### IMCloud EP工厂搭建部署文档

根所服务器环境分别采用两种搭建部署方式：

1.单服务器环境：采用Docker + Docker Compose 进行搭建

2.多服务器（集群）环境：采用 Docker + Docker Swarm 进行搭建

系统环境准备：

1. 操作系统：centos7.4（当前最新版本，释出版本号1708）

# cat /etc/centos-release //查看发行版本号

CentOS Linux release 7.4.1708 (Core)



1. 修改主机名（以实际环境名称为准）

# hostnamectl set-hostname --static imcloud.test.com

# hostname

Imcloud.test.com



1. 关闭selinux服务

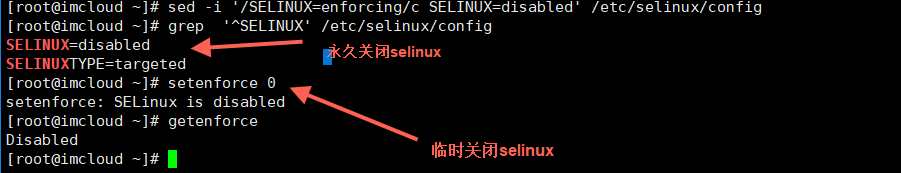
# sed -i '/SELINUX=enforcing/c SELINUX=disabled' /etc/selinux/config

# grep '^SELINUX' /etc/selinux/config

SELINUX=disabled

SELINUXTYPE=targeted

# setenforce 0



1. 关闭防火墙服务

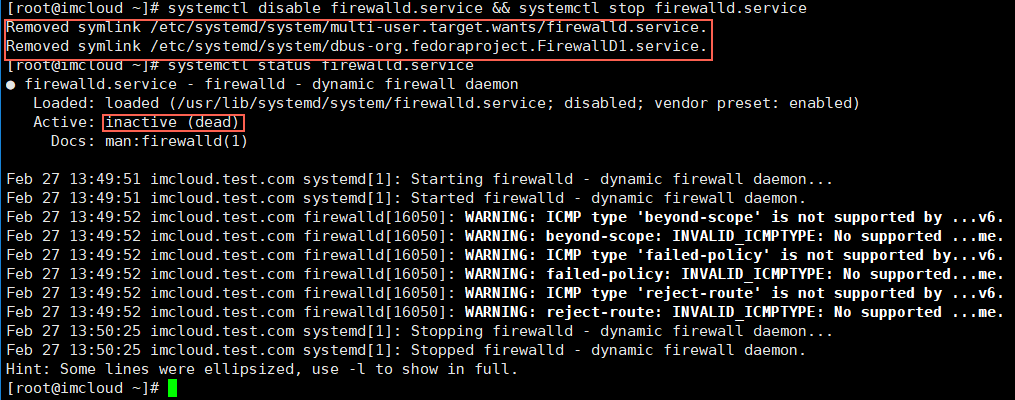
[root@imcloud ~]# systemctl start firewalld.service

[root@imcloud ~]# systemctl disable firewalld.service && systemctl stop firewalld.service

Removed symlink /etc/systemd/system/multi-user.target.wants/firewalld.service.

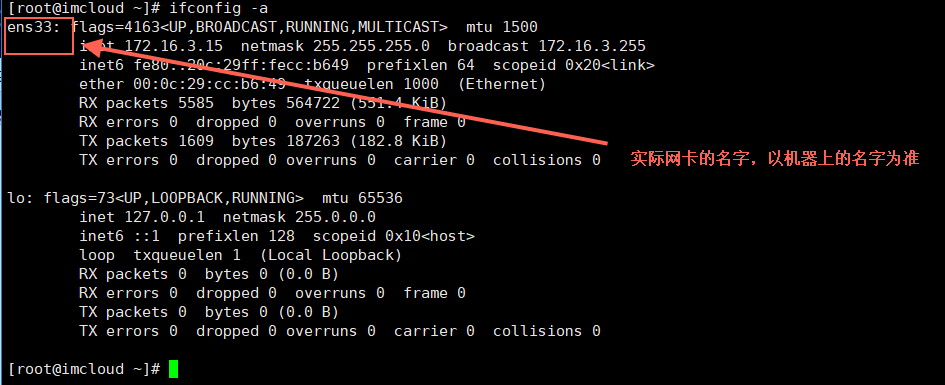
Removed symlink /etc/systemd/system/dbus-org.fedoraproject.FirewallD1.service.

[root@imcloud ~]# systemctl status firewalld.service



1. 设置网络接口IP地址（服务器设置为固定地址，IP地址以实际环境为准）

[root@imcloud ~]# ifconfig -a //查看服务器所有的网络接口，找到实际网卡的接口名



[root@imcloud ~]# vim /etc/sysconfig/network-scripts/ifcfg-ens33 //编辑网卡配置文件

//配置如下：

TYPE=Ethernet

PROXY\_METHOD=none

BROWSER\_ONLY=no

BOOTPROTO=static //引导方式

DEFROUTE=yes

IPV4\_FAILURE\_FATAL=no

IPV6INIT=no

IPV6\_AUTOCONF=yes

IPV6\_DEFROUTE=yes

IPV6\_FAILURE\_FATAL=no

IPV6\_ADDR\_GEN\_MODE=stable-privacy

NAME=ens33

UUID=5f2ed8ca-cd25-403b-a0fb-25acf1344319

DEVICE=ens33 //设备名

ONBOOT=yes //开机启动

IPADDR=172.16.3.15 //静态指定的IP地址

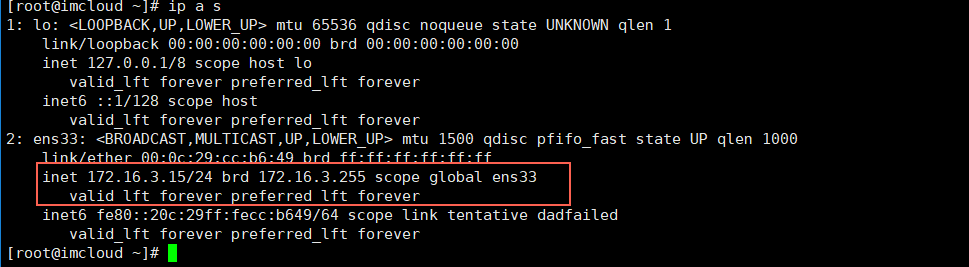
NETMASK=255.255.255.0 //子网掩码

GATEWAY=172.16.3.1 //默认网关

DNS1=114.114.114.144 //DNS服务器地址

[root@imcloud ~]# systemctl restart network //重启网络服务

[root@imcloud ~]# ip a s //验证IP是否修改成功



1. 设置yum源，使用阿里云开源镜像站点

[root@imcloud ~]# cd /etc/yum.repos.d/

[root@imcloud yum.repos.d]# for i in `ls` ; do mv $i $i.bak ; done //备份官方源

[root@imcloud yum.repos.d]# curl -o /etc/yum.repos.d/CentOS-Base.repo \

http://mirrors.aliyun.com/repo/Centos-7.repo //下载阿里云源

[root@imcloud yum.repos.d]# ls

CentOS-Base.repo CentOS-CR.repo.bak CentOS-fasttrack.repo.bak CentOS-Sources.repo.bak CentOS-Base.repo.bak CentOS-Debuginfo.repo.bak CentOS-Media.repo.bak CentOS-Vault.repo.bak

[root@imcloud yum.repos.d]# yum clean all //清除所有缓存

[root@imcloud yum.repos.d]# yum makecache fast //更新yum元数据

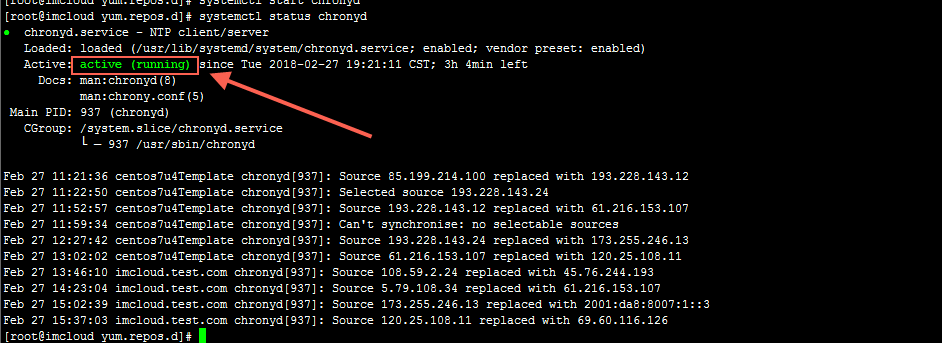
1. 设置服务器自动时间同步

[root@imcloud yum.repos.d]# yum -y install chrony //安装chrony服务

[root@imcloud yum.repos.d]# systemctl enable chronyd // 设置开机启动

[root@imcloud yum.repos.d]# systecctl start chronyd // 启动服务

[root@imcloud yum.repos.d]# systemctl status chronyd //查看服务运行状态



1. 更新、重启系统

[root@imcloud yum.repos.d]# yum -y update

[root@imcloud yum.repos.d]# reboot

安装Docker CE：

如果系统已经有Docker，则需要先删除它们：

[root@imcloud ~]# yum remove docker-common docker-selinux docker-engine

使用Docker repository

1.安装yum工具

yum install -y yum-utils device-mapper-persistent-data lvm2

2.添加Docker repo

yum-config-manager \

--add-repo \

https://download.docker.com/linux/centos/docker-ce.repo

3.更新yum缓存

[root@imcloud ~]# yum makecache fast

4.安装Docker-ce

[root@imcloud ~]# yum install docker-ce -ydoc

5.验证Docker 版本

[root@imcloud ~]# docker --version

Docker version 17.12.0-ce, build c97c6d6

6.配置Docker镜像加速器

由于国内网络限制，dockerhub上面的镜像经常会有拉取失败的问题。 我们使用的是DaoCloud加速器，也可以使用阿里云加速器，进入DaoCloud官网，注册帐号，即可在控制台找到加速器配置，复制执行即可。





[root@imcloud ~]# curl -sSL https://get.daocloud.io/daotools/set\_mirror.sh | \

sh -s http://ffce2532.m.daocloud.io

7.启动Docker

[root@imcloud ~]# systemctl enable docker.service

Created symlink from /etc/systemd/system/multi-user.target.wants/docker.service to /usr/lib/systemd/system/docker.service.

[root@imcloud ~]# systemctl start docker.service

安装Docker-compose：

1.运行下面的命令下载最新版的Docker Compose

sudo curl -L https://github.com/docker/compose/releases/download/1.19.0/docker-compose-`uname -s`-`uname -m` -o /usr/local/bin/docker-compose

2.给文件增加可执行权限

[root@imcloud ~]# chmod +x /usr/local/bin/docker-compose

3.验证安装Docker compose成功否

[root@imcloud ~]# docker-compose --version

docker-compose version 1.19.0, build 9e633ef

单服务器环境DOCKER容器部署

1.创建一个桥接网络，用于单主机Docker容器间的通信

[root@imcloud ~]# docker network create imcloudep-external //创建名为imcloudep-external 网络

71e74e9af36fa017793a4cd25c003bc67b285653a7a2495d619bd5b56c30eac1

[root@imcloud ~]# docker network ls //验证是否创建成功

NETWORK ID NAME DRIVER SCOPE

08980bf26d46 bridge bridge local

b198efa4e505 host host local

71e74e9af36f imcloudep-external bridge local

3132d89a5fbf none null local

[root@imcloud ~]#

2.部署微服务环境需要用到的基础容器镜像

需要用到mysql、rabbitmq、redis、gitlab-ce镜像,这里使用docker-compose来进行服务编排。

1.进入/opt/imcloudep目录，imcloudep目录下创建docker-compos.yml文件

[root@imcloud opt]# cd /opt/imcloudep

[root@imcloud imcloudep]# mkdir dockercompose-base/

[root@imcloud dockercompose-base]# vim docker-compose.yml

//配置如下：

version: '3' //版本号

services:

db-mysql: //服务名

image: mysql //基础镜像

volumes: //实现数据持久化

- /opt/mysql/data:/var/lib/mysql

- /opt/mysql/conf:/etc/mysql/conf.d

- /etc/localtime:/etc/localtime //保证容器时间和主机时间时区一致

environment: //设置容器的环境参数

- MYSQL\_ROOT\_PASSWORD=Hello!!!123 //此处设置的是root用户登录密码

ports: //容器端口到主机端口的映射

- "3306:3306"

command: mysqld --character-set-server=utf8 --collation-server=utf8\_unicode\_ci

//容器的启动命令，并指定mysql服务的服务端字符集和校验字符集

cache-redis:

image: redis

ports:

