 指定。

container 模式，使用--net=container:NAME_or_ID 指定。

none 模式，使用--net=none 指定。

bridge 模式，使用--net=bridge 指定，默认设置。

默认选择 bridge 的情况下，容器启动后会通过 DHCP 获取一个地址，这可能不是我们想要的，在 centos7 系统上， docker 环境下可以使用 pipework 脚本对容器分配固定 IP（这个 IP 可以是和物理机同网段 IP）。

注： docker 默认是 bridge（--net=bridge）模式，相当于 VMware 中 NAT 模式。

docker 环境下可以使用 pipework 脚本对容器分配固定 IP，相当于 VMware 中桥接模式。

注：Pipework 有个缺陷，容器重启后 IP 设置会自动消失，需要重新设置。

10.1.2　配置桥接网络：

桥接本地物理网络的目的,是为了局域网内用户方便访问 docker 实例中服务,不要需要各种端口映射即可访问服务。　但是这样做,又违背了 docker 容器的安全隔离的原则,工作中辩证的选择.

创建桥设备：

安装包：

[root@xuegod63 ~]# rpm -ivh /mnt/Packages/bridge-utils-1.5-9.el7.x86_64.rpm

把 eth0 绑到 br0 桥设备上：

[root@xuegod63 ~]# cd /etc/sysconfig/network-scripts/

[root@xuegod63 network-scripts]# vim ifcfg-ens33#编辑配置文件为以下内容
[root@xuegod63 network-scripts]# vim ifcfg-ens33
TYPE="Ethernet"
BOOTPROTO="none"
DEFROUTE="yes"
IPV4_FAILURE_FATAL="no"
IPV6INIT="yes"
IPV6_AUTOCONF="yes"
IPV6_DEFROUTE="yes"
IPV6_FAILURE_FATAL="no"
NAME="eno16777736"
UUID="7a556ff6-f865-4549-b08f-9e526c9bb638"
DEVICE="eno16777736"
ONBOOT="yes"
IPADDR="192.168.1.63" #删除这些 IP 地址相关内容
PREFIX="24"
GATEWAY="192.168.1.1"
DNS1="8.8.8.8"
IPV6_PEERDNS="yes"
IPV6_PEERROUTES="yes"
IPV6_PRIVACY="no"
BRIDGE="br0" #在文件最后插入这一行

生成桥设备 br0 的配置文件：
[root@xuegod63 network-scripts]# vim ifcfg-br0 #创建 ifcfg-br0 文件，并写入以下内容
DEVICE="br0"
NM_CONTROLLED="yes"
ONBOOT="yes"
TYPE="Bridge"
BOOTPROTO=none
IPADDR=192.168.1.63
NETMASK=255.255.255.0
GATEWAY=192.168.1.1
DNS1=114.114.114.114
注：TYPE="Bridge" B要大写。 不大写也可以。

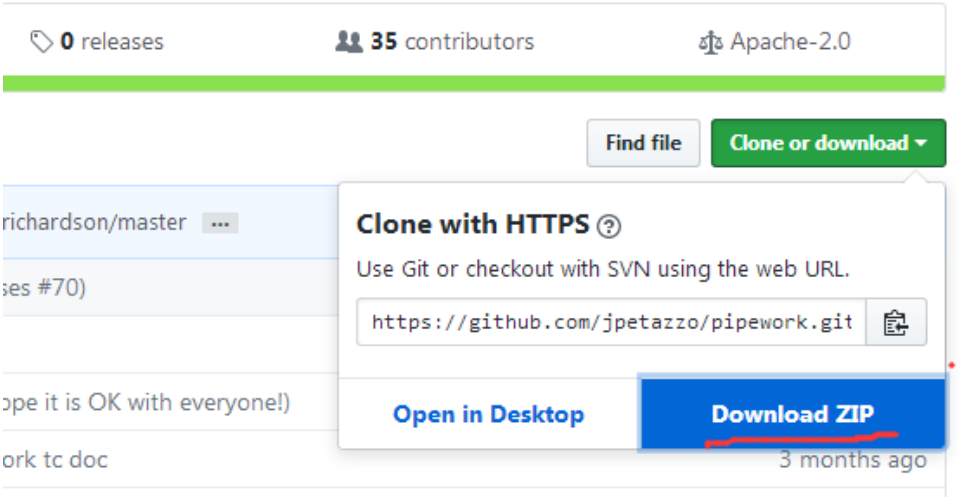
[root@xuegod63 network-scripts]# service network restart
Restarting network (via systemctl): [确定]

