**使用Docker构建微服务**

**准备工作**

| **安装软件** | **版本** | **功能** | **必要程度** |
| --- | --- | --- | --- |
| Docker | 1.12.1 | Docker | 是 |
| CentOS7.0或其他系统 | 7.0 | Docker的宿主机，本章的讲解都是在CentOS 7.0下进行的。Docker现已支持Windows系统，但考虑到绝大多数Docker容器还是跑在Linux环境下的，故而只讲解Linux环境下的使用。Windows下的安装使用大致类似，请读者自行研究。 | 是 |
| Maven | 3.3.9 |  | 是 |
| JDK | 8u65 |  | 是 |

注意：

本章的讲解都是在CentOS7下进行的，建议新手使用CentOS 7.x进入学习。

**参照网站**

http://www.runoob.com/docker/centos-docker-install.html

**CentOS下JDK 1.8的安装**

(1) 到Oracle官网下载好 jdk-8u65-linux-x64.rpm 备用

(2) 卸载系统自带java

java -**version** *# 如果有结果出来，则说明自带了java*

rpm -**qa**|**grep** java *# 查询出已经安装的java*

yum -**y** remove **[**上面查出来的东西，多个用空格分隔]

(3) 安装JDK

cd /usr

mkdir /usr/java

rpm -ivh jdk-8u65-linux-x64.rpm

(4) 配置环境变量：

**vim** /etc/**profile**

找到： export PATH USER LOGNAME MAIL HOSTNAME HISTSIZE HISTCONTROL 这一行，并在其下面一行添加如下内容：

*# 设置java环境变量*

export JAVA\_HOME**=**/usr/java/jdk1.8.0\_65 *# 根据情况修改*

export PATH**=**$JAVA\_HOME/bin:$PATH

export CLASSPATH**=**.:$JAVA\_HOME/lib/dt.jar:$JAVA\_HOME/lib/tools.jar

(5) 使环境变量生效：

**source** /etc/profile

(6) 测试：

java -version

javac

**CentOS下Maven的安装**

Maven的安装比较简单，只需要下载后解压，配置环境变量即可。

(1) 下载并解压：

cd /opt

wget http:*//apache.fayea.com/maven/maven-3/3.3.9/binaries/apache-maven-3.3.9-bin.tar.gz*

tar -zxvf apache-maven-3.3.9-bin.tar.gz

(2) 配置环境变量：

**vim** /etc/**profile**

找到export PATH USER LOGNAME MAIL HOSTNAME HISTSIZE HISTCONTROL ， 并在其下面一行添加如下内容：

*# 设置Maven环境变量*

export MAVEN\_HOME**=**/opt/apache-maven-3.3.9/

export PATH**=**$MAVEN\_HOME/bin:$PATH

(3) 使环境变量生效：

**source** /etc/profile

(4) 测试

mvn -version

(5) 修改Maven配置

本地仓库路径配置：

*<!-- 本地仓库路径配置 -->*

<localRepository>/path/to/local/repo</localRepository>

国内Maven镜像配置：

<mirror>

<id>aliyun</id>

<mirrorOf>central</mirrorOf>

<name>Aliyun Central mirror</name>

<url>http://maven.aliyun.com/nexus/content/groups/public</url>

</mirror>

## CentOS下安装Openssh

使用下面命令安装ssh server/ssh client：

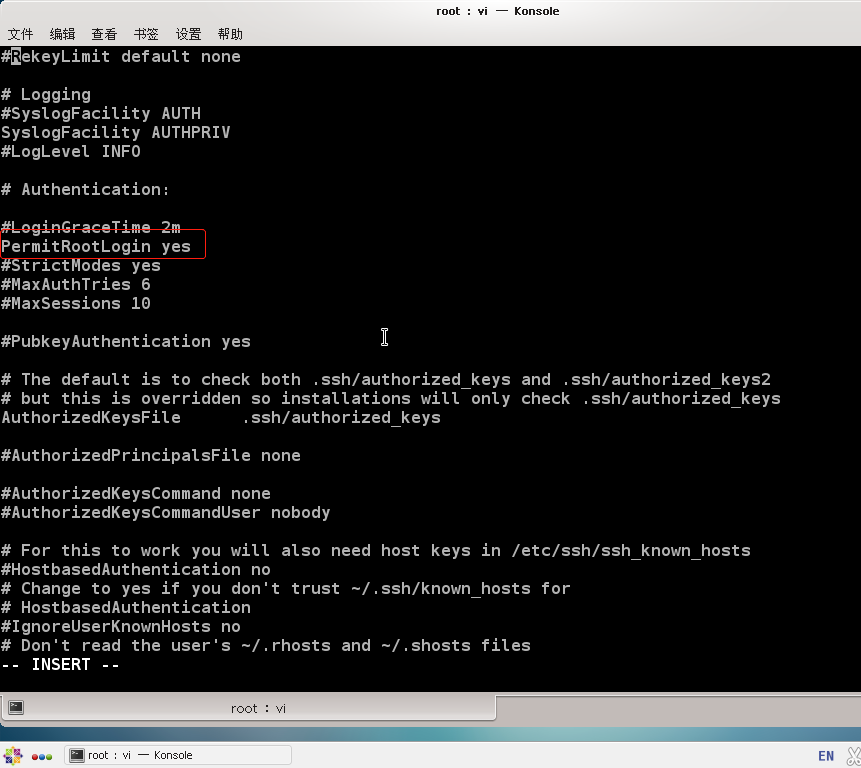
$ sudo yum -y install openssh-server

$ sudo yum -y install openssh-clients

修改SSH配置文件以下选项，去掉#注释，将四个选项启用：

$ vi /etc/ssh/sshd\_config

PermitRootLogin yes #root能使用ssh登录



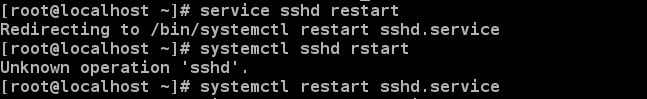
重启ssh服务，并设置开机启动：

$ service sshd restart

$ chkconfig sshd on

新版本的命令用下面的

$ systemctl restart sshd.service



安装完了之后可以使用xshell连接到CentOS并可以使用docker命令

**CentOS下Git的安装**

yum **install** git

## CentOS 7.0下Docker的安装

1. 查看内核版本(Docker需要64位版本，同时内核版本在3.10以上，如果版本低于3.10，需要升级内核)：
2. **uname** -r
3. 更新yum包：
4. **yum** update
5. 添加yum仓库：
6. sudo tee /etc/yum.repos.d/docker.repo **<<-**'EOF'
7. [dockerrepo]
8. name=Docker Repository
9. baseurl=https:*//yum.dockerproject.org/repo/main/centos/7/*
10. enabled=1
11. gpgcheck=1
12. gpgkey=https:*//yum.dockerproject.org/gpg*
13. EOF
14. 安装Docker
15. yum **install** docker-**engine**
16. 启动Docker
17. **service** docker start
18. 使用Docker国内镜像（为Docker镜像下载提速，非必须）
19. curl -sSL https://get.daocloud.io/daotools/set\_mirror.sh | sh -s http://fe8a7d6e.m.daocloud.io

**CentOS 6.5下Docker的安装**

Docker容器最早受到RHEL完善的支持是从最近的CentOS 7.0开始的，官方说明是只能运行于64位架构平台，内核版本为2.6.32-431及以上（即 >= CentOS 6.5，运行docker时实际提示3.10.0及以上）。 需要注意的是CentOS 6.5与7.0的安装是有一点点不同的，CentOS 6.x上Docker的安装包叫docker-io，并且来源于Fedora epel库，这个仓库维护了大量的没有包含在发行版中的软件，所以先要安装EPEL，而CentOS 7.x的Docker直接包含在官方镜像源的Extras仓库（CentOS-Base.repo下的[extras]节enable=1启用）。