- "6379:6379"

volumes:

- /opt/redis/redis.conf:/usr/local/etc/redis/redis.conf

- /opt/redis/data:/data

command: redis-server /usr/local/etc/redis/redis.conf

imcloudep-rabbitmq:

image: rabbitmq:3-management

hostname: imcloudep-rabbitmq

environment:

- RABBITMQ\_DEFAULT\_USER=root //登录用户名

- RABBITMQ\_DEFAULT\_PASS=root //登录密码

volumes:

- /opt/rabbitmq:/var/lib/rabbitmq

ports:

- "15672:15672"

networks:

default:

external:

name: imcloudep-external //全局定义默认网络

:wq //保存退出

2.主机上创建项目目录及基础容器的挂载目录

[root@imcloud opt]# pwd //显示当前目录

/opt

[root@imcloud opt]# mkdir imcloudep //imcloudep表示一个测试环境目录

[root@imcloud opt]# mkdir gitlab mysql rabbitmq redis //创建四个目录

[root@imcloud opt]#

[root@imcloud opt]# mkdir gitlab/{config,data,logs}

[root@imcloud opt]# mkdir mysql/{conf,data}

[root@imcloud opt]# cd mysql/conf/

[root@imcloud opt]# vim my.cnf

[client]

default-character-set=utf8

[mysql]

default-character-set=utf8

[mysqld]

pid-file = /var/run/mysqld/mysqld.pid

socket = /var/run/mysqld/mysqld.sock

datadir = /var/lib/mysql

#log-error = /var/log/mysql/error.log

# By default we only accept connections from localhost

#bind-address = 127.0.0.1

# Disabling symbolic-links is recommended to prevent assorted security risks

symbolic-links=0

lower\_case\_table\_names=1

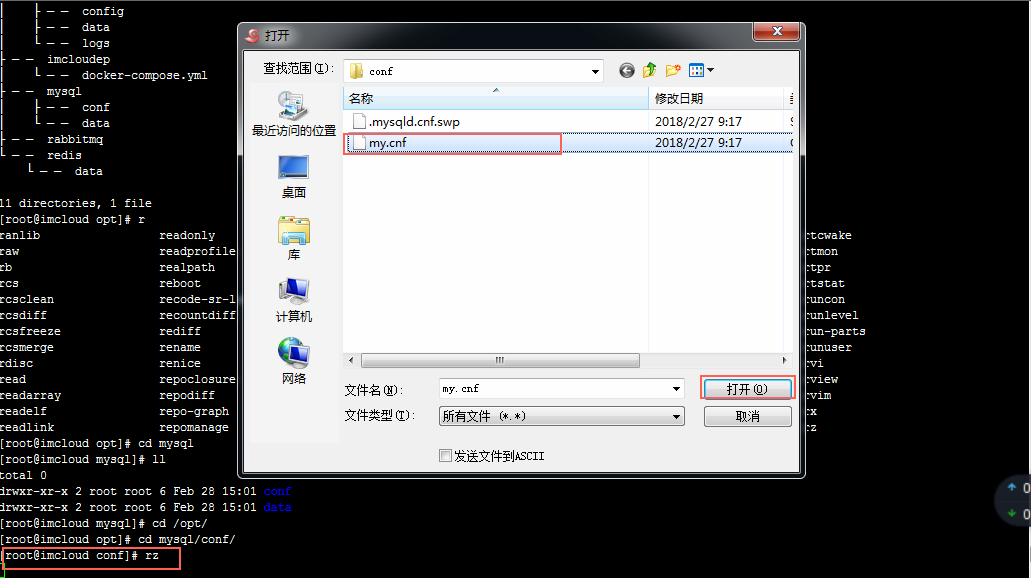
collation-server=utf8\_general\_ci

character-set-server=utf8

init-connect='SET NAMES utf8'

sql\_mode='STRICT\_TRANS\_TABLES,NO\_ZERO\_IN\_DATE,NO\_ZERO\_DATE,ERROR\_FOR\_DIVISION\_BY\_ZERO,NO\_AUTO\_CREATE\_USER,NO\_ENGINE\_SUBSTITUTION'

或者上传已编辑好的mysql配置文件到/opt/mysql/conf/目录下



[root@imcloud opt]#cd /opt/redis

[root@imcloud redis]# vim redis.conf

//配置如下(约去了注释)：

protected-mode yes

port 6379

tcp-backlog 511

timeout 0

tcp-keepalive 300

supervised no

pidfile /var/run/redis\_6379.pid

loglevel notice

logfile ""

databases 16

always-show-logo yes

save 900 1

save 300 10

save 60 10000

stop-writes-on-bgsave-error yes

rdbcompression yes

rdbchecksum yes

dbfilename dump.rdb

dir ./

slave-serve-stale-data yes

slave-read-only yes

repl-diskless-sync no

repl-diskless-sync-delay 5

repl-disable-tcp-nodelay no

slave-priority 100

requirepass imcloudep2018 //这个参数设置的redis的登录密码

lazyfree-lazy-eviction no

lazyfree-lazy-expire no

lazyfree-lazy-server-del no

slave-lazy-flush no

appendonly no

appendfilename "appendonly.aof"

appendfsync everysec

no-appendfsync-on-rewrite no

auto-aof-rewrite-percentage 100

auto-aof-rewrite-min-size 64mb

aof-load-truncated yes

aof-use-rdb-preamble no

lua-time-limit 5000

slowlog-log-slower-than 10000

slowlog-max-len 128

latency-monitor-threshold 0

notify-keyspace-events ""

hash-max-ziplist-entries 512

hash-max-ziplist-value 64

list-max-ziplist-size -2

list-compress-depth 0

set-max-intset-entries 512

zset-max-ziplist-entries 128

zset-max-ziplist-value 64

hll-sparse-max-bytes 3000

activerehashing yes

client-output-buffer-limit normal 0 0 0

client-output-buffer-limit slave 256mb 64mb 60

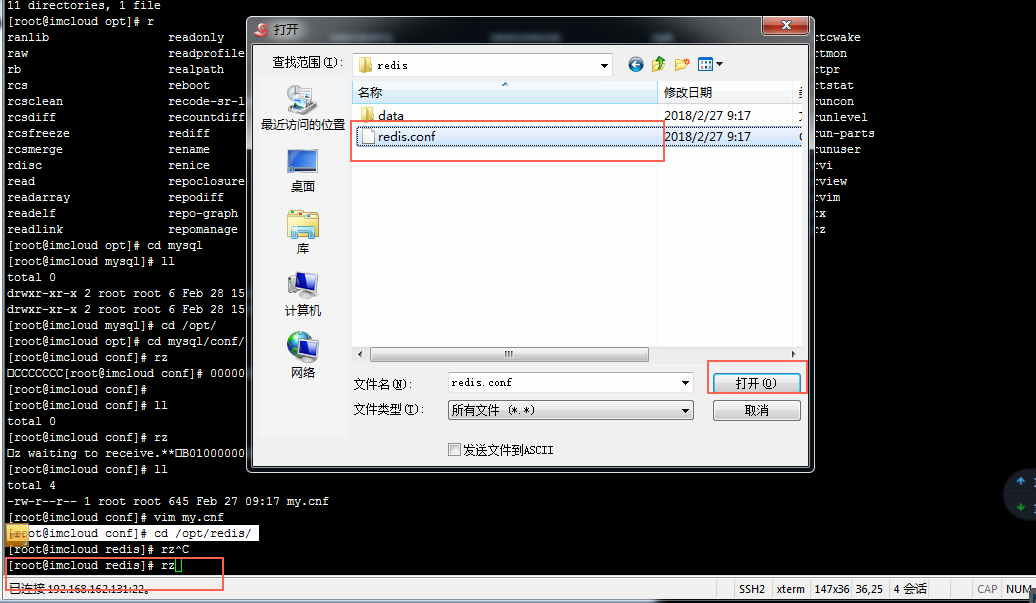
client-output-buffer-limit pubsub 32mb 8mb 60

hz 10

aof-rewrite-incremental-fsync yes

[root@imcloud redis]#

上传已编辑好的redis配置文件到/opt/redis/目录下



最后目录文件结构如下：

[root@imcloud opt]# tree /opt/ -L 1

/opt/

├── gitlab

├── imcloudep

├── mysql

├── rabbitmq

└── redis

5 directories, 0 files

[root@imcloud opt]#

3.通过Docker-compose启动基础容器

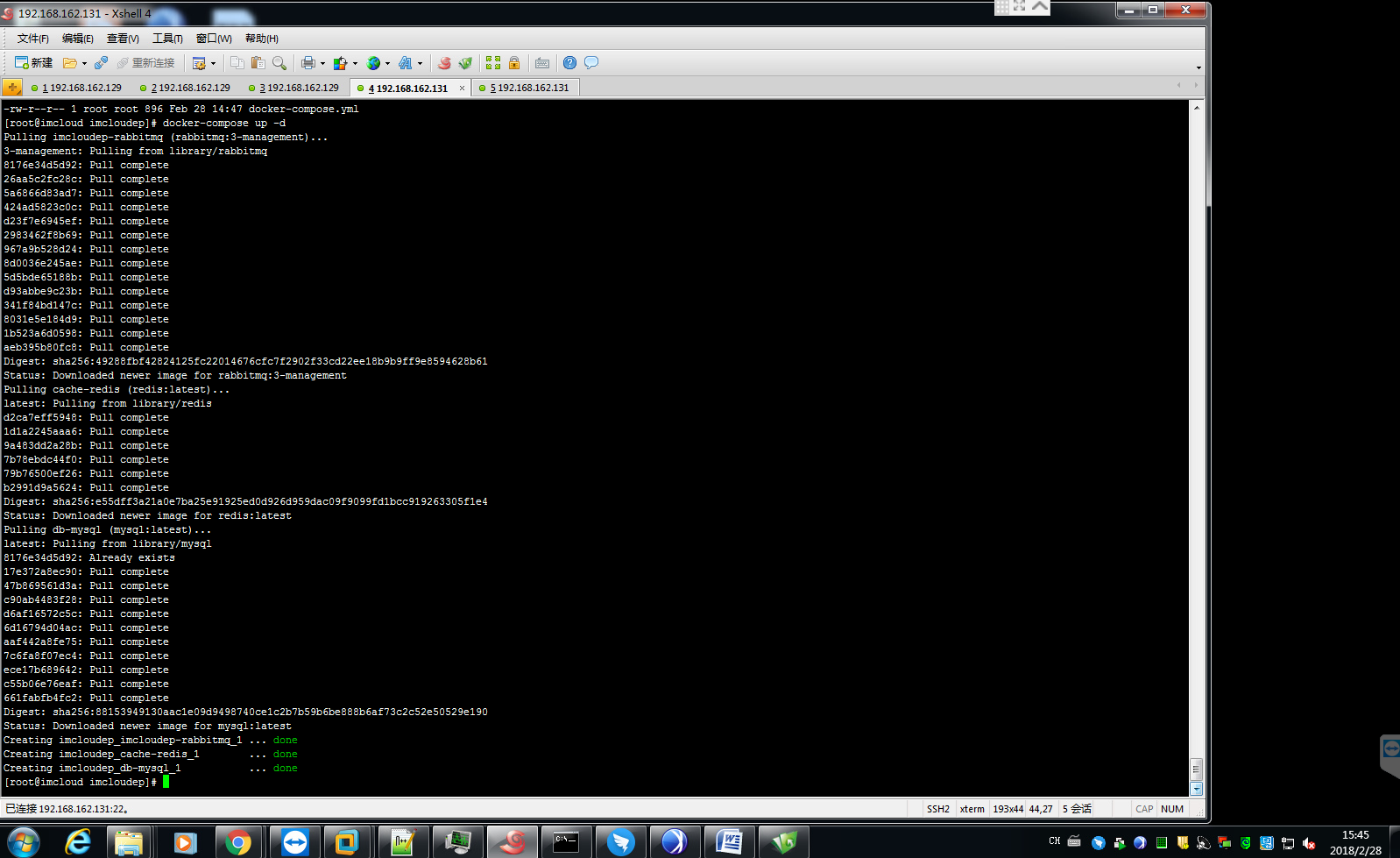
//进入doceker-compose.yml文件所在目录

[root@imcloud redis]# cd /opt/imcloudep/dockercompose-base

//以后台护进程方式启动容器

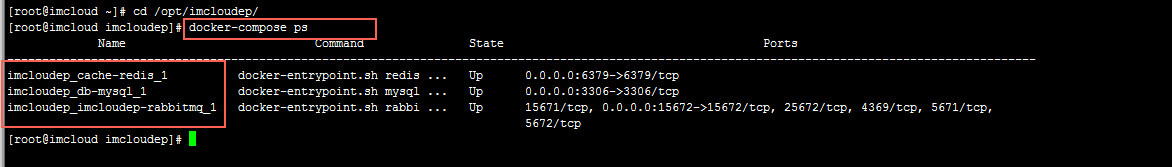
[root@imcloud dockercompose-base]# docker-compose up -d

第一次启动过程中，会去拉取基础镜像，过程如下图：



4.查看启动的容器情况

[root@imcloud imcloudep]# docker-compose ps



验证mysql容器运行参数是否满足预期要求：

[root@centosTemplate src]# mysql -h 192.168.162.131 -uroot -p //密码Hello!!!123

Enter password:

Welcome to the MariaDB monitor. Commands end with ; or \g.