测试 br0：
root@xuegod63 network-scripts]# ifconfig

[root@xuegod63 network-scripts]# ping g.cn
PING g.cn (203.208.37.20) 56(84) bytes of data.
64 bytes from 203.208.37.20: icmp_seq=1 ttl=57 time=12.3 ms

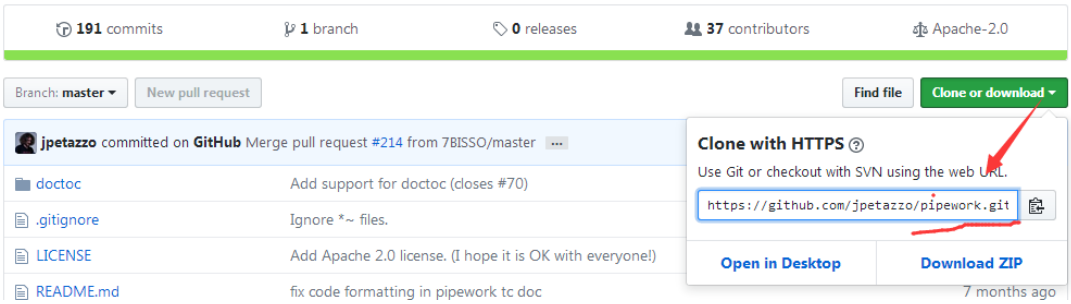
10.1.3 下载 pipework 包

方法 1：直接下载 pipework zip 包
https://github.com/jpetazzo/pipework



把 pipework-master.zip 上传到 Linux 中

扩展：
方法 2：使用 git 获得：
git 下载链接：https://github.com/jpetazzo/pipework



下载 pipework 工具:https://github.com/jpetazzo/pipework.git
[root@xuegod63 ~]# rpm -qf `which git`
git-1.8.3.1-5.el7.x86_64
[root@xuegod63 ~]# cd /opt/
[root@xuegod63 opt]# git clone https://github.com/jpetazzo/pipework.git

咱们使用方法 1：
将 pipework-master.zip 上传 xuegod63 上：
[root@xuegod63 ~]# unzip pipework-master.zip # 不需要编译，因为 pipework 是一个 shell 脚本

查看：

[root@xuegod63 ~]# vim ./pipework-master/pipework

[root@xuegod63 ~]# cp /root/pipework-master/pipework /usr/local/bin/ #方便后期使用 pipework 命令

到此 pipework 已经安装成功。

启动 docker：

[root@xuegod63 ~]# systemctl start docker

把 centos-lastest-docker-image.tar 镜像上传 Linux 上，并导入 docker 平台

[root@xuegod63 ~]# docker load -i centos-lastest-docker-image.tar

10.1.4 使用静态 IP 启动一个 docker 实例

例：以 none 模式，使用--net=none 启动一个容器，并且开启 docker 特权模式。

扩展：

--privileged=true #允许开启特权功能

大约在 docker 0.6 版以后，privileged 被引入 docker。使用该参数，container 内的 root 拥有真正的 root 权限。否则，container 内的 root 只是外部的一个普通用户权限。

使用 privileged 启动的容器，可以看到很多 host 上的设备，并且可以执行 mount。甚至允许你在 docker 容器中启动 docker 容器。不启用 privileged，容器中 root 用户不能执行 mount。

扩展： 测试 privileged 特权功能 可以：1

1、未设置 privileged 启动的容器：

[root@localhost ~]# docker run -it centos:latest bash

[root@65acccbba42f /]# ls /dev #可以看到的设备文件比较少

console fd full fuse kcore null ptmx pts random shm stderr stdin stdout tty urandom zero

[root@00931099722f /]# mount -o bind /etc /opt/

mount: permission denied

而在物理机是可以挂载成功的：

[root@xuegod63 ~]# mount -o bind /etc /opt/

[root@00931099722f /]# exit

2、使用 privileged 启动的容器

[root@xuegod63 ~]# docker run -it --privileged centos:latest bash

[root@4a51d0fde3ce /]# ls /dev/ #可以看到很多设备文件

[root@4a51d0fde3ce /]# mount -o bind /etc /opt/ #可以挂载成功

[root@4a51d0fde3ce /]# mount /dev/sda1 /opt/ #可以挂载物理机上的 sda1 分区

[root@4a51d0fde3ce /]# ls /opt/

[root@4a51d0fde3ce /]# init 0 #不行，还是使用 exit 退出 docker

Couldn't find an alternative telinit implementation to spawn.