下面就CentOS 6.5讲解Docker的安装过程，以下是软件版本：

| **Linux版本** | **Docker版本** |
| --- | --- |
| CentOS 6.5 X64（只能X64） | 1.7.1 |

**升级内核**

查看内核版本：

**uname** -r

结果：2.6.32-431.el6.x86\_64，不满足上文的需求，故此需要升级内核。

升级步骤：

1. 导入公钥数字证书
2. rpm --**import** https:*//www.elrepo.org/RPM-GPG-KEY-elrepo.org*
3. 安装ELRepo
4. rpm -ivh http:*//www.elrepo.org/elrepo-release-6-5.el6.elrepo.noarch.rpm*
5. 安装kernel长期版本
6. yum --enablerepo**=**elrepo-kernel install kernel-lt -y *# lt表示long-term的意思，长期维护版本，也可以将kernel-lt改为kernel-ml，安装主线版本*
7. 编辑grub.conf文件，修改Grub引导顺序，确认刚安装好的内核在哪个位置，然后设置default值（从0开始），一般新安装的内核在第一个位置，所以设置default=0。
8. vim /etc/grub.conf
9. *# 以下是/etc/grub.conf的内容*
10. default**=**0 *# 修改该值即可*
11. timeout**=**5
12. splashimage**=(**hd0,0**)**/grub/splash.xpm.gz
13. hiddenmenu
14. title CentOS **(**3.10.103-1.el6.elrepo.x86\_64**)**
15. 重启并查看内核版本，将会发现内核已经更新。

**安装Docker**

1. 禁用selinux，因为selinux和LXC有冲突，故而需要禁用
2. vim /etc/selinux/config的内容
3. **# 以下是/etc/selinux/config的内容**
4. **# enforcing - SELinux security policy is enforced.**
5. **# permissive - SELinux prints warnings instead of enforcing.**
6. **# disabled - No SELinux policy is loaded.**
7. SELINUX=disabled # 将SELINUX设为disabled，注意修改后最好重启下机器。
8. 安装 Fedora EPEL
9. yum -y install http://dl.fedoraproject.org/pub/epel/6/x86\_64/epel-release-6-8.noarch.rpm
10. 安装Docker
11. yum **install** -y docker-io
12. 以守护模式运行Docker
13. docker -d
14. 如果不报错，那就是启动成功了，如果报以下异常：
15. docker: relocation **error**: docker: symbol dm\_task\_get\_info\_with\_deferred\_remove, version Base **not** defined **in file** libdevmapper.so.1.02 **with** link time **reference**
16. INFO[0000] Listening **for** HTTP **on** unix **(**/var/run/docker.sock**)**

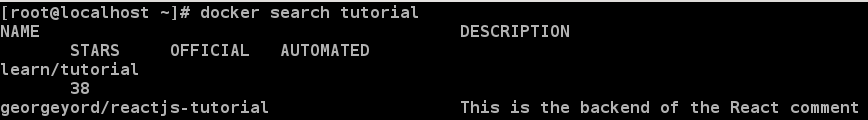
执行以下内容：

**yum** upgrade device-mapper-libs

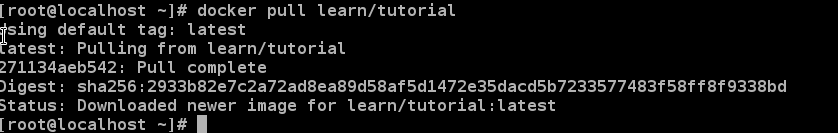
1. 将Docker开机启动
2. chkconfig docker **on**
3. 重启机器

**测试Docker安装**

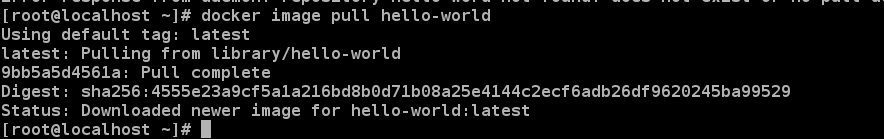
1．使用 docker search tutorial 命令搜索测试镜像learn/tutorial



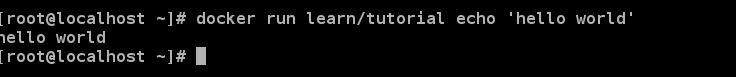
2.下使用镜像docker pull learn/tutorial



也可以拉取hell-world来做测试

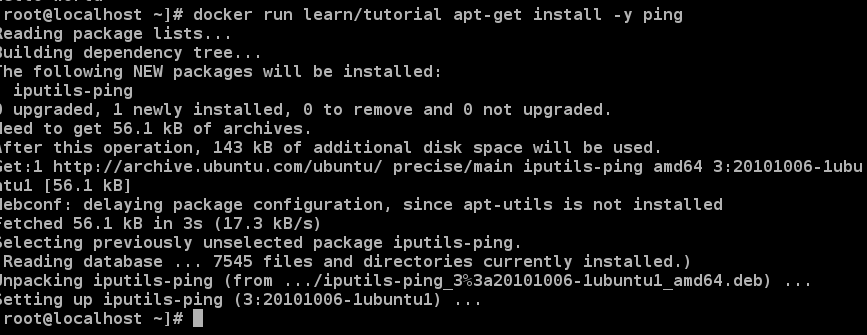


3 使用docker run learn/ tutorial echo ‘hello world’ 运行测试

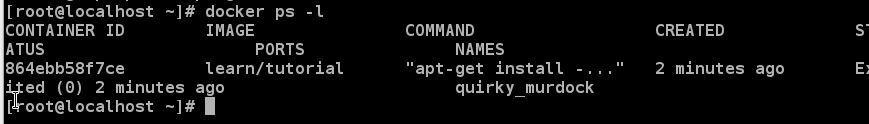


４在容器中安装ping命令

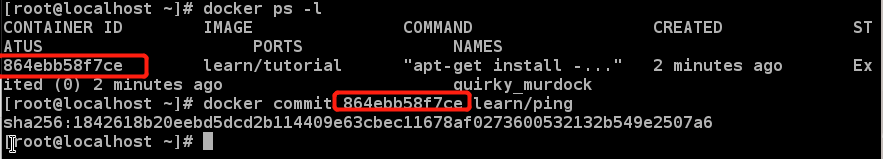
docker run learn/tutorial apt-get install -y ping



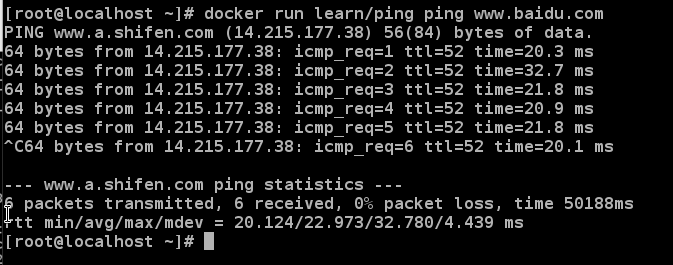
5 查看所有正处于运行中的Docker容器的进程信息



6当容器中的文件产生变化后，可以提交生成一个新的image因为生成之后的容器是不能修改的

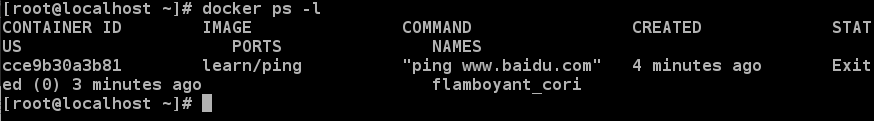


７运行新的容器，并去ping [www.baidu.com](http://www.baidu.com)