Your MySQL connection id is 3

Server version: 5.7.21 MySQL Community Server (GPL)

Copyright (c) 2000, 2017, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

//查看版本号

MySQL [(none)]> select version();

+-----------+

| version() |

+-----------+

| 5.7.21 |

+-----------+

1 row in set (0.00 sec)

//查看服务器字符集编码是否为utf8

MySQL [(none)]> show variables like '%character%';

+--------------------------+----------------------------+

| Variable\_name | Value |

+--------------------------+----------------------------+

| character\_set\_client | utf8 |

| character\_set\_connection | utf8 |

| character\_set\_database | utf8 |

| character\_set\_filesystem | binary |

| character\_set\_results | utf8 |

| character\_set\_server | utf8 |

| character\_set\_system | utf8 |

| character\_sets\_dir | /usr/share/mysql/charsets/ |

+--------------------------+----------------------------+

8 rows in set (0.02 sec)

//查看sql\_mode变量配置

MySQL [(none)]> show variables like '%sql\_mode%';

+---------------+------------------------------------------------------------------------------------------------------------------------+

| Variable\_name | Value |

+---------------+------------------------------------------------------------------------------------------------------------------------+

| sql\_mode | STRICT\_TRANS\_TABLES,NO\_ZERO\_IN\_DATE,NO\_ZERO\_DATE,ERROR\_FOR\_DIVISION\_BY\_ZERO,NO\_AUTO\_CREATE\_USER,NO\_ENGINE\_SUBSTITUTION |

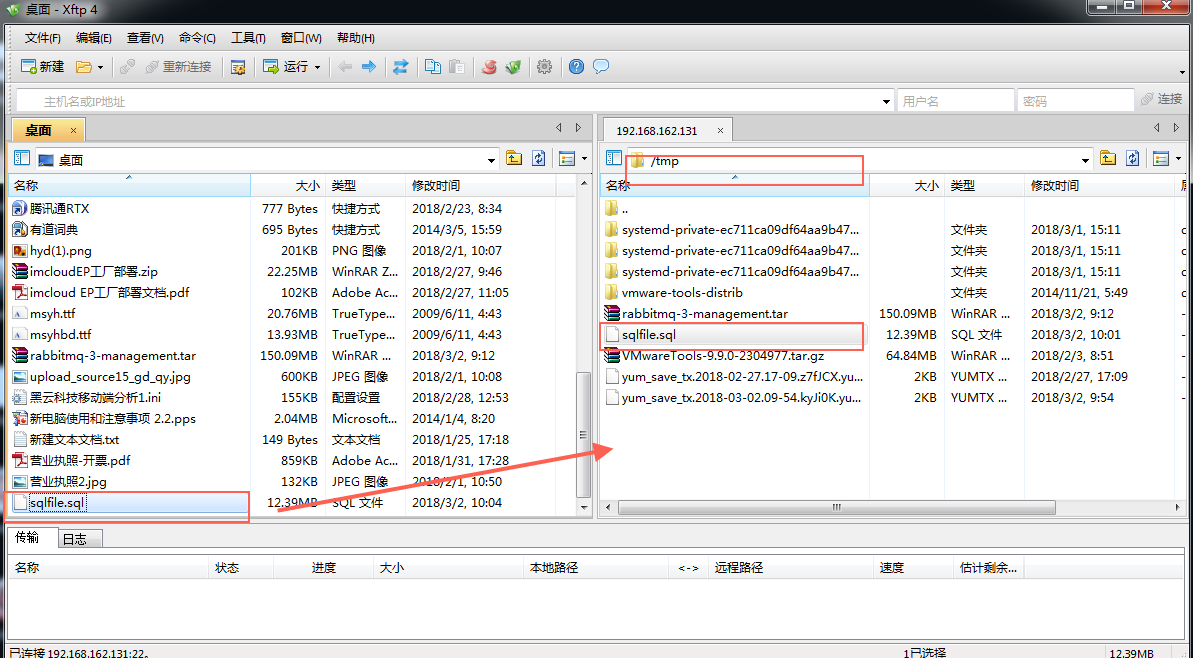
+---------------+------------------------------------------------------------------------------------------------------------------------+

1 row in set (0.01 sec)



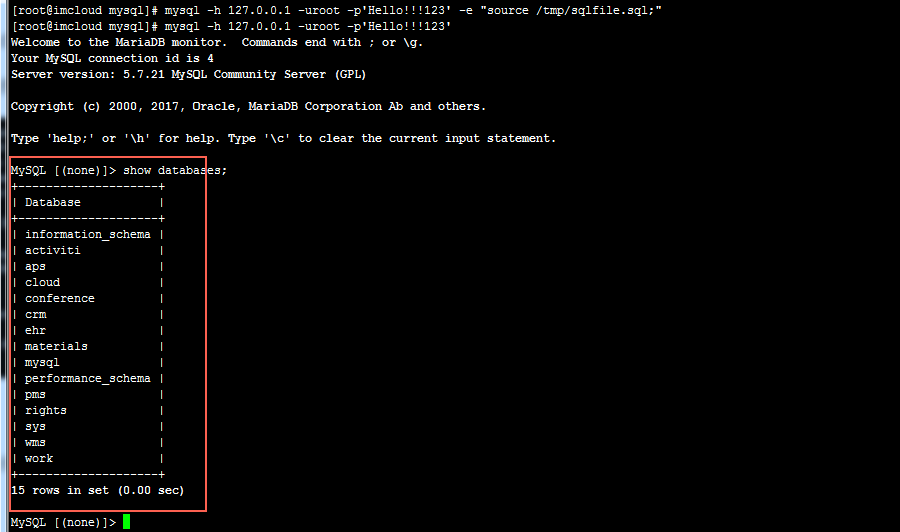
为MySQL导入数据库

通过xftp软件上传之前导出的数据库备份到/tmp/目录下：



导入数据库

[root@imcloud mysql]# mysql -h 127.0.0.1 -uroot -p'Hello!!!123' -e "source /tmp/sqlfile.sql;"



验证redis容器：

//获取容器的IP

[root@imcloud imcloudep]# docker inspect --format='{{range .NetworkSettings.Networks}}{{.IPAddress}}{{end}}' imcloudep\_cache-redis\_1

172.18.0.4

[root@imcloud imcloudep]#

//主机安装redis-cli客户端工具，登录密码为imcloudep2018

[root@imcloud imcloudep]# cd /usr/local/src/

[root@imcloud src]# wget http://download.redis.io/releases/redis-2.8.17.tar.gz

[root@imcloud src]# tar xzf redis-2.8.17.tar.gz

[root@imcloud src]# cd redis-2.8.17/

[root@imcloud redis-2.8.17]# make

[root@imcloud redis-2.8.17]# cd src

[root@imcloud src]# ./redis-cli -h 192.168.162.131

192.168.162.131:6379> auth imcloudep2018 //输入认证密码imcloudep2018

OK

192.168.162.131:6379> ping

PONG

192.168.162.131:6379> set foo bar

OK

192.168.162.131:6379> get foo

"bar"