[root@4a51d0fde3ce /]# exit

扩展结束，接着给容器配置地址

[root@xuegod63 ~]# docker ps

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

e4698f625a56 centos "bash" 30 seconds ago Up 27 seconds suspicious_colden

可以看到容器启动的 ID ,比如是 e4698f625a56

给此容器配置地址

pipework 语法：pipework 网桥名 容器实例 ID 分配给容器的 IP/掩码@网关

[root@xuegod63 ~]# pipework br0 c88c4c7f01f9 192.168.1.71/24@192.168.1.1

测试 IP：

[root@xuegod63 ~]# ping 192.168.1.71 #可以看到 docker 实例的 IP 已经可以使用

PING 192.168.1.71 (192.168.1.71) 56(84) bytes of data.

64 bytes from 192.168.1.71: icmp_seq=1 ttl=64 time=0.639 ms

进入容器,测试网络：

[root@xuegod63 ~]# docker exec -it 87fadc0249a9 /bin/bash #进入容器

[root@6e38ee3f9672 /]# cat /etc/resolv.conf

Generated by NetworkManager

search xuegod63.cn

nameserver 114.114.114.114

[root@e4698f625a56 /]# yum install -y net-tools #安装 ifconfig 命令

[root@e4698f625a56 /]# ifconfig

eth1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500

inet 192.168.1.71 netmask 255.255.255.0 broadcast 192.168.1.255

[root@e4698f625a56 /]# route -n

Kernel IP routing table

Destination	Gateway	Genmask	Flags	Metric	Ref	Use	Iface
0.0.0.0	192.168.1.1	0.0.0.0	UG	0	0	0	eth1
192.168.1.0	0.0.0.0	255.255.255.0	U	0	0	0	eth1

到此，docker 实例配置静态 IP 成功。

总结：

1、创建一个 br0 桥接设备

2、下载 pipework 包并安装

3、安装并运行 docker

4、导入 centos docker 镜像

5、启动一个 docker 实例 注意加参数： --net=none --privileged=true

6、使用 pipework 给 docker 实例配置 IP

实战 1： 使用静态 IP 启动一个 web 服务器

[root@1e1db6c6c17c ~]# yum install httpd -y #安装

[root@1e1db6c6c17c ~]# systemctl start httpd #这个方式，无法启动

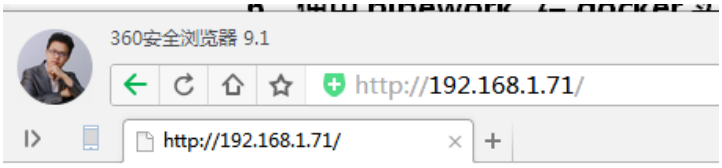
[root@1e1db6c6c17c ~]# httpd #直接运行 httpd 命令

[root@1e1db6c6c17c ~]# netstat -antup | grep 80 #发现 80 已经监听

[root@1e1db6c6c17c ~]# cd /var/www/html/ #

[root@1e1db6c6c17c ~]# echo aaaaa > index.html

查看



aaaaa

10.2 创建 docker 私有化仓库

有时候使用 Docker Hub 这样的公共仓库可能不方便（有时候无法访问），用户可以创建一个本地仓库供私人使用，这里使用官方提供的工具 docker-registry 来配置私有库