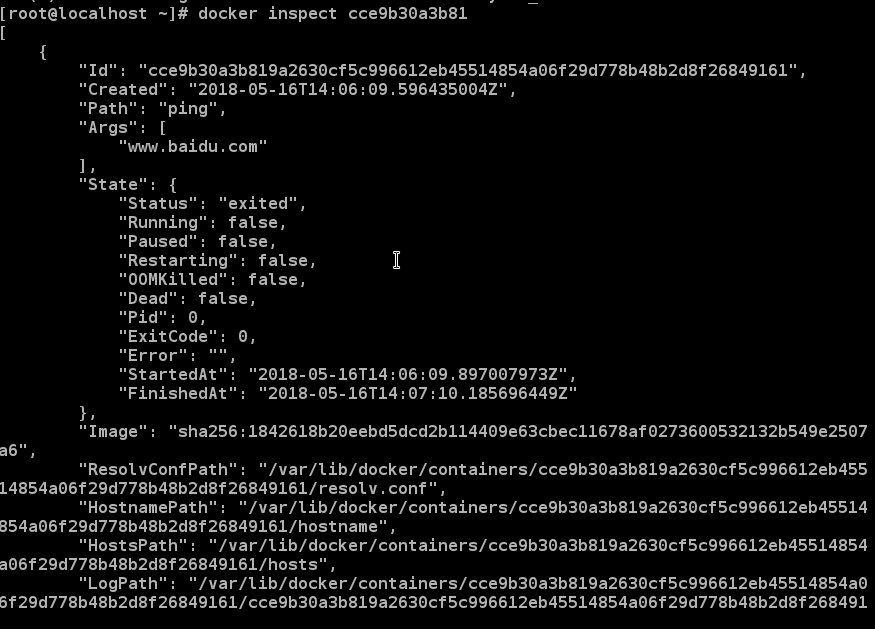


8 查看运行中的容器信息

先查${id号}



使用 docker inspect ${id号} 来查看信息



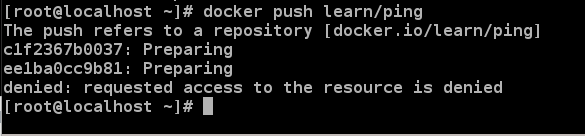
9查看本机有的docker镜像

命令：docker images



10．把镜像保存到docker hub上

命令：docker push learn/ping



**容器的基本操作**

运行一个新的容器

docker run [OPTIONS] IMAGE [COMMAND][ARG...]

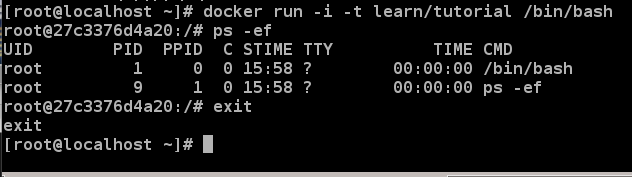
启动交互式容器

容器启动时 运行bash命令,可以进入容器并可以进行命令交互

docker run -i -t IMAGE /bin/bash

-i –interactive=true|false 默认是false,让docker守护进程为容器始终都打开标准输入

-t—tty=true|false 默认是false,要为创建的容器分配一个虚拟终端



docker run -it learn/tutorial /bin/bash

查看Docker容器

docker ps [-a] [-l]

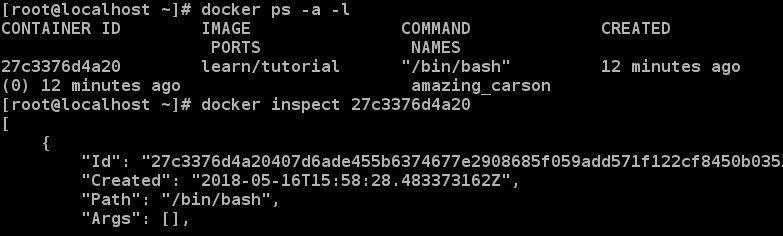
-a：列表所有容器；-f：过滤；-q 只列表容器的id；-l：列出最新创建的容器

当不给参数时，查看当前正在运行中的Docker容器



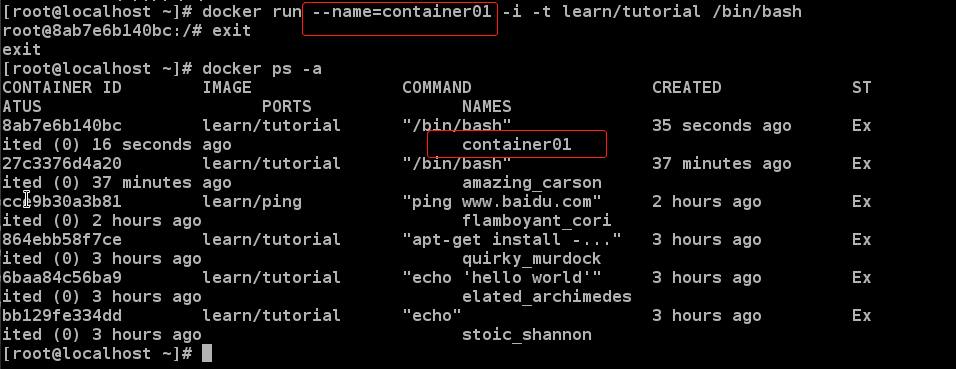
查看容器信息

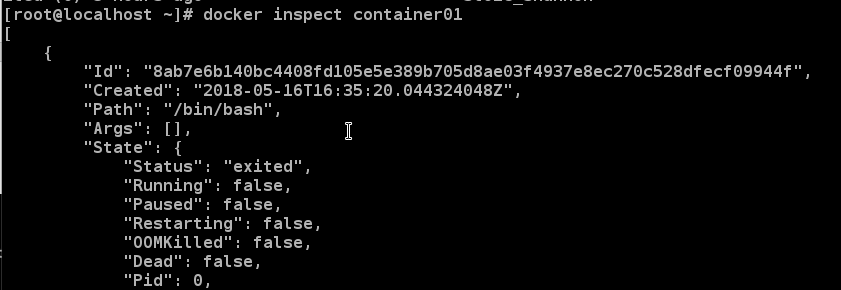
docker inspect



自定义容器名

docker run –name=自定义名 -i -t IMAGE /bin/bash

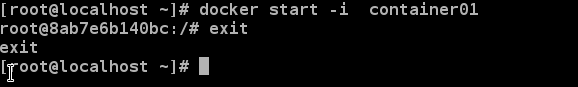




重新启动停止的容器

docker start [-i] 容器名

docker start / stop / restart 容器id



删除停止的容器

docker rm 容器名/容器id

注意：默认只有处于exited状态的Docker容器才能被删除，处于 running/paused/restarting状态的容器是不允许被删除的，否则会被

提示如下异常信息：

Error response from daemon: Conflict, You cannot remove a running container.

Stop the container before attempting removal or use -f

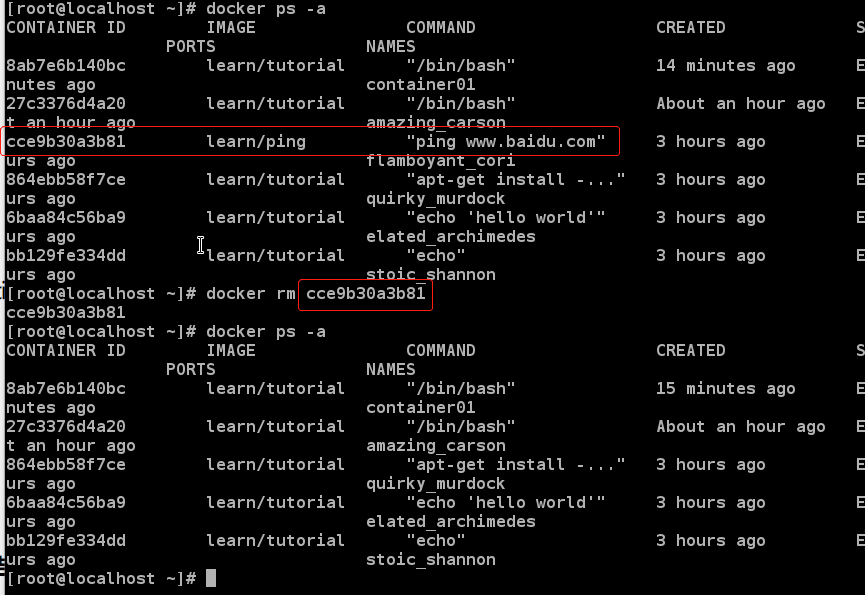
FATA[0000] Error: failed to remove one or more containers