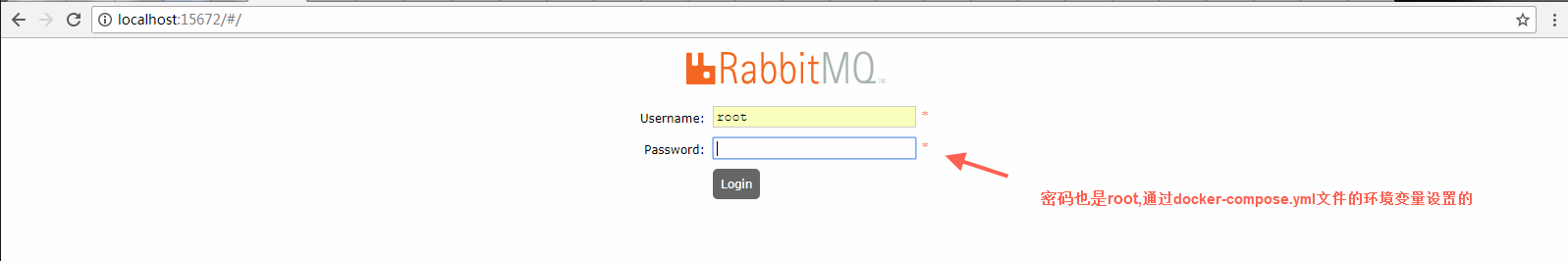
192.168.162.131:6379>

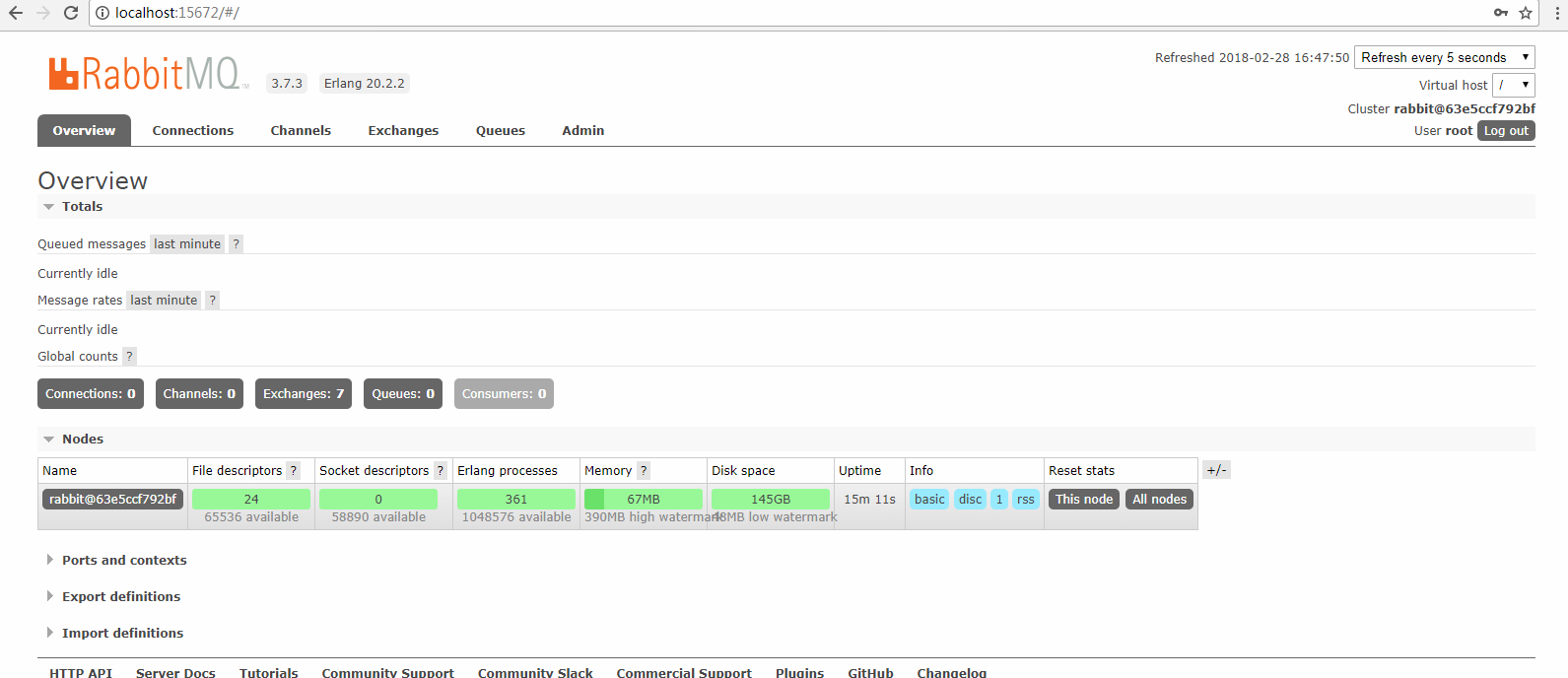
验证rabbitmq容器：

登录rabbitmq管理系统，登录地址为：http://{IP}:15672

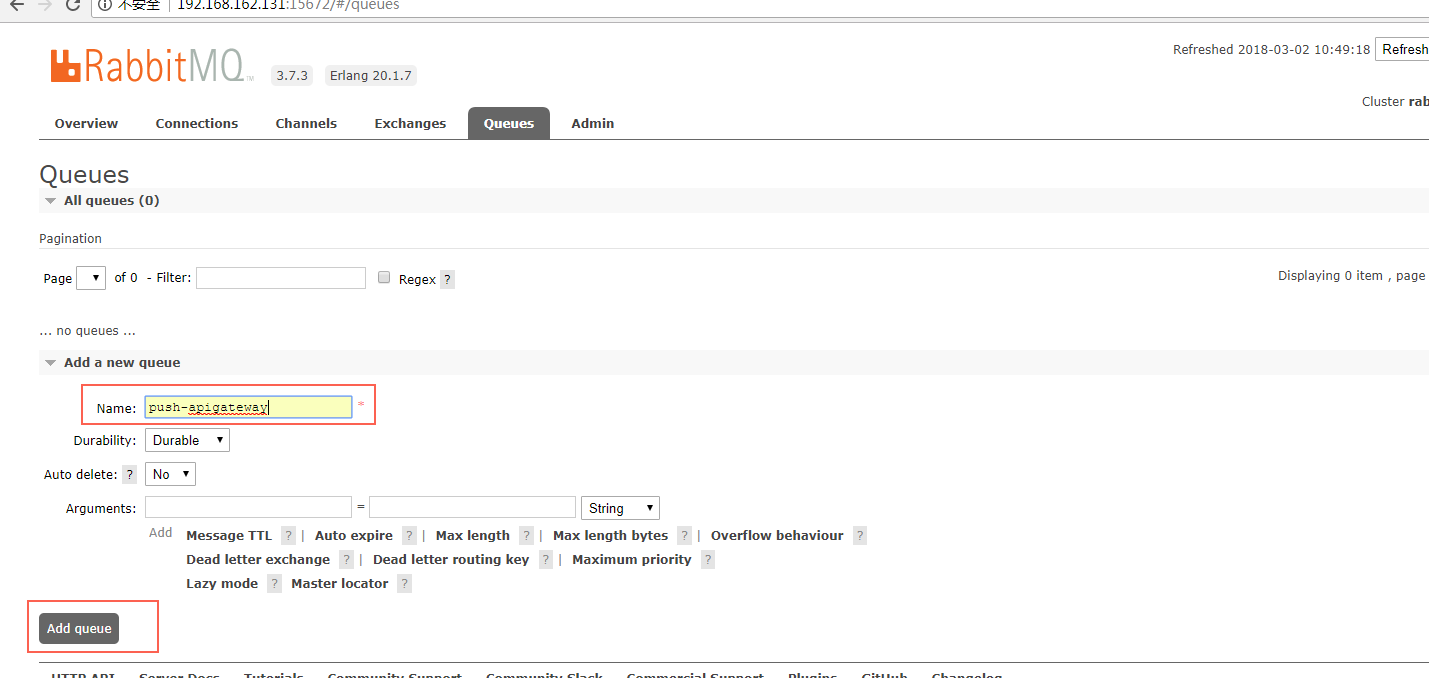
用户名：root

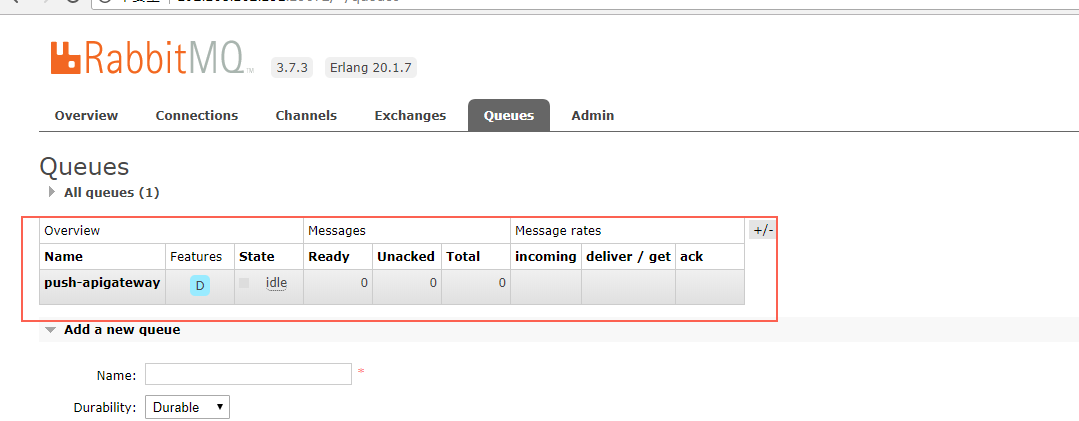
密码：root





由于在微服务core-apigateway中，要使用到push-apigateway队列，因此需要创建





部署gitlab-ce容器（可选）

1.编辑docker-compose.yml文件

[root@imcloud imcloudep]# pwd

/opt/imcloudep

[root@imcloud imcloudep]# vim docker-compose.yml

//配置如下：

version: '3'

services:

db-mysql:

image: mysql #lkjlkfaj

volumes:

- /opt/mysql/data:/var/lib/mysql

- /opt/mysql/conf:/etc/mysql/conf.d

- /etc/localtime:/etc/localtime

environment:

- MYSQL\_ROOT\_PASSWORD=Hello!!!123

ports:

- "3306:3306"

command: mysqld --character-set-server=utf8 --collation-server=utf8\_unicode\_ci

cache-redis:

image: redis

ports:

- "6379:6379"

volumes:

- /opt/redis/redis.conf:/usr/local/etc/redis/redis.conf

- /opt/redis/data:/data

command: redis-server /usr/local/etc/redis/redis.conf

imcloudep-rabbitmq:

hostname: imcloudep-rabbitmq //让创建的队例持久化

image: rabbitmq:3-management

environment:

- RABBITMQ\_DEFAULT\_USER=root

- RABBITMQ\_DEFAULT\_PASS=root

volumes:

- /opt/rabbitmq:/var/lib/rabbitmq

ports:

- "15672:15672"

gitlab-ce:

image: 'gitlab/gitlab-ce:latest'

restart: always

hostname: 'gitlab.test.com'

environment:

GITLAB\_OMNIBUS\_CONFIG: |

external\_url 'http://gitlab.test.com:9090'

gitlab\_rails['gitlab\_shell\_ssh\_port'] = 2224

ports:

- '9090:9090'

- '2224:22'

volumes:

- '/opt/gitlab/config:/etc/gitlab'

- '/opt/gitlab/logs:/var/log/gitlab'

- '/opt/gitlab/data:/var/opt/gitlab'

networks:

default:

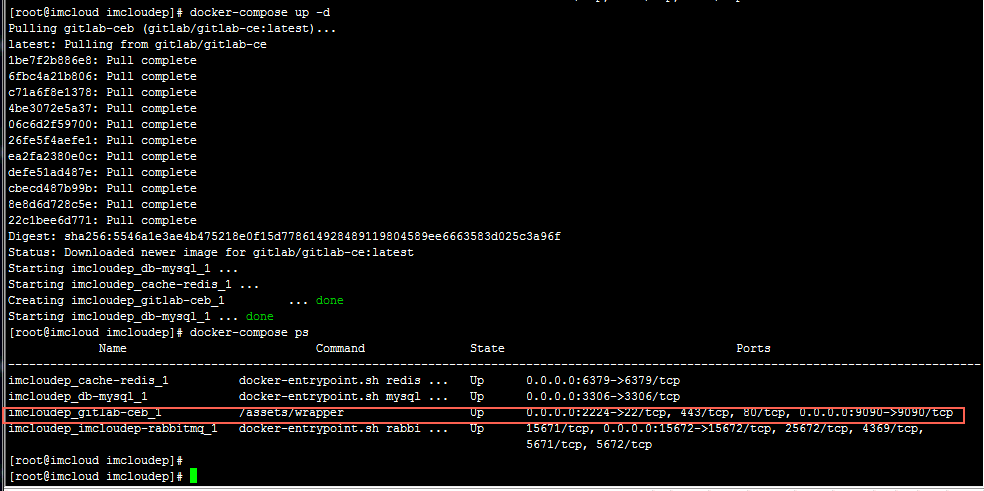
external:

name: imcloudep-external

2.运行docker-compose up -d 命令

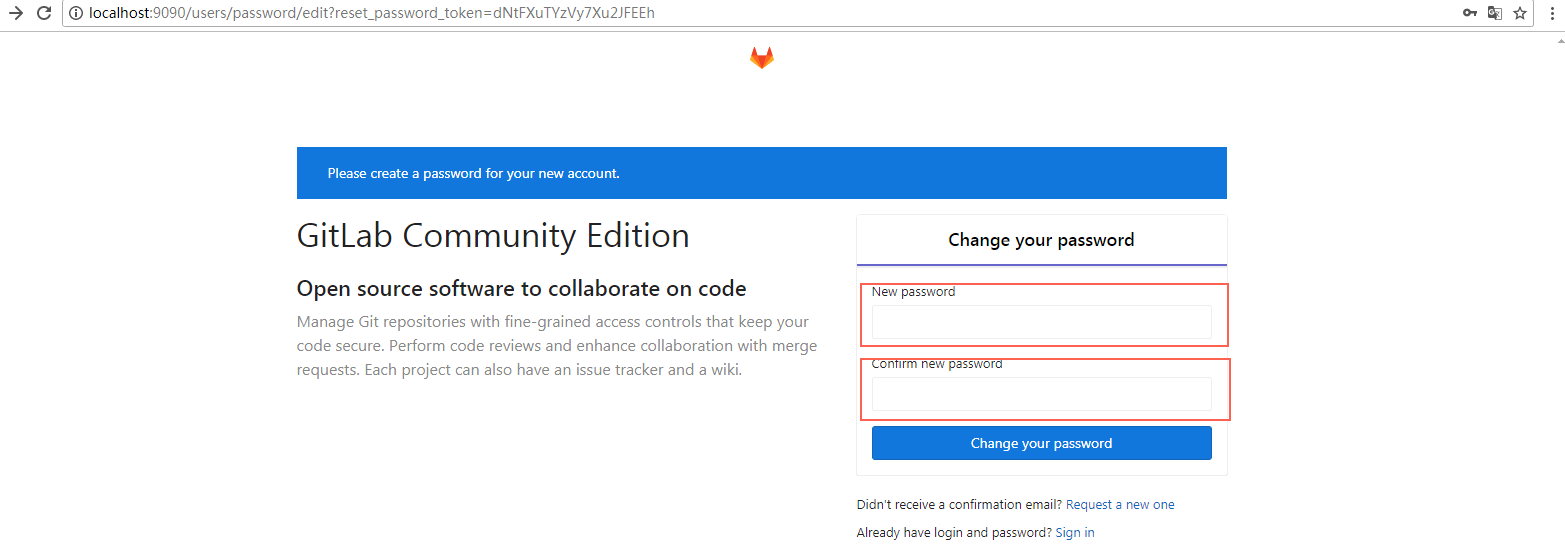
[root@imcloud imcloudep]# docker-compose up -d