1、使用官方提供的工具来配置

docker-registry 是官方提供的工具，可以用于构建私有的镜像仓库。

私有仓库好处：

1、节约带宽

2、可以自己定制系统

docker-registry 这个工具是一个镜像，直接下载并使用 registry 镜像启动 docker 实例就可以了。

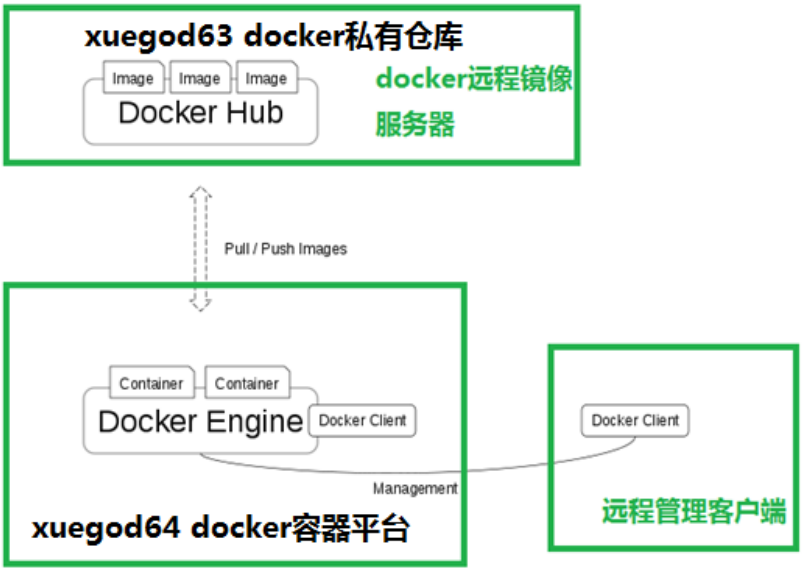
有了 docker 以后，所有软件不再以 office.exe 或 lrzsz.rpm 形式发布，而以 docker 镜像发布。你只需要下载 docker 镜像并运行一个 docker 实例。

实验环境：

docker 私有仓库地址：192.168.1.63

docker 服务器地址 ： 192.168.1.64 ,xuegod64 会使用 xuegod63 上 docker 私有仓库来 pull/push 镜像

实验拓扑图：



10.2.1 配置 xuegod63 为 docker 私有仓库

[root@xuegod63 ~]# systemctl stop firewalld

[root@xuegod63 ~]# systemctl disable firewalld

[root@xuegod63 ~]# iptables -L -n

1、 #不能关闭防火墙，因为 docker 后期端口转发，需要使用 firewalld

[root@xuegod63 ~]# systemctl start firewalld.service #必需开启防火墙，因为后面要使用 5000 端口进行通信

2、关闭 selinux

[root@xuegod63 ~]# vim /etc/sysconfig/selinux

改：SELINUX = enforcing

为：SELINUX=disabled

[root@xuegod63 ~]#reboot

[root@xuegod63 ~]# getenforce

Disabled

10.2.2 配置 xuegod63 为 docker 私有仓库服务端

1.启动 docker

[root@xuegod63 ~]# systemctl start docker

2.拉取 registry 镜像。 registry 镜像中包括搭建本地私有仓库的软件：

把 registry.tar 上传到 Linux 上

方法一：导入本地镜像：

[root@xuegod63 ~]# docker load -i registry.tar

方法二： 在线导入镜像，比较慢

[root@xuegod63 ~]# docker pull registry

Using default tag: latest

Trying to pull repository docker.io/library/registry ...

latest: Pulling from docker.io/library/registry

acf34ba23c50: Waiting

error pulling image configuration: Get https://dseasb33srnrn.cloudfront.net/registry-v2/docker/registry/v2/blobs/sha256/13/136c8b16df203ef26b2f39e24bd3f403b63be67610ec99a5b5af0cceac5c1b51/data?Expires=1491661458&Signature=VpBWJnckUbRqJol8EWTw2ZswQ-xOjrbqDfUstwjJwA55NoaOlESDpUC2AOloQXQRXx~F7-DGwaOY4bjJpdymnVhyv5ylO2ZB1tlkgANsNYhyoKOSyT8IycW94Cee~GaXqdcwkdECsLqWbRW1S297k4jK2GXTtaZqUsBrrmx3oAQ_&Key-Pair-Id=APKAJECH5M7VWIS5YZ6Q: EOF

注： 这是因为访问不了国外的网址导致的。

解决：多再试试几次， 终于一次，你不会被墙，可以访问成功的。

扩展：下载 registry 镜像时，有时会访问不到国外网站，导致下载失败

3.查看 registry 镜像

[root@xuegod63 ~]# docker images

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
docker.io/registry	latest	047218491f8c	3 weeks ago	33.17 MB