如果你想要无视Docker容器的当前状态，直接强制删除，那么此时你需要添加-f参数

docker rm -f 等价于 docker kill

为了避免手工删除Docker容器的工作负担，你可以在创建Docker容器时指定--rm参数，告诉Docker，当当前Docker容器的状态处于

exited，就自动将其删除，无需人工干预，这在你做学习测试时会很有用。



以守护形式运行容器

方法1

容器启动时 运行bash命令,可以进入容器并可以进行命令交互

docker run -i -t IMAGE /bin/bash

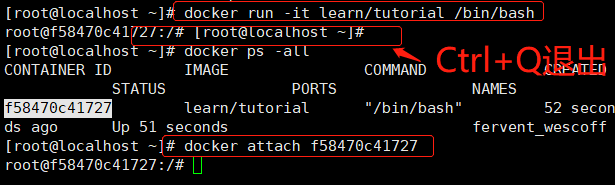
docker run -I -t learn/tutorial /bin/bash

再使用 Ctrl+P && Ctrl+Q(两个快捷都要按先Ctrl+P再Ctrl+Q)来退出交互，但容器会在后台运行，一般用来运行服务和应用。

，如果想要再次进入容器，可以使用

docker attach 容器名/容器id

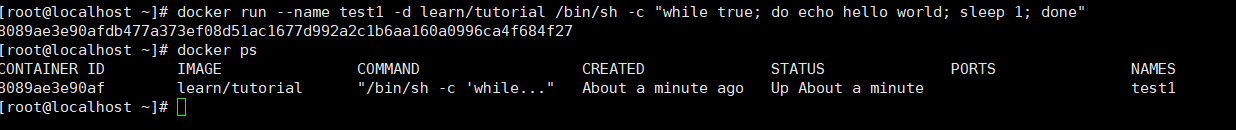
重新进入容器的命令窗口



方法2

docker run -d 镜像名 [COMMAND][ARG…]

docker run --name test1 -d learn/tutorial /bin/sh -c "while true; do echo hello world; sleep 1; done"



查看容器的运行日志

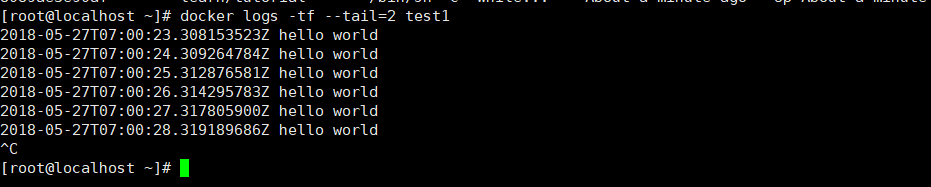
docker logs [-f][-t][--tail] 容器名

-f --follows=true|false 默认是false, 参数表示实时显示容器的日志信息

-t --timestamps=true|false 默认是false,在显示容器的日志信息加上时间戳

-tail =”all”, 返回结尾数量的日志如，--tail=2查看末尾2行的容器日志信息。

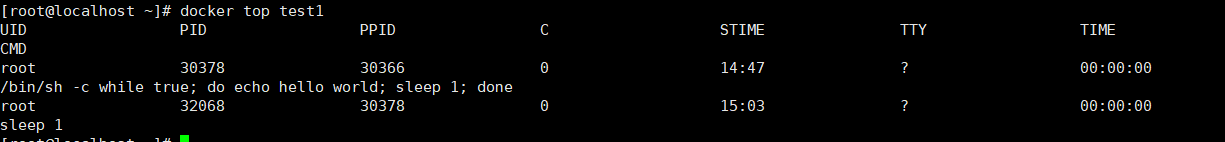
docker logs -tf --tail=2 test1



查看容器内的进程

docker top 容器名

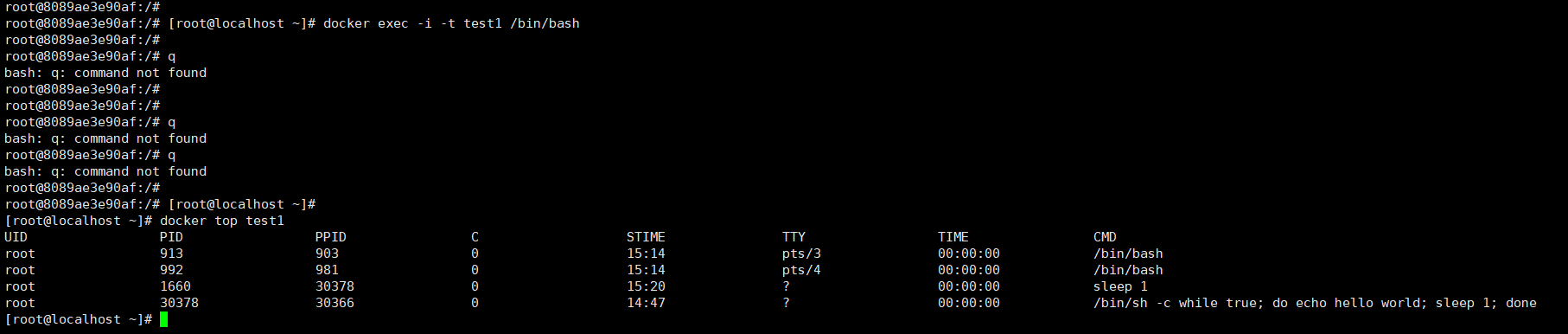
docker top test1



在运行中的容器内启动新进程

docker exec [-d][-i][-t] 容器名 [COMMAND][ARG...]

docker exec -i -t test1 /bin/bash



停止守护式容器

docker stop 容器名

发送命令等待停止

docker kill 容器名

直接停止容器

设置容器的端口映射

docker run [-P][-p] IMAGE

-P –publish-all=true|false 默认是false, 参数表示为容器暴露所有的端口进行映射。

docker run -P -I -t learn/tutorial /bin/bash

-p -- publish =[] 参数表示为容器暴露指定的端口进行映射。

docker run -p 8080:80 -I -t learn/tutorial /bin/bash

指定宿机8080端品对应容器80端口

docker run -p 0.0.0.0:80 -I -t learn/tutorial /bin/bash

指定宿机IP对应容器80端口

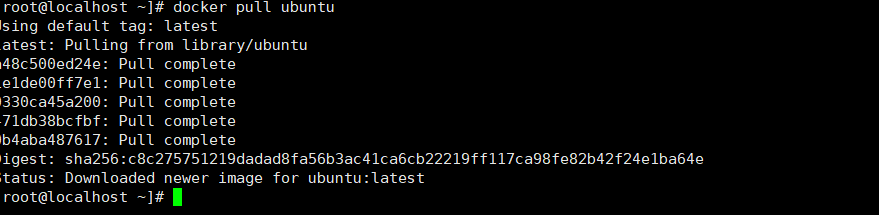
docker run -p 0.0.0.0:8080:80 -I -t learn/tutorial /bin/bash

指定宿机IP与端口对应容器80端口

Nginx部署流程

1. 下一个基础镜像

docker pull ubuntu



1. 创建映射80端口的交互式容器

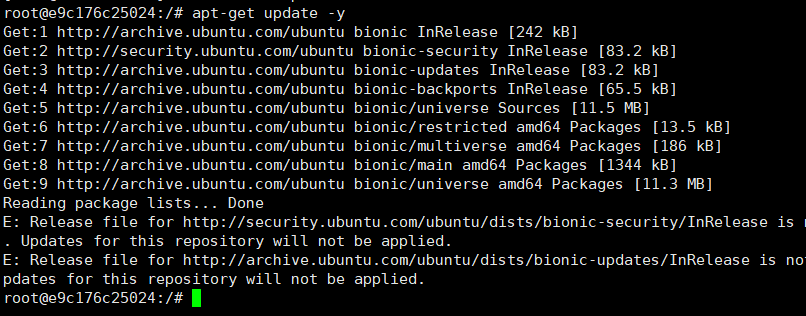
docker run -p 80 --name web -i -t ubuntu /bin/bash