[root@imcloud imcloudep]# docker-compose ps

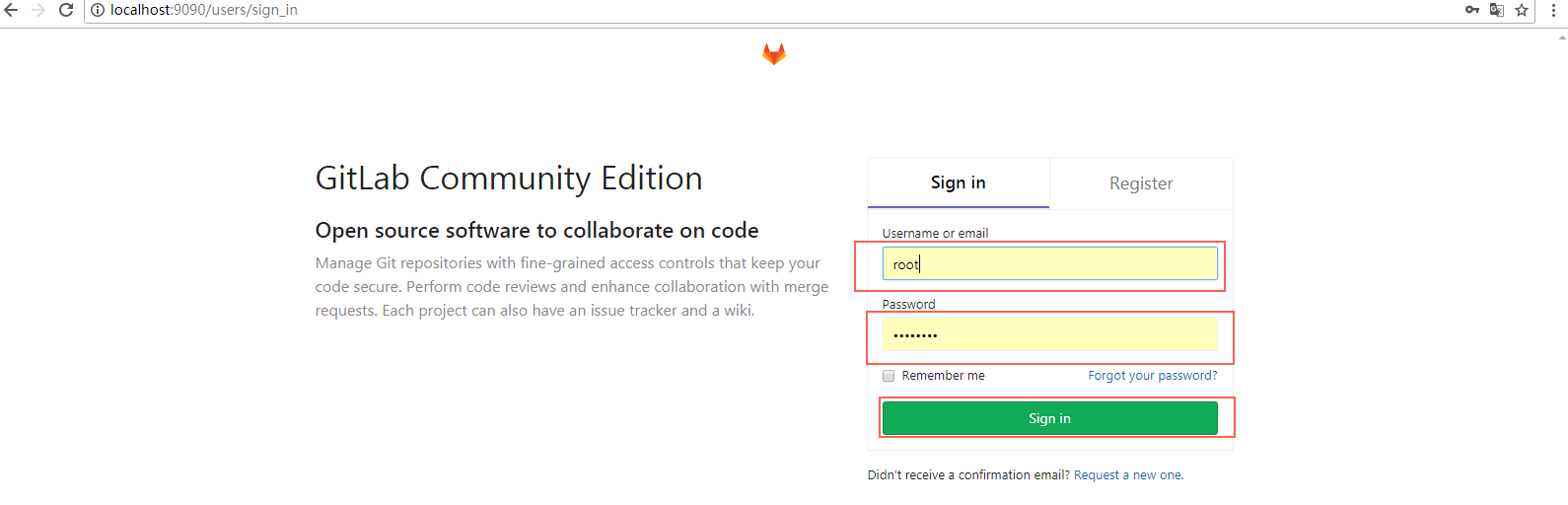


3.验证gitlab仓库

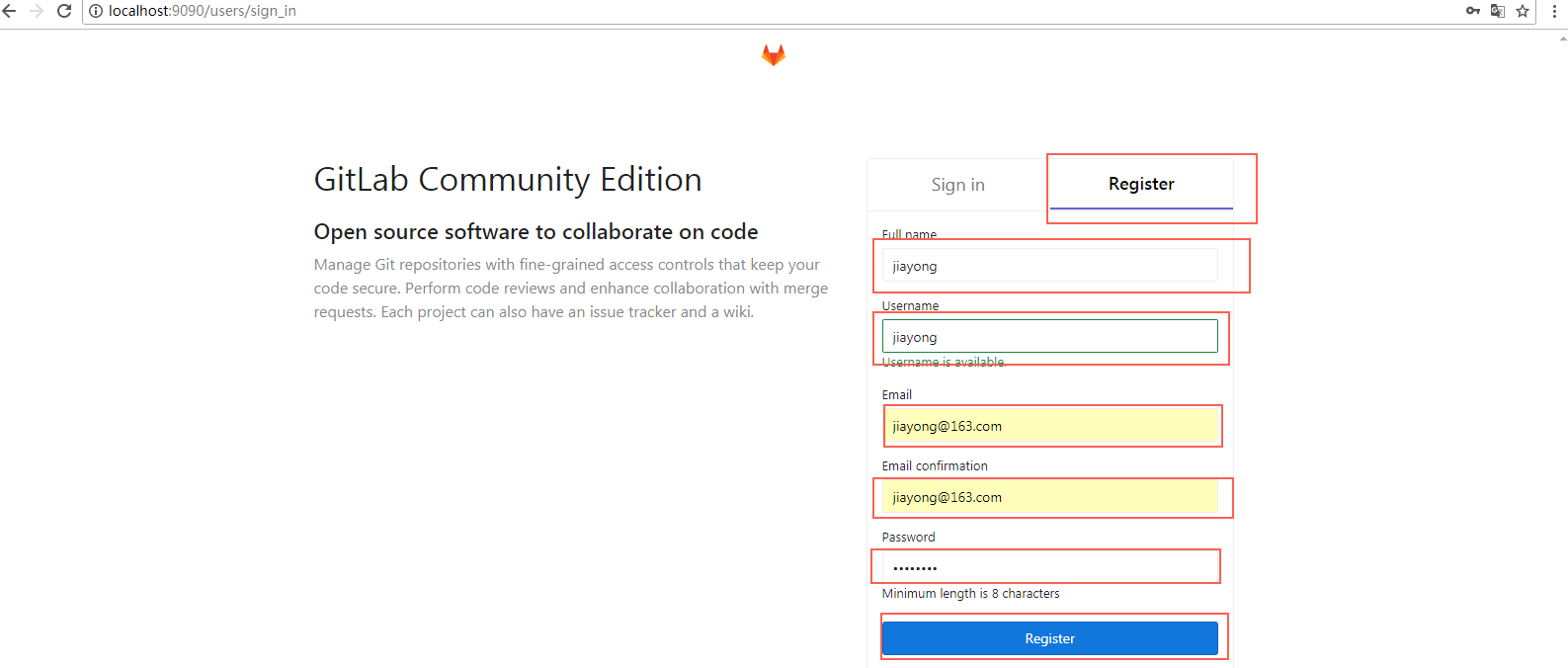
第一次登录时，会要求修改root用户的密码，密码必须满足8位



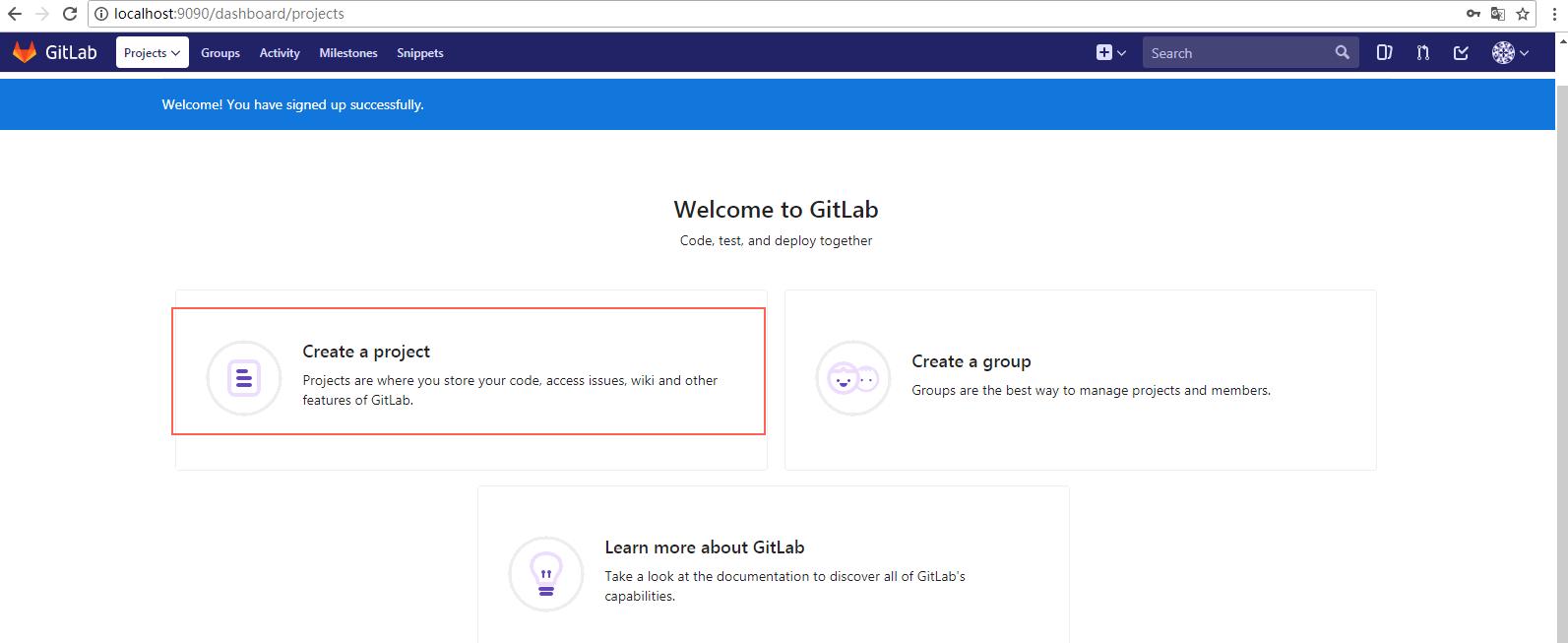
以root用户及修改后的密码登录

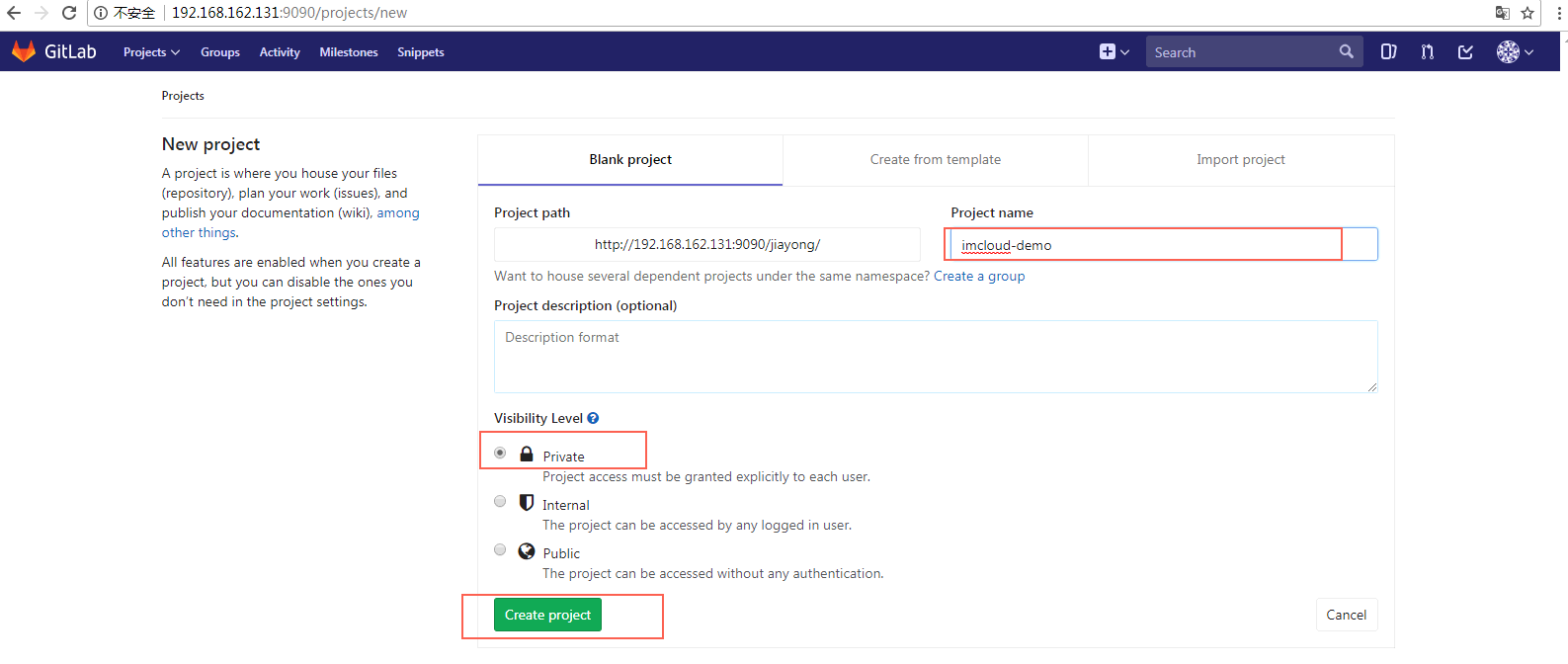


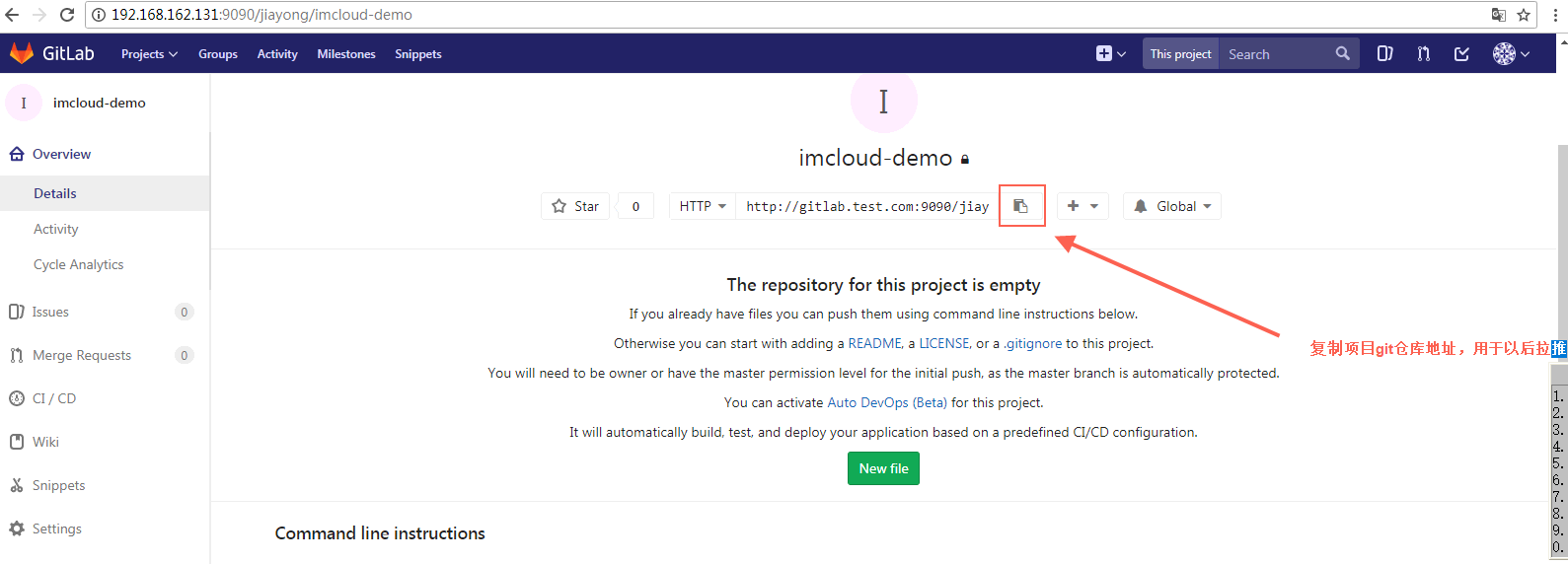
注册一个新用户



以注册的新用户登录，并创建一个空项目仓库