扩展：.io 域名

.io 是[british indian ocean territory] 英属印度洋领地的简写

4.从 Docker HUB 上拉取一个镜像测试

方法一：本地导入

上传 busybox.tar 镜像到 Linux 系统上，作为测试镜像。

[root@xuegod63 ~]# docker load -i busybox.tar

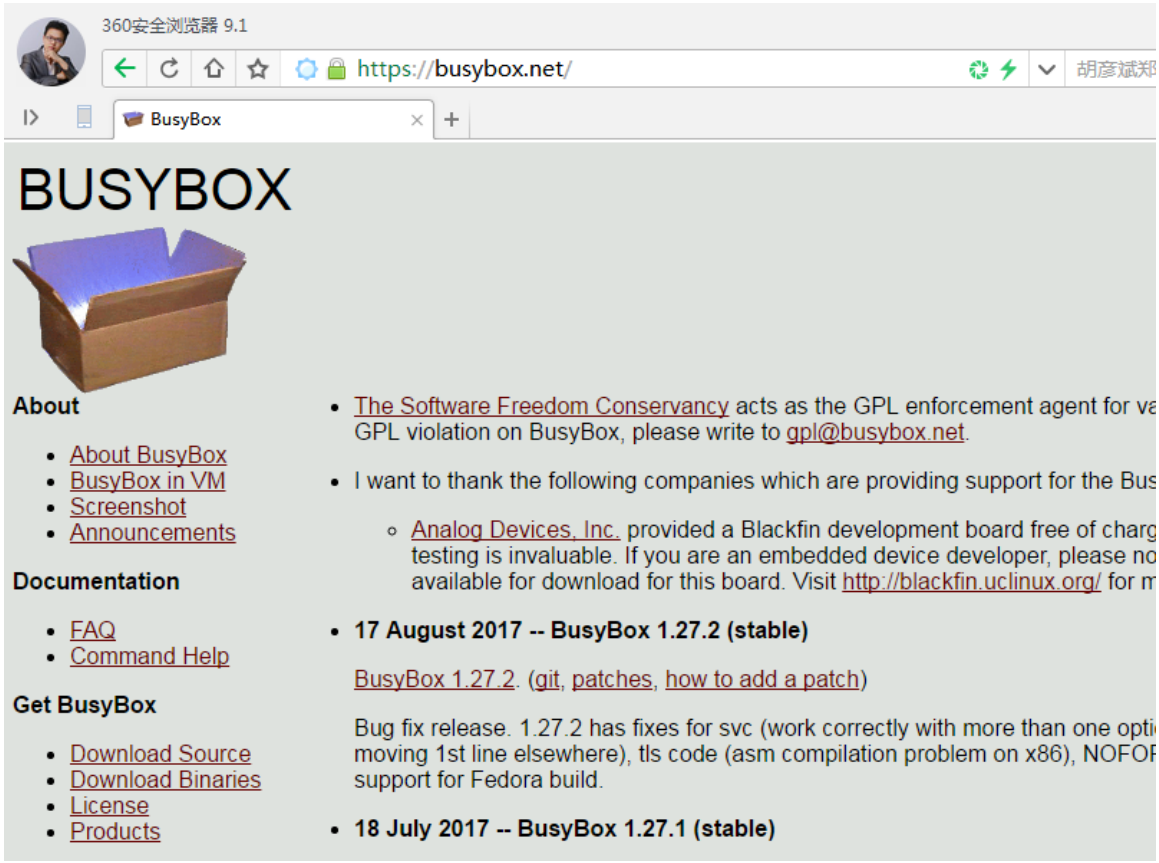
[root@xuegod63 ~]# docker images

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
docker.io/busybox	latest	00f017a8c2a6	2 weeks ago	1.11 MB
docker.io/registry	latest	047218491f8c	3 weeks ago	33.17 MB

注：

BusyBox 概述： BusyBox 是一个集成了一百多个最常用 Linux 命令和工具的软件。BusyBox 包含了 BusyBox 包含了一些简单的工具，例如 ls、cat 和 echo 等等，还包含了一些更大、更复杂的工具，例 grep、find、mount 以及 telnet。有些人将 BusyBox 称为 Linux 工具里的瑞士军刀。简单的说 BusyBox 就好像是个大工具箱，它集成压缩了 Linux 的许多工具和命令，也包含了 Android 系统的自带的 shell。

官网： www.busybox.net



方法二：在线安装

[root@xuegod63 opt]# docker pull busybox

Using default tag: latest

Trying to pull repository docker.io/library/busybox ...