可能会出现错误为docker: Error response from daemon: driver failed programming external connectivity on endpoint，[参考](#driverfailedprogramming)

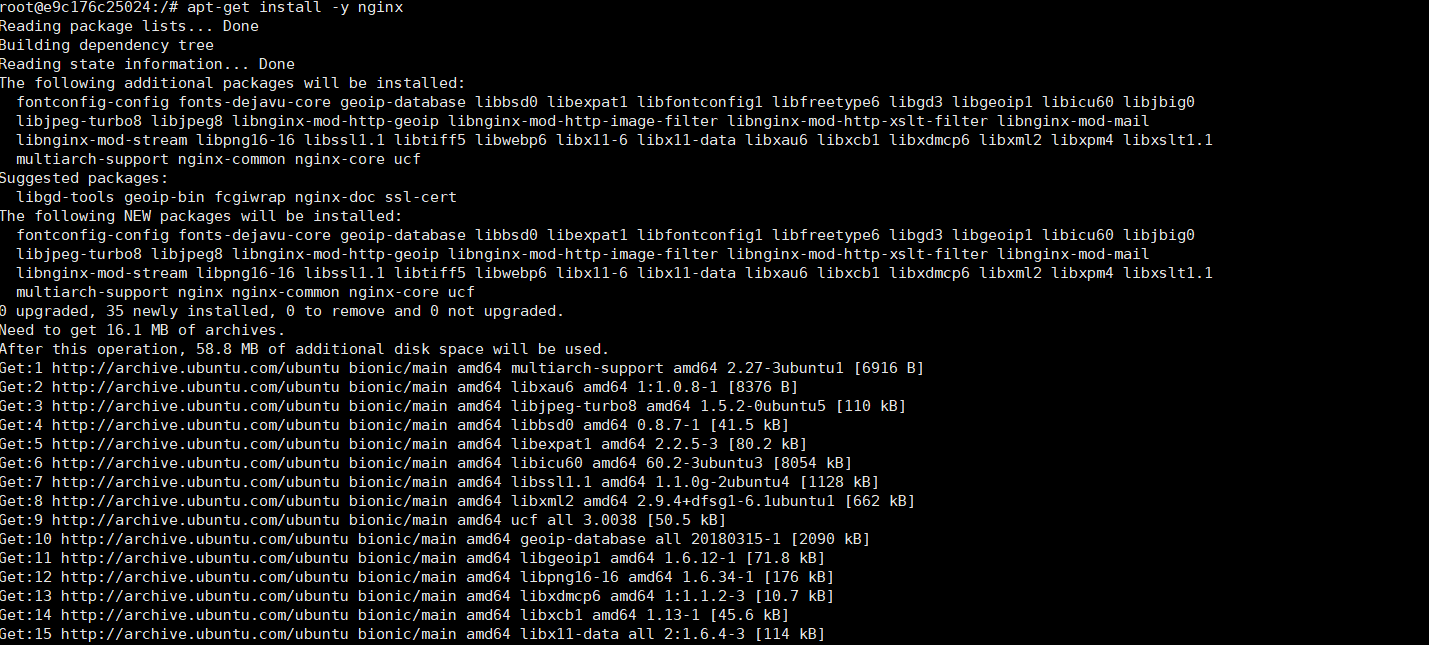
1. 在容器中更新一下源

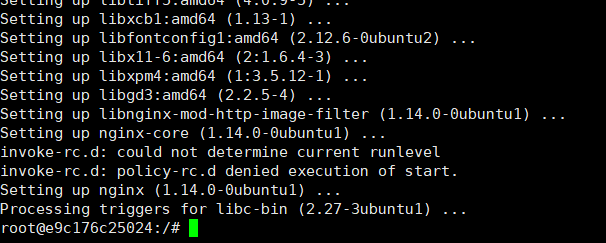
apt-get update -y



1. 在容器中更新一下源安装Nginx

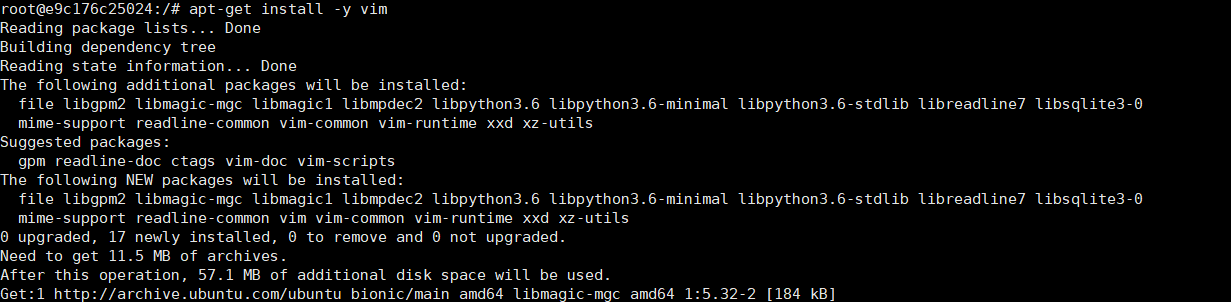
apt-get install -y nginx





1. 安装文本编辑器vim

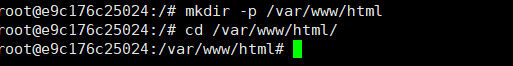
apt-get install -y vim



1. 创建静态页面
   * 1. 创建一个目录,并时入目录

mkdir -p /var/www/html

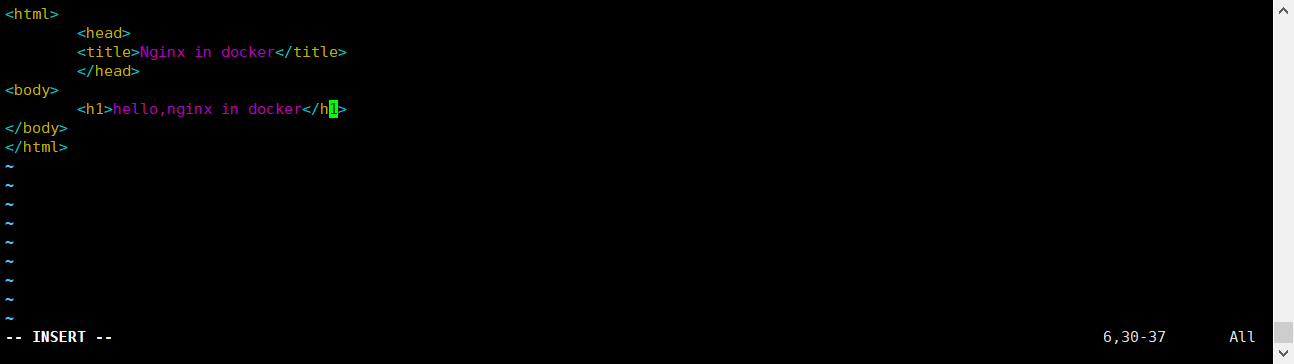
cd /var/www/html/



* + 1. 使用vim来创建一个页面

vim index.html





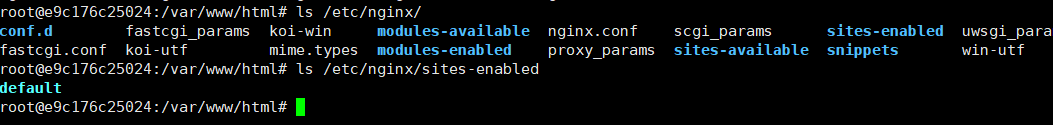
1. 修改Nginx配置文件
2. 查找Nginx安装的位置

whereis nginx

1. 查看Nginx配置文件位置

ls /etc/nginx/

ls /etc/nginx/sites-enabled

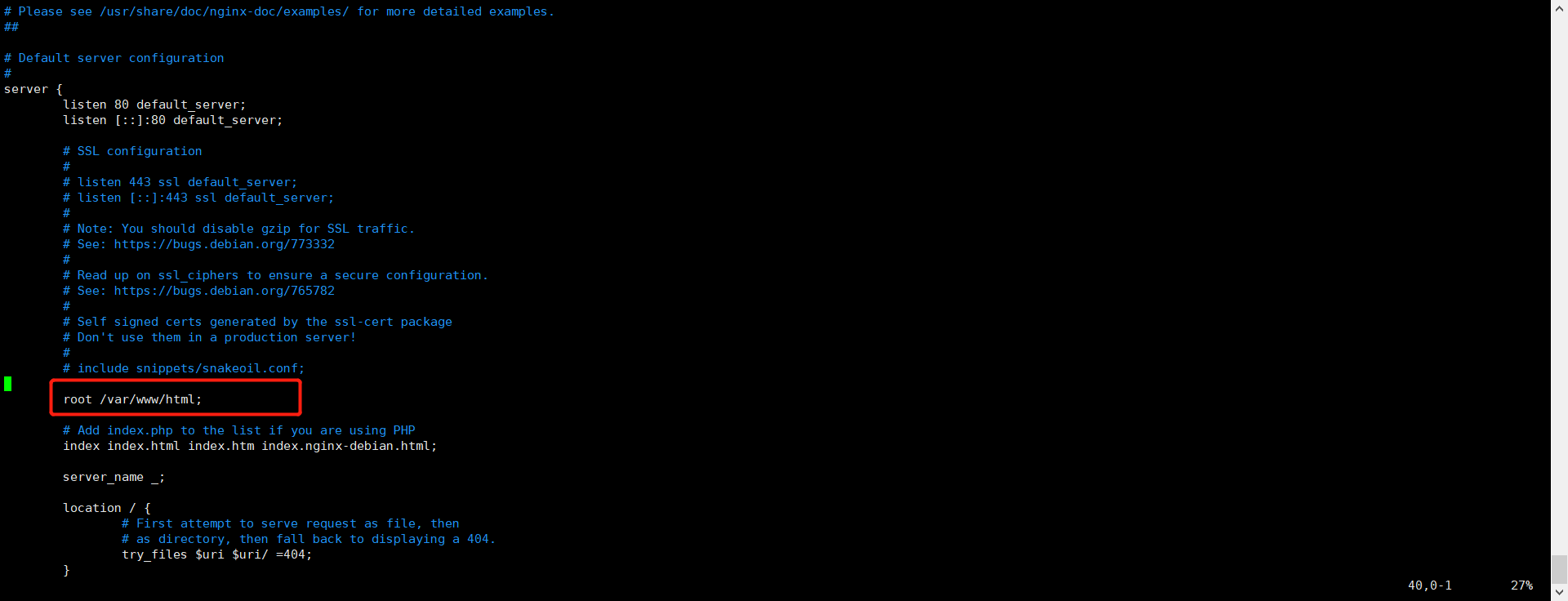


vim /etc/nginx/sites-enabled/default

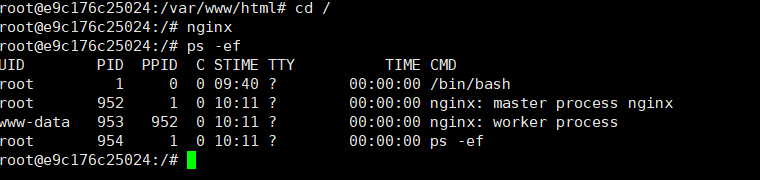
1. 打开配置文件，并修改root值为上面刚刚创建的文件（/var/www/html）这个目录

vim /etc/nginx/sites-enabled/default