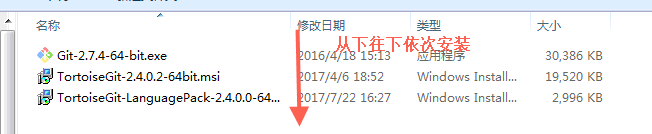




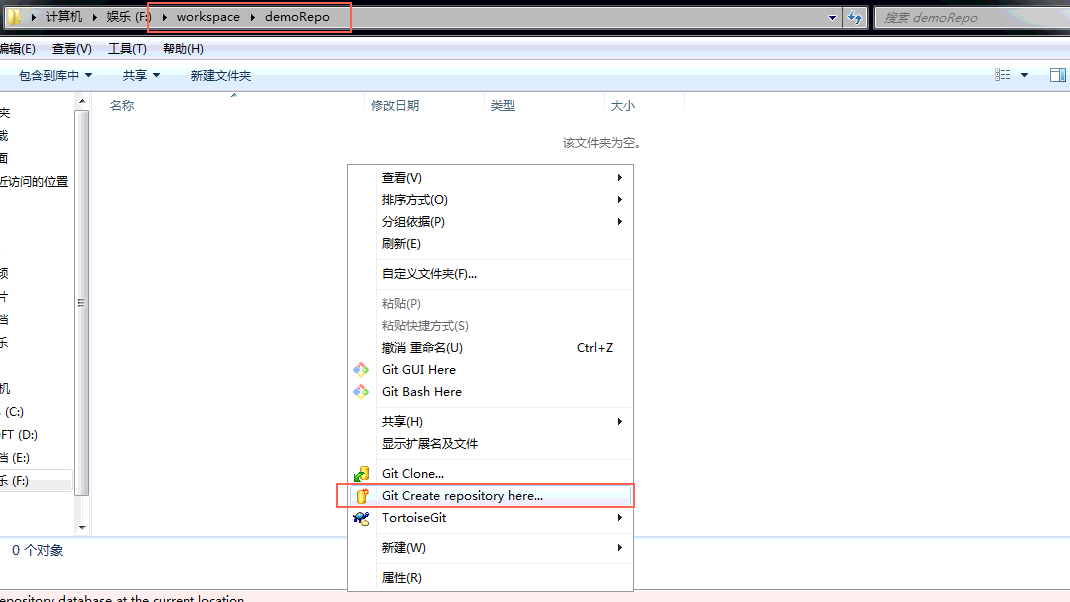


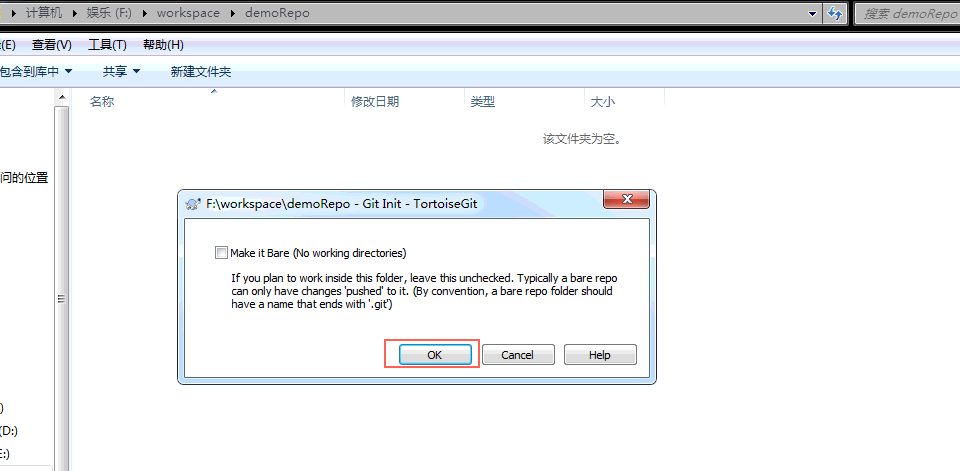
测试gitlab仓库

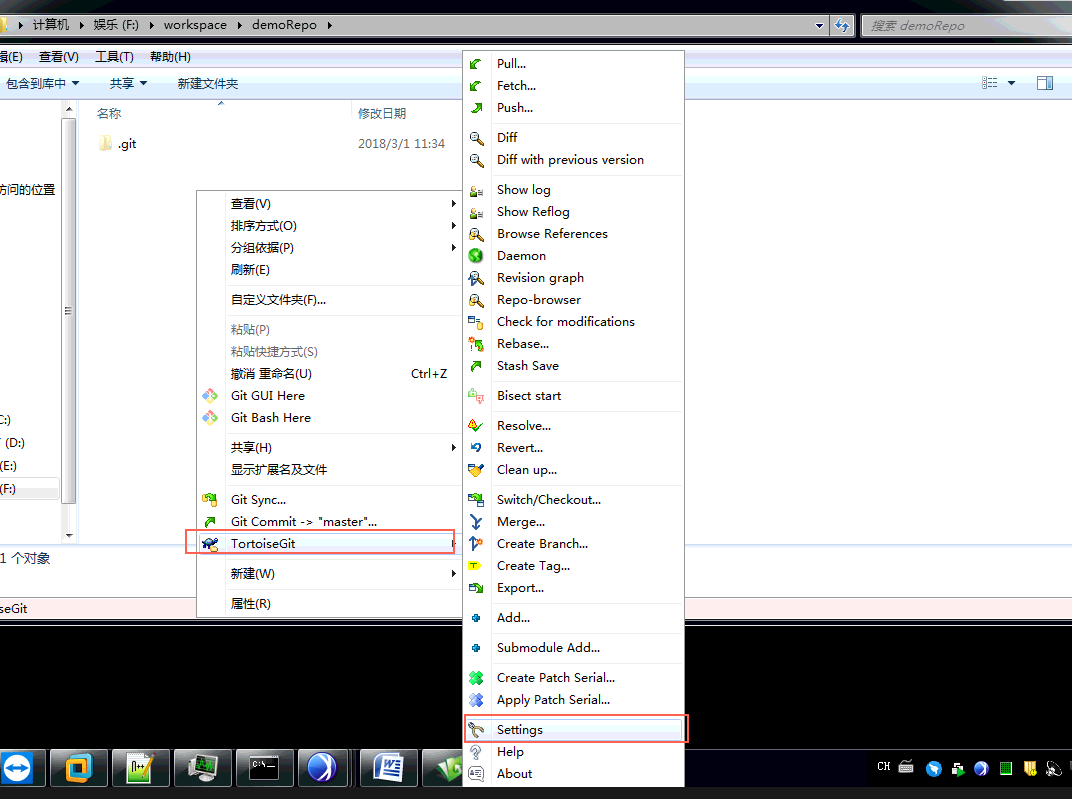
windows上安装客户端软件

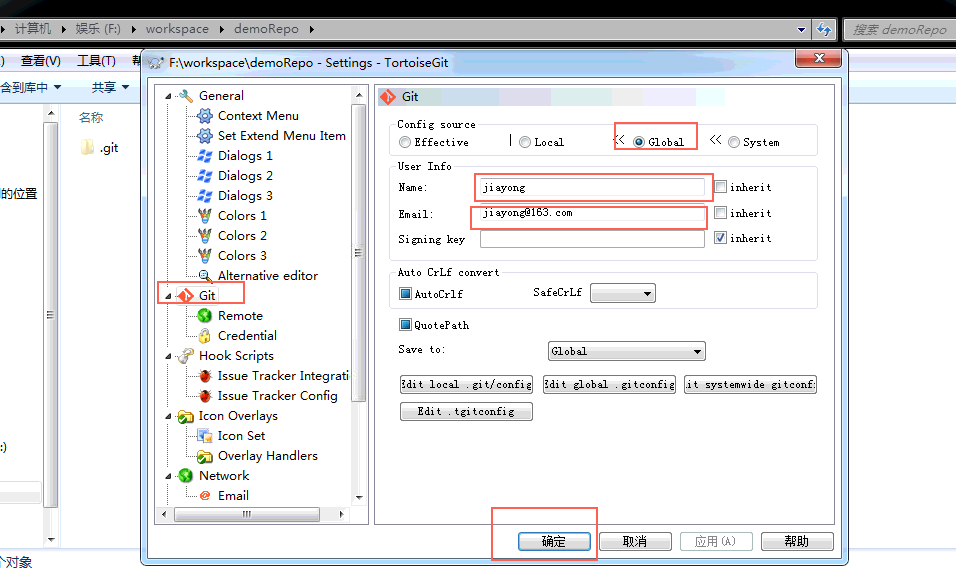


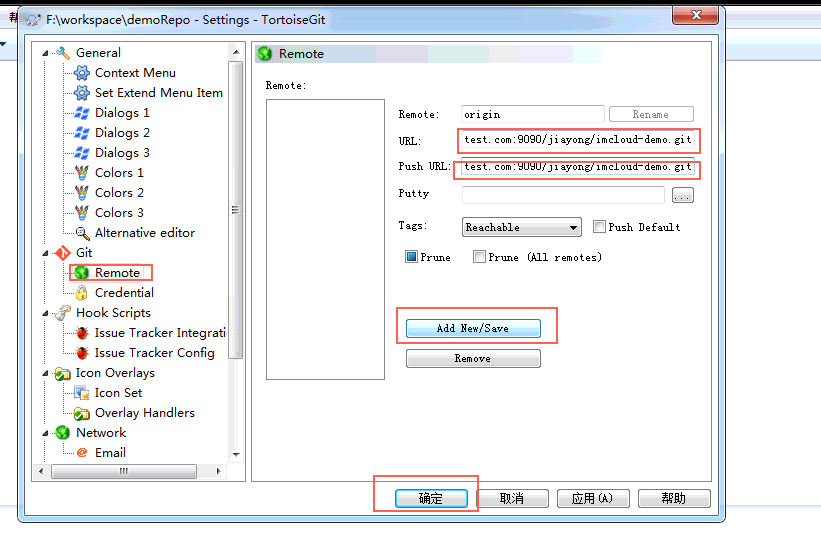
创建一个目录做本地仓库目录，进入目录，空白处右键->选择“git create repository here..”