latest: Pulling from docker.io/library/busybox

5、创建镜像链接或为基础镜像打个标签（复制一个镜像并起一个名字）

语法：　　docker　tag　　镜像名:标签　　私有仓库地址/镜像名：标签

[root@xuegod63　　~]#　　docker　　tag　　docker.io/registry:latest 192.168.1.63:5000/busybox:latest

注：　不写镜像标签，默认是：latest

[root@xuegod63 ~]# docker images

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
192.168.1.63:5000/busybox	latest	00f017a8c2a6	2 weeks ago	1.11 MB
docker.io/busybox	latest	00f017a8c2a6	2 weeks ago	1.11 MB
docker.io/registry	latest	047218491f8c	3 weeks ago	33.17 MB

6.修改 docker 配置文件，指定私有仓库 url

registry　　['redʒɪstri]　　记录，登记，注册

需要安装 docker-common-1.12.6-11.el7.centos.x86_64　这个包，不然没有配置文件。默认我已经安装了。

[root@xuegod63 ~]# rpm -qf　/etc/sysconfig/docker

docker-common-1.12.6-11.el7.centos.x86_64

[root@xuegod63 ~]# vim /etc/sysconfig/docker

改：4 OPTIONS='--selinux-enabled --log-driver=journald --signature-verification=false'

为：OPTIONS='--selinux-enabled --log-driver=journald --signature-verification=false --insecure-registry 192.168.1.63:5000'

注：在原文件后，追加红色标记文字。 --insecure-registry 不安全的注册。 即通信使用 http 协议。如果使用安全的通信，就使用 https

[root@xuegod63 ~]# systemctl restart docker #启动时需要等待时间长一些

到此，所有准备工作已经完成。

排错：如果不写 --insecure-registry 这个， 默认要求使用 https 通信。所以一定要有这个参数

[root@xuegod63 ~]# docker push 192.168.1.63:5000/busybox

The push refers to a repository [192.168.1.63:5000/busybox]

Get https://192.168.1.63:5000/v1/_ping: http: server gave HTTP response to HTTPS client

7. 实战：使用 registry 镜像搭建一个私有仓库

使用 registry 镜像搭建一个私有仓库。 registry 镜像中已经把搭建私有库程序安装好了，我只需要使用 registry 镜像运行一个 docker 实例就可以了。

默认情况下，Registry 程序的存放镜像信息的目录是/var/lib/registry 目录下，这样如果容器被删除，则存放于容器中的镜像也会丢失，所以我们一般情况下会指定本地物理机一个目录如/opt/registry 挂载到容器的/var/lib/registry 下，这样两个目录下都有！

注：registry 运行的实例的默认存储路径是/var/lib/registry，只是个临时目录，一段时间之后就会消失。所以使用-v 参数，指定本地持久的路径。

[root@xuegod63 ~]# docker run -d -p 5000:5000 -v /opt/registry:/var/lib/registry registry

e4698f625a56661edd2678269215ba42d4fa41c2da881768a741a72b4a3d0c60

[root@xuegod63 ~]# ls /opt/registry # 这个目录会自动创建

[root@xuegod63 ~]# docker ps

```
CONTAINER ID        IMAGE                  COMMAND                  CREATED             STATUS              PORTS                    NAMES
e4698f625a56        docker.io/registry     "/entrypoint.sh /etc/"   7 minutes ago       Up 6 minutes        0.0.0.0:5000->5000/tcp   suspicious_colden
```

[root@xuegod63 ~]# netstat -antup | grep 5000

tcp6 0 0 :::5000 :::* LISTEN 4032/docker-proxy

说明，私有库已经启动成功。

8、将刚新打好标签的 192.168.1.63:5000/busybox 镜像，push 到本地私有仓库中。

[root@xuegod63 ~]# docker push 192.168.1.63:5000/busybox

The push refers to a repository [192.168.1.63:5000/busybox]

c0de73ac9968: Pushed

latest: digest: sha256:68effe31a4ae8312e47f54bec52d1fc925908009ce7e6f734e1b54a4169081c5 size: 527