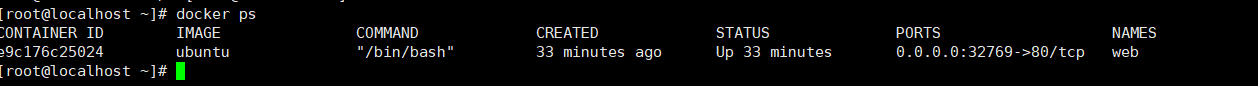


1. 运行Nginx



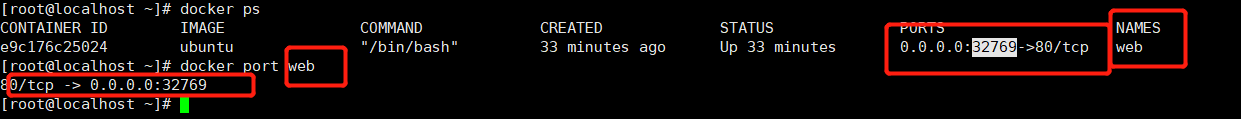
再使用 Ctrl+P && Ctrl+Q(两个快捷都要按先Ctrl+P再Ctrl+Q)来退出交互

查看容器是否在后台运行



发现32769端口映射到容器的80端口

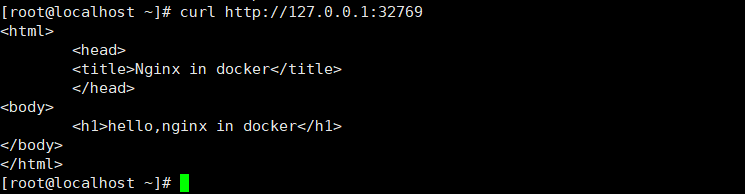
也可以使用命令docker port 容器名 来查看，如：docker port web



也可以使用docker top容器名 来查看进程的状态

1. 验证网站访问

curl <http://127.0.0.1:32769>



查看和删除镜像

列出镜像

docker images [OPTSIONS] [REPOSITORY]

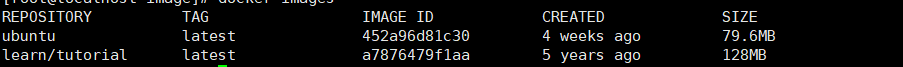
-a –all=false 默认是false, 显示所有图像（默认过滤出用于构建的中间图像）

-f —fiter=[] 显示的过滤条件

--no-trunc =false , 不截断输出IMAGE\_ID

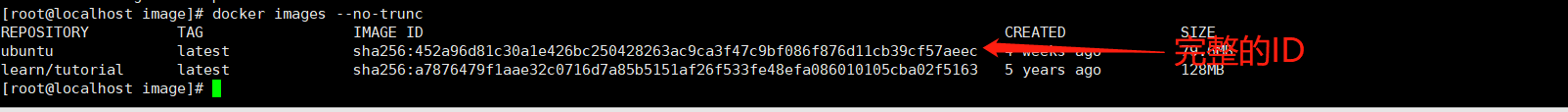
-tree = false：以树格式输出图形

-q,--quiet=false 只显示IMAGE\_ID这列

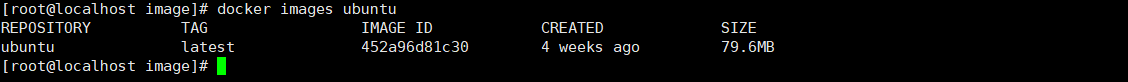


第一列：仓库名， 第二列：镜像的标签名，第三列镜像的唯一ID，第四列镜像的创建时间，第五列：镜像的大小

docker images --no-trunc



docker images ubuntu



查看镜像

即支持容器的查看也支持镜像的查看，可以使用仓库名: 镜像的标签名查看也可以用镜像的唯一ID来查看。

docker inspect [OPTSIONS] CONTAINER|IMAGE [CONTAINER|IMAGE..]

-f,--format=””,

删除镜像

docker rmi [OPTSIONS] IMAGE [IMAGE..]