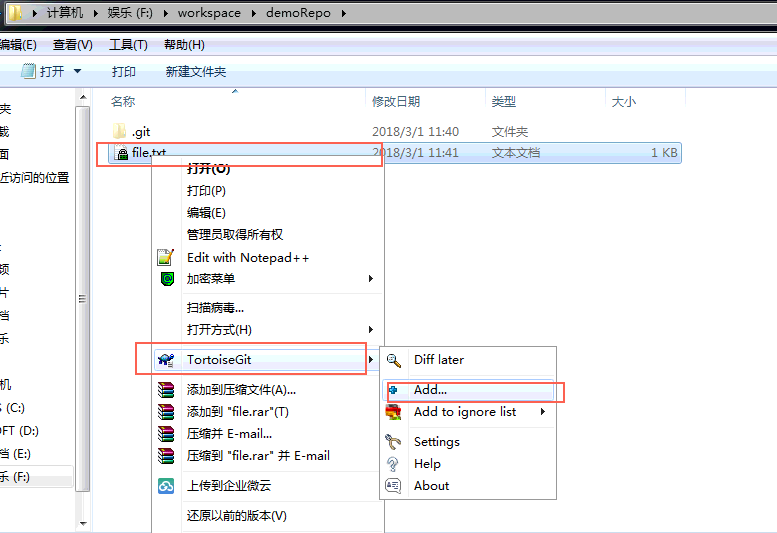


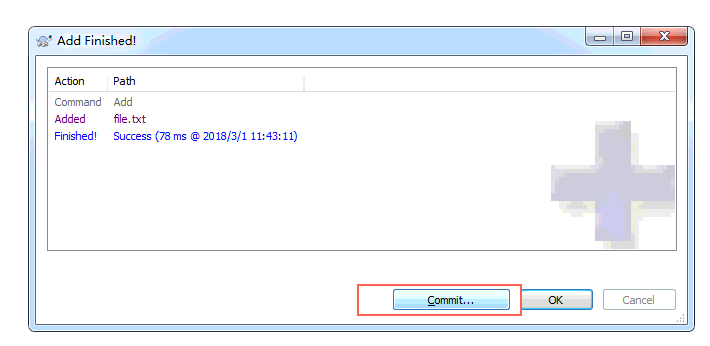


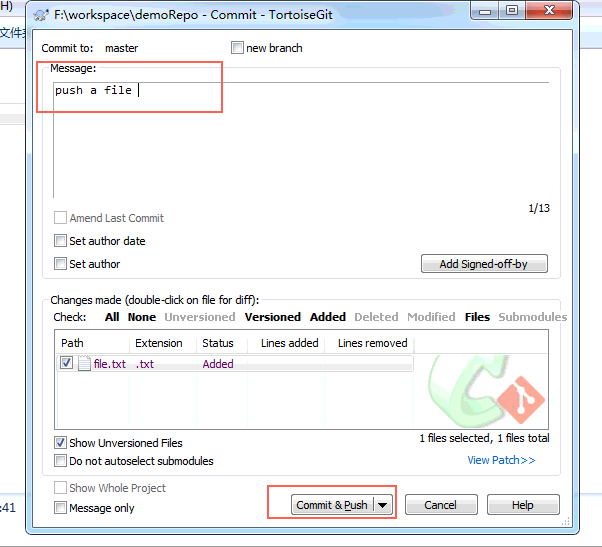


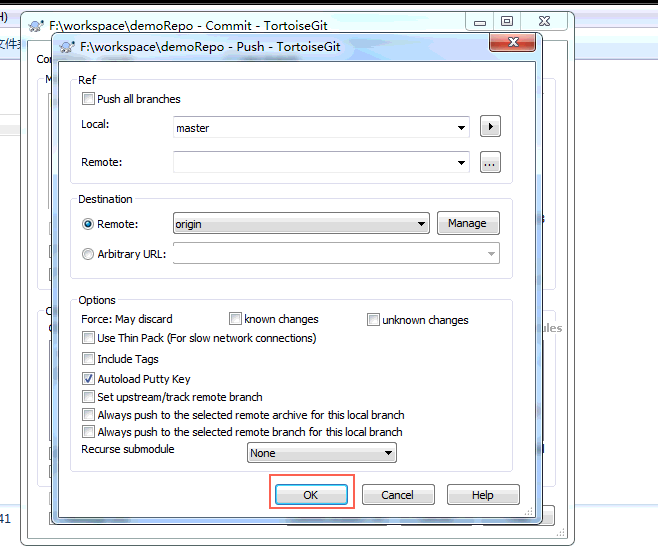


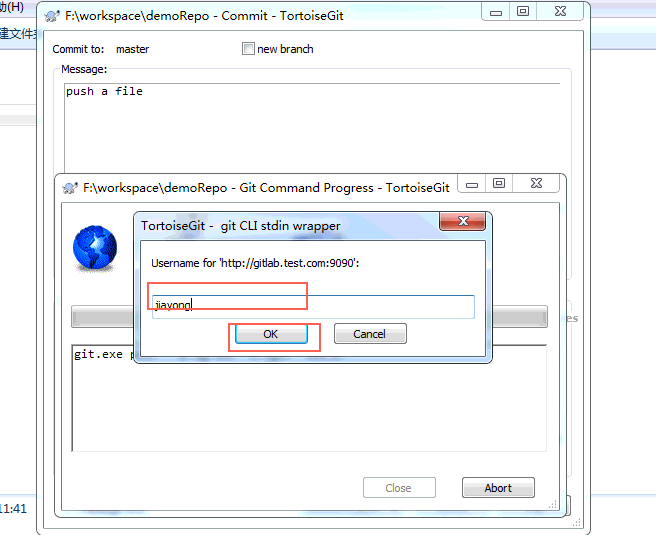
创建一个文件，并push到项目仓库

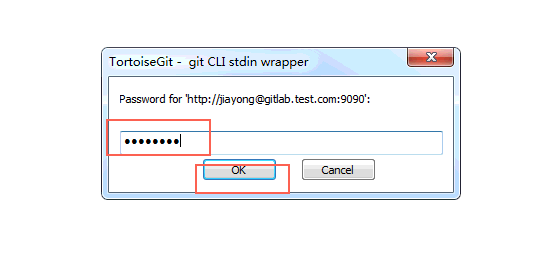


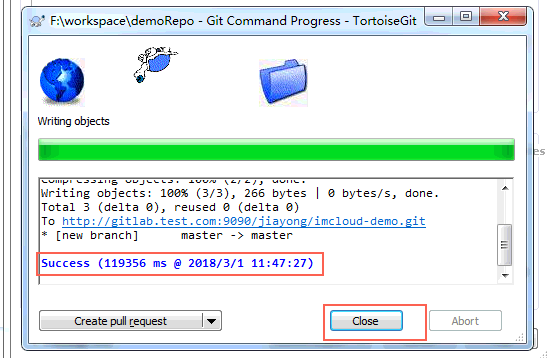




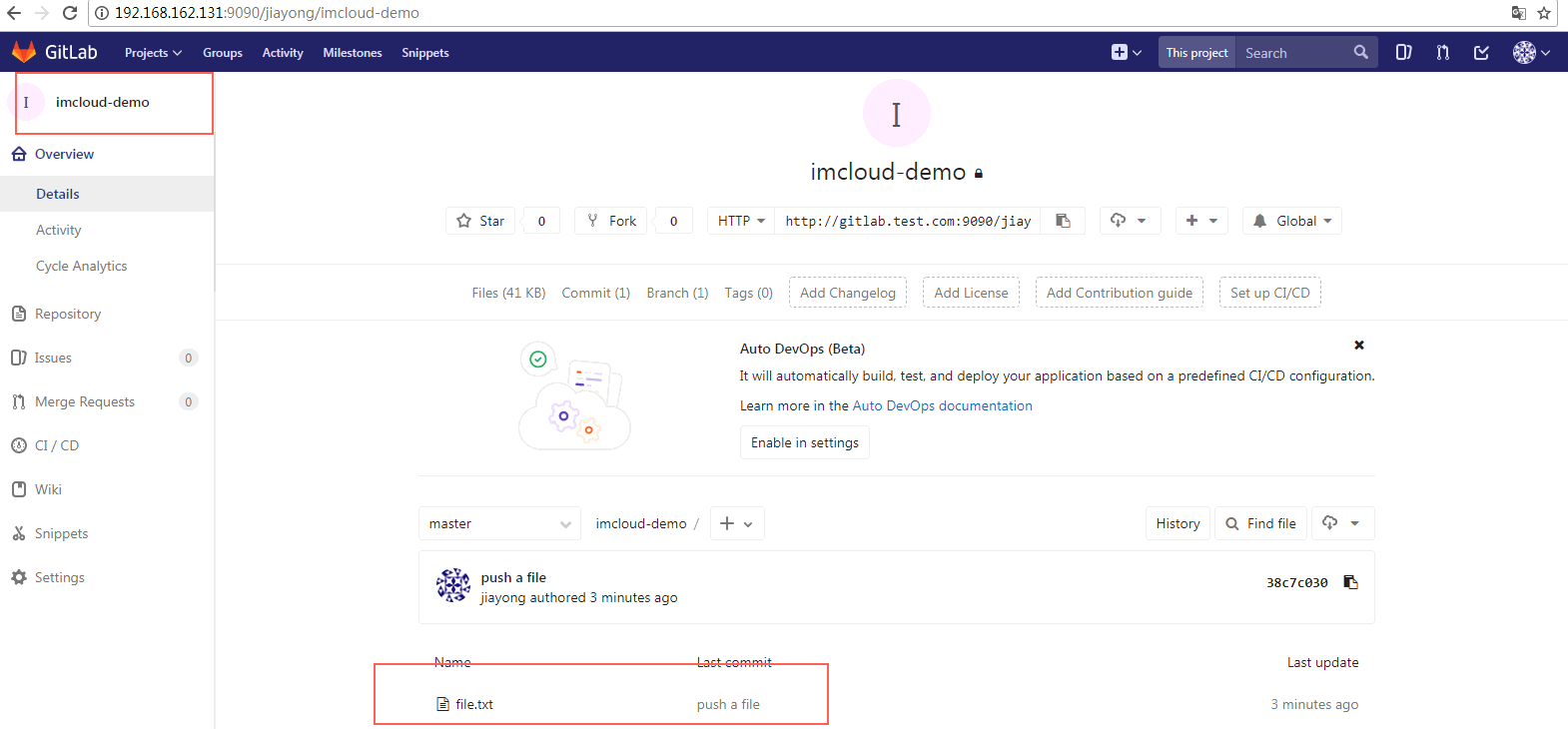








成功推到gitlab项目仓库

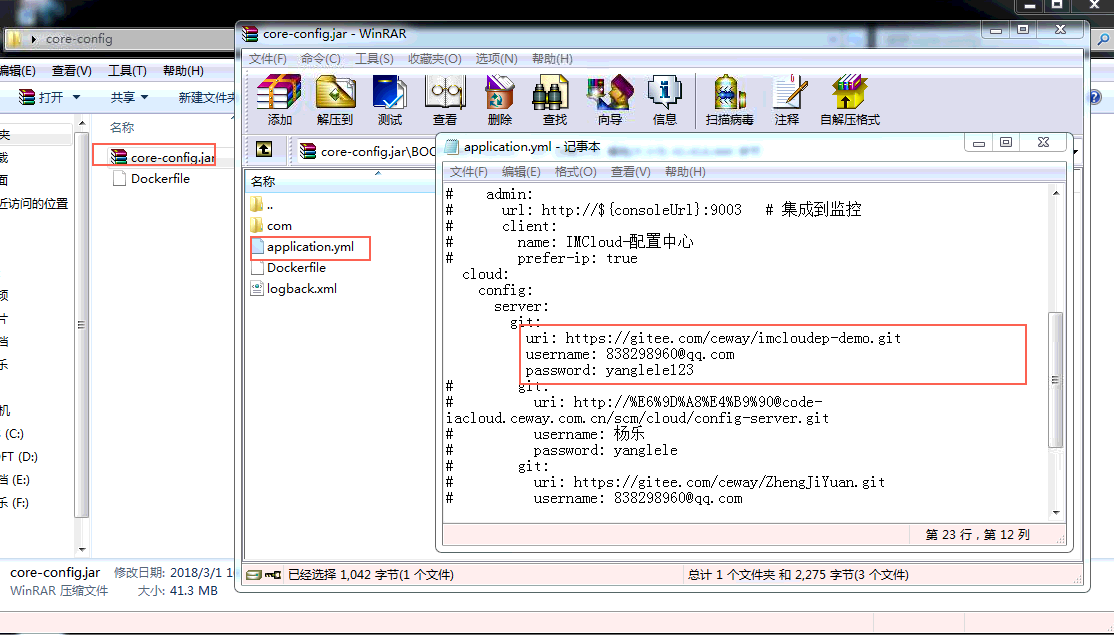


部署微服务镜像

在微服务中架构中由于core-config服务，为其它微服务提供配置文件；而core-config微服务中提供的配置文件是通过git仓库下载而来的。git仓库的http地址，是写在core-config项目代码的application.yml文件中的，不同的工厂可能git仓库地址不一样，因此core-config镜像需要重新构建。

可以通过一些压缩软件打开core-config.jar文件，修改application.yml文件中的红色标记代码，然后进行保存，进行重新制作镜像即可。

修改core-config项目代码中git仓库地址



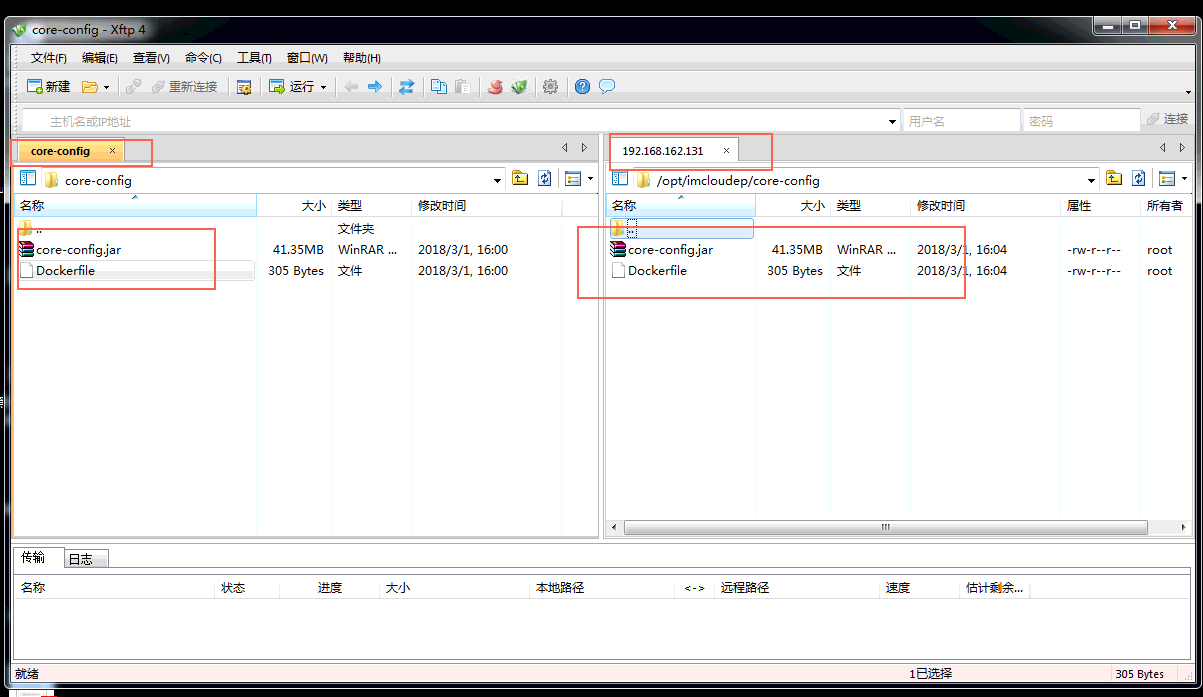
在服务器上创建core-config构建目录

[root@imcloud imcloudep]# pwd

/opt/imcloudep

[root@imcloud imcloudep]# mkdir core-config/

将core-conifg的jar包及Dockerfile文件上传到目标服务器



编辑Dockerfile文件

[root@imcloud core-config]# vim Dockerfile

//配置如下：

FROM java:8

ENV TZ=Asia/Shanghai

RUN ln -snf /usr/share/zoneinfo/$TZ /etc/localtime && echo $TZ > /etc/timezone //基础镜像包

VOLUME /tmp //指定容器的挂载巻

ADD core-config.jar core-config.jar //将当前目录的jar包拷贝到容器根目录

RUN bash -c 'touch /core-config.jar' //修改jar时间戮

EXPOSE 9999 //容器暴露的端口

ENTRYPOINT ["java","-Djava.security.egd=file:/dev/./urandom","-jar","/core-config.jar"]

//容器启动后，程序运行入口点

构建core-config镜像

[root@imcloud core-config]# ll

total 42344

-rw-r--r-- 1 root root 43352940 Mar 1 16:24 core-config.jar

-rw-r--r-- 1 root root 305 Mar 1 17:24 Dockerfile

[root@imcloud core-config]# docker build -t imcloudep/core-config .

//为镜像命令的时候，前面是项目名，后面是服务名,后面有一个点，不要忽略了

Sending build context to Docker daemon 43.36MB

Step 1/8 : FROM java:8

---> d23bdf5b1b1b

Step 2/8 : ENV TZ=Asia/Shanghai

---> Running in c20d6598732b

Removing intermediate container c20d6598732b

---> 874f1cc66d9d

Step 3/8 : RUN ln -snf /usr/share/zoneinfo/$TZ /etc/localtime && echo $TZ > /etc/timezone

---> Running in b0e69c13ea6b

Removing intermediate container b0e69c13ea6b

---> 3c8dbfaf8720

Step 4/8 : VOLUME /tmp

---> Running in e3eeb1bc4408

Removing intermediate container e3eeb1bc4408

---> d3b40dba2b68

Step 5/8 : ADD core-config.jar core-config.jar

---> 17ccb30dec5c

Step 6/8 : RUN bash -c 'touch /core-config.jar'

---> Running in d1631f95220d

Removing intermediate container d1631f95220d

---> 069043b64dc4

Step 7/8 : EXPOSE 9999

---> Running in 6e608dabd405

Removing intermediate container 6e608dabd405

---> a1724d53f0be

Step 8/8 : ENTRYPOINT ["java","-Djava.security.egd=file:/dev/./urandom","-jar","/core-config.jar"]

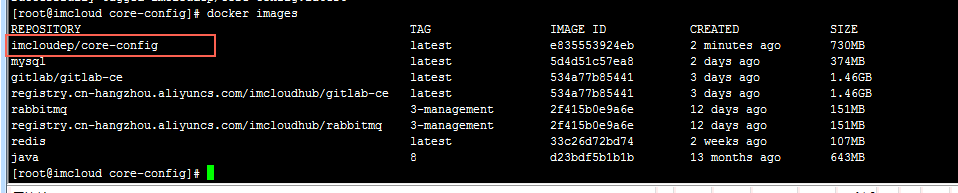
---> Running in 53283bf28d05

Removing intermediate container 53283bf28d05

---> e835553924eb

Successfully built e835553924eb

Successfully tagged imcloudep/core-config:latest



其它的微服镜像如法炮制已经构建好，并上传到了阿里云docker镜像仓库，阿里云帐号密码（jimaisi2017/Yy123456），注意下载命名空间为imcloudep的为测试包镜像，imcloudep-pro的为正式包镜像。

在imcloudep项目下创建dockercompose-core 和 dockercompose-function目录，用于存放核心的和功能的docker-compose.yml服务编排文件

[root@imcloud dockercompose-core]# cd /opt/imcloudep/

[root@imcloud imcloudep]# mkdir dockercompose-core/ dockercompose-function/

[root@imcloud imcloudep]# cd dockercompose-core/

[root@imcloud dockercompose-core]# vim docker-compose.yml

//配置内容如下：

version: '3'

services:

core-eureka:

image: registry.cn-hangzhou.aliyuncs.com/imcloudep/core-eureka

ports:

- "9000:9000"

core-config:

image: imcloudep/core-config

ports:

- "9999:9999"

environment:

- EUREKA\_SERVER\_ADDRESS=core-eureka

networks:

default:

external:

name: imcloudep-external

[root@imcloud dockercompose-core]#

[root@imcloud dockercompose-core]# cd ../dockercompose-function/

[root@imcloud dockercompose-function]# vim docker-compose.yml

[root@imcloud dockercompose-function]#

//配置内容如下：

version: '3'

services:

core-apigateway:

image: registry.cn-hangzhou.aliyuncs.com/imcloudep/core-apigateway

ports:

- "9001:9001"

environment:

- EUREKA\_SERVER\_ADDRESS=core-eureka

- DB\_INSTANCE=db-mysql

base-push:

image: registry.cn-hangzhou.aliyuncs.com/imcloudep/base-push

ports:

- "9008:9008"

environment:

- EUREKA\_SERVER\_ADDRESS=core-eureka

ehr-profile:

image: registry.cn-hangzhou.aliyuncs.com/imcloudep/ehr-profile

environment:

- EUREKA\_SERVER\_ADDRESS=core-eureka

- DB\_INSTANCE=db-mysql

ehr-signin:

image: registry.cn-hangzhou.aliyuncs.com/imcloudep/ehr-signin

environment:

- EUREKA\_SERVER\_ADDRESS=core-eureka

- DB\_INSTANCE=db-mysql

mms-materialsweb:

image: registry.cn-hangzhou.aliyuncs.com/imcloudep/mms-materialsweb

environment:

- EUREKA\_SERVER\_ADDRESS=core-eureka

- DB\_INSTANCE=db-mysql

platform-rights:

image: registry.cn-hangzhou.aliyuncs.com/imcloudep/platform-rights

environment:

- EUREKA\_SERVER\_ADDRESS=core-eureka

- DB\_INSTANCE=db-mysql

pms-purchase:

image: registry.cn-hangzhou.aliyuncs.com/imcloudep/pms-purchase

environment:

- EUREKA\_SERVER\_ADDRESS=core-eureka

- DB\_INSTANCE=db-mysql

pms-supplier:

image: registry.cn-hangzhou.aliyuncs.com/imcloudep/pms-supplier

environment:

- EUREKA\_SERVER\_ADDRESS=core-eureka

- DB\_INSTANCE=db-mysql

pms-app:

image: registry.cn-hangzhou.aliyuncs.com/imcloudep/pms-app

environment:

- EUREKA\_SERVER\_ADDRESS=core-eureka

- DB\_INSTANCE=db-mysql

aps-plan:

image: registry.cn-hangzhou.aliyuncs.com/imcloudep/aps-plan

environment:

- EUREKA\_SERVER\_ADDRESS=core-eureka

- DB\_INSTANCE=db-mysql

aps-order:

image: registry.cn-hangzhou.aliyuncs.com/imcloudep/aps-order

environment:

- EUREKA\_SERVER\_ADDRESS=core-eureka

- DB\_INSTANCE=db-mysql

aps-technology:

image: registry.cn-hangzhou.aliyuncs.com/imcloudep/aps-technology