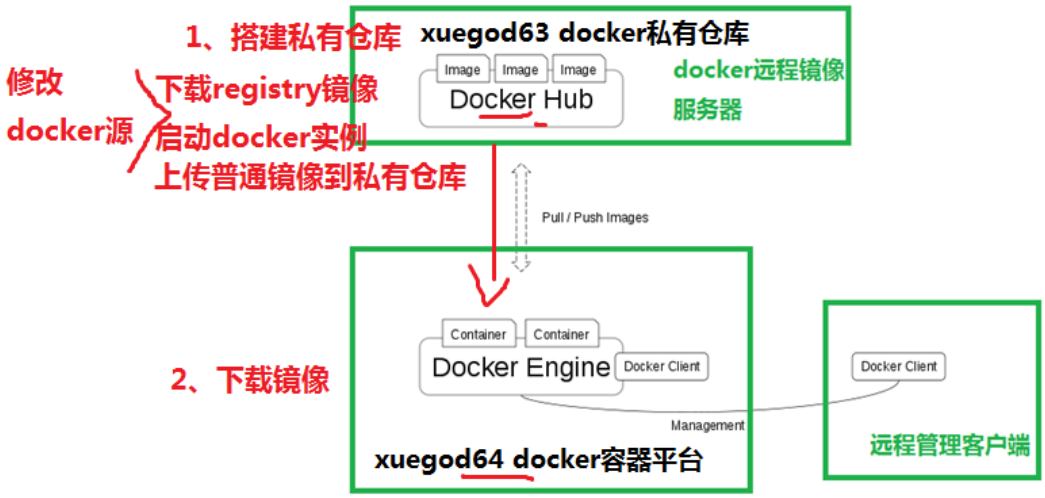
9.查看镜像的存储目录和文件

[root@xuegod63 ~]# rpm -ivh /mnt/Packages/tree-1.6.0-10.el7.x86_64.rpm

[root@xuegod63 ~]# tree /opt/registry/docker/registry/v2/

```
...
└── busybox
    ├── _layers
    │   └── sha256
    │       ├── 00f017a8c2a6e1fe2ffd05c281f27d069d2a99323a8cd514dd35f228ba26d2ff
    │       │   └── link
    │       └── 04176c8b224aa0eb9942af765f66dae866f436e75acef028fe44b8a98e045515
    │           └── link
```

总结：



10.2.3 测试私有库是否可用

在另外一台 CENTOS7.4-64-64 系统测试 pull 功能

开启 CENTOS7.4-64-64 虚拟机：

1.安装 docker

[root@xuegod63 ~]# scp -r docker-rpm 192.168.1.64:/root/

[root@xuegod63 ~]# scp /etc/yum.repos.d/docker.repo 192.168.1.64:/etc/yum.repos.d/

安装：

[root@xuegod64 ~]# yum install docker -y

2.修改 docker 配置文件，指定 docker 镜像加速结点为：私有仓库的地址

[root@xuegod64 ~]# vim /etc/sysconfig/docker

修改此行

改：4 OPTIONS='--selinux-enabled --log-driver=journald --signature-verification=false'

为：OPTIONS='--selinux-enabled --log-driver=journald --signature-verification=false --insecure-registry 192.168.1.63:5000'

注：添加红色标记文字。 --insecure-registry 不安全的注册

[root@xuegod64 ~]# systemctl restart docker #启动 docker 服务

3.测试，下载刚才上传的镜像

[root@xuegod64 ~]# docker pull 192.168.1.63:5000/busybox

[root@xuegod64 ~]# docker images #查看导入的镜像

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
192.168.1.63:5000/busybox	latest	00f017a8c2a6	2 weeks ago	1.11 MB

使用新导入的镜像，运行一个新 docker 实例：

[root@xuegod64 ~]# docker run 192.168.1.63:5000/busybox:latest echo "hello"

hello

运行成功。

创建镜像链接并命名：

[root@xuegod64 ~]# docker tag 192.168.1.63:5000/busybox busybox:v1

[root@xuegod64 ~]# docker images

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
192.168.1.63:5000/busybox	latest	00f017a8c2a6	5 months ago	1.11 MB
busybox	v1	00f017a8c2a6	5 months ago	1.11 MB

删除镜像：

语法： docker rmi 镜像名：标签

[root@xuegod64 ~]# docker rmi 192.168.1.63:5000/busybox:latest #删除镜像

[root@xuegod64 ~]# docker images #查看镜像

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
busybox	v1	00f017a8c2a6	5 months ago	1.11 MB

[root@xuegod64 ~]# docker run busybox:v1 echo aaaaaa #使用新命名的镜像运行实

10.3 使用阿里云私有仓库存储自己的 docker 镜像

放假来了讲