-f,--force=false, 强制删除图像

--no-prue=false, 不要删除未标记的父项

获取和推送镜像

查找镜像

Docker Hub

https://registry.hub.docker.com

docker search [OPTSIONS] IMAGE TERM

--automated=false ，只显示自动化构建的镜像

--no-trunc=false，不要截断输出

-s,--stars=0，指定星级以上的结果集

最多返回25个结果

3星级以上的ubuntu仓库命令： docker search -s 3 ubuntu

拉取镜像

docker pull [OPTSIONS] NAME [:TAG]

-a,--all-tags=false ，下载存储库中的所有标记图像

使用—registry-mirror选项，来使用国内镜像，可加速下载

1. 修改：vim /etc/default/docker 修改docker配置
2. 添加一行:DOCKER\_OPTS=”—registry-mirror=加速器给的连接”

进入<https://www.daocloud.io>网站注册之后，生成加速器，会返回一个连接放到第二步中去

1. 重启docker进程

Service docker restart

推送镜像

把自己生成的镜像上传到docker hub中去

docker push NAME [:TAG]

镜像的基本操作

构建镜像

保存对容器的修改，并再次使用

自定义镜像的能力

以软件的形式打包并分发服务及运行环境

布署微服务

环境准备

下面的操作都在centOS7中操作

阿里镜像仓库：https://cr.console.aliyun.com

阿里git仓库：<https://code.aliyun.com/>

[Maven安装](#maven安装)

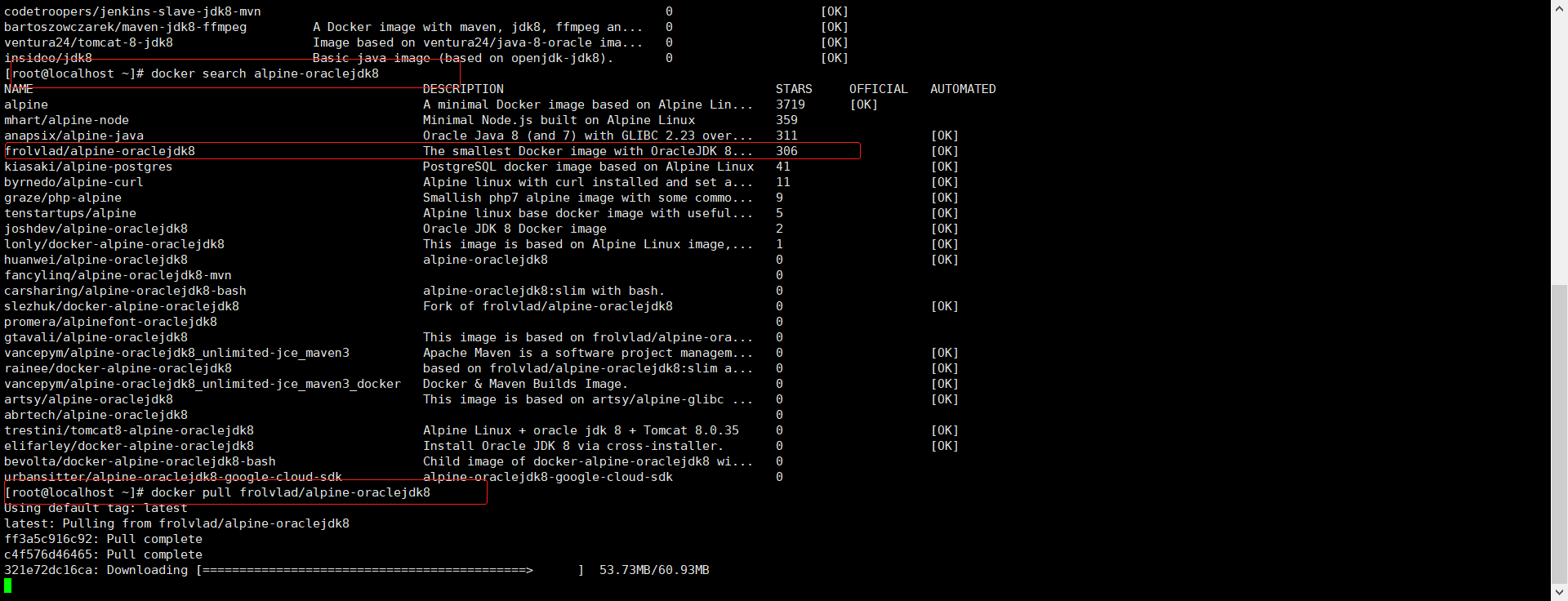
[Docker安装](#_CentOS_7.0下Docker的安装)

[docker-compose安装](#dockercompose安装)

下载JDK8镜像

docker search alpine-oraclejdk8

docker pull frolvlad/alpine-oraclejdk8



docker search java:8

docker pull java:8

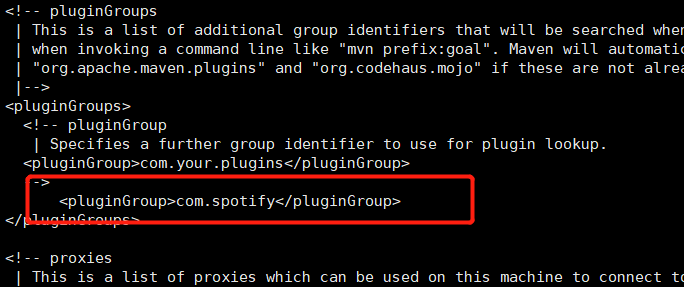


docker run -p 8761:8761 -d springcloud/eureka-server

docker rmi -f springcloud/eureka-server



修改maven配置文件



<pluginGroups>

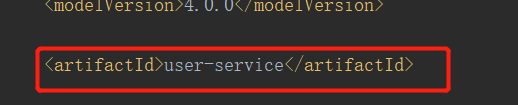
<pluginGroup>com.spotify</pluginGroup>

</pluginGroups>

在每个打包并运行的项目中的pom.xml增加docker插件

<build>  
 <plugins>  
 <!-- 添加spring-boot的maven插件 -->  
 <plugin>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-maven-plugin</artifactId>  
 </plugin>  
 <!-- 添加docker的maven插件 -->  
 <plugin>  
 <groupId>com.spotify</groupId>  
 <artifactId>docker-maven-plugin</artifactId>  
 <version>${docker.plugin.version}</version>  
 <executions>  
 <execution>  
 <phase>package</phase>  
 <goals>  
 <goal>build</goal>  
 </goals>  
 </execution>  
 </executions>  
 <configuration>  
 <imageName>${docker.image.prefix}/${project.artifactId}</imageName>  
 <dockerDirectory>${project.basedir}/src/main/docker</dockerDirectory>  
 <buildArgs>  
 <JAR\_FILE>${project.build.finalName}.jar</JAR\_FILE>  
 </buildArgs>  
 <resources>  
 <resource>  
 <targetPath>/</targetPath>  
 <directory>${project.build.directory}</directory>  
 <include>${project.build.finalName}.jar</include>  
 </resource>  
 </resources>  
 </configuration>  
 </plugin>  
 <plugin>  
 <artifactId>maven-dependency-plugin</artifactId>  
 <executions>  
 <execution>  
 <id>copy-dependencies</id>  
 <phase>prepare-package</phase>  
 <goals>  
 <goal>copy-dependencies</goal>  
 </goals>  
 </execution>  
 </executions>  
 <configuration>  
 <includeTypes>jar</includeTypes>  
 <overWriteSnapshots>true</overWriteSnapshots>  
 <type>jar</type>  
 <outputDirectory>${project.build.directory}/lib</outputDirectory>  
 </configuration>  
 </plugin>  
 </plugins>  
</build>

修改所有要打包镜像pom.xml中的artifactId的值

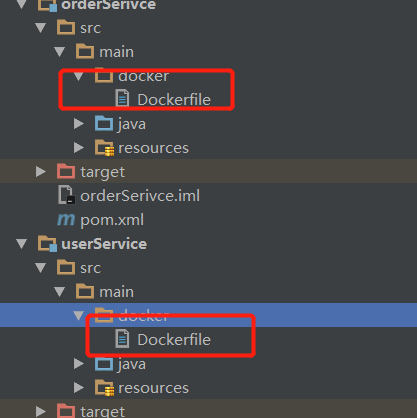


所有的值都要小写，因为这会做为docker的镜像文件名，不能大写

为每个项目application.yml配置增加**spring.profiles.active=**docker配置

增加docker配置，与dev配置区分开来

每个项目增加Dockerfile文件

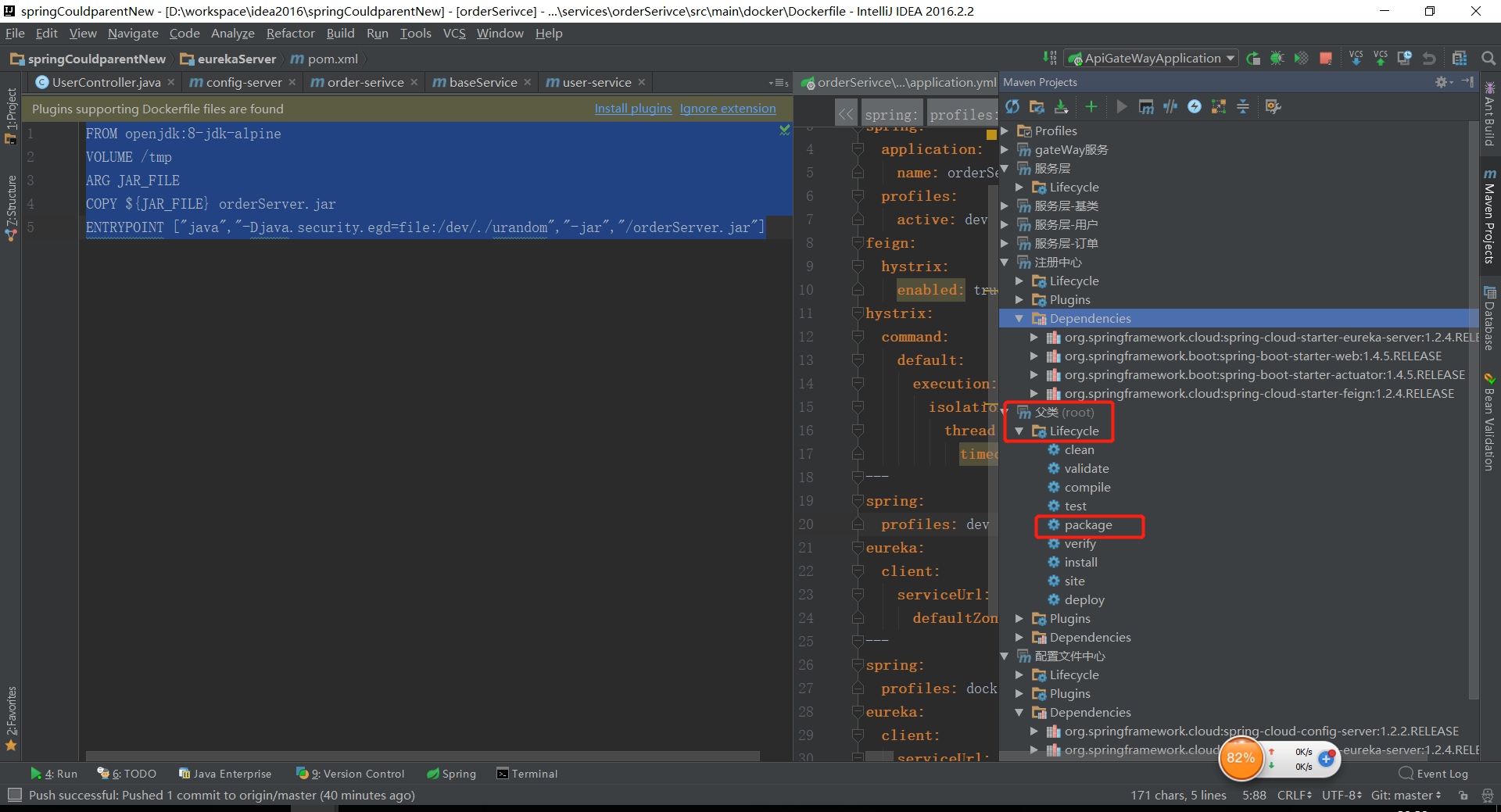


增加Dockerfile文件，内容如下

FROM openjdk:8-jdk-alpine  
VOLUME /tmp  
ARG JAR\_FILE  
COPY ${JAR\_FILE} orderServer.jar  
ENTRYPOINT ["java","-Djava.security.egd=file:/dev/./urandom","-jar","/orderServer.jar"]

orderServer.jar名字做下修改，对应每个项目名

生成镜像



在idea中执行下package 就会生成所有镜像，

也可以在对应的项目下有pom.xml文件执行

mvn package docker:build

来单独打成镜像

推送镜像到阿里

先设置阿里镜像加速器，进入<https://cr.console.aliyun.com>， 有对应的教程

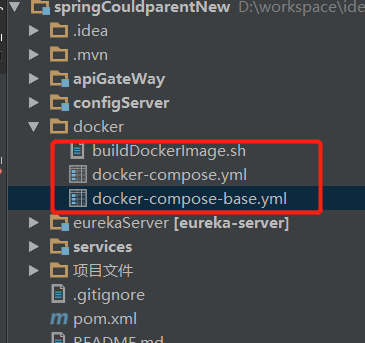
docker login 你自己阿里的镜像仓库，

docker push 镜像名

安装docker-compose

<https://docs.docker.com/compose/install/#alternative-install-options>

编写docker-compose并启动项目



在shell中到这个目录下执行下面命令

docker-compose -f docker-compose-base.yml up

docker-compose -f docker-compose.yml up

用来启用所有服务

docker-compose常用命令

mvn package docker:build

chmod -v a+wx docker-compose-base.yml

docker-compose -f docker-compose-base.yml up

docker-compose -f docker-compose-base.yml stop

docker-compose -f docker-compose-base.yml rm

docker-compose -f docker-compose.yml up

docker-compose -f docker-compose.yml stop

docker-compose -f docker-compose.yml rm

docker rmi -f springcloud/eureka-server

docker rmi $(docker images | grep "none" | awk '{print $3}')

docker rmi $(docker images | grep "springcloud\*" | awk '{print $3}')

出现的问题

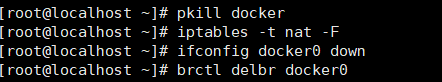
**driver failed programming external connectivity on endpoint wordpress**

[root@docker ~]# docker run -itd --name wordpress -p 88:80 wordpress:v1  
b77482f8075042e9cc6723d6922a1211c37d99339678a00cc040396b23d40ef0  
docker: Error response from daemon: driver failed programming external connectivity on endpoint wordpress (77cb6b1ea5387ac97b1b90178b2ccda831aa9713e0e9a83be057083fed66fc69):  (iptables failed: iptables --wait -t nat -A DOCKER -p tcp -d 0/0 --dport 88 -j DNAT --to-destination 172.17.0.2:80 ! -i docker0: iptables: No chain/target/match by that name.

 (exit status 1)).

**解决办法 => 重建docker0网络恢复**

pkill docker                         #终止进程  
iptables -t nat -F                 #清空nat表的所有链  
ifconfig docker0 down        #停止docker默认网桥  
brctl delbr docker0             #删除网桥  
service docker restart       #重启docker





我为什么会出现这样的错误呢 ，因为我在docker运行的时候，执行了systemctl stop firewalld，关闭防火墙可能会清空docker规则。

[apt-get时候出现Package ssh is not available, but is referred to by another package 错误](https://www.cnblogs.com/cyttina/archive/2013/01/29/2882111.html)

这个问题的原因是ubuntu的/etc/apt/source.list中的源比较旧了，需要更新一下，更新方法：

$ sudo apt-get -y update

更新完毕之后，在使用apt-get就没有问题了

使用mvn package docker:build时，No plugin found for prefix 'docker' in the current project

in maven ：settings.xml  
add

<settings>

<pluginGroups>

<pluginGroup>com.spotify</pluginGroup>

</pluginGroups>

</settings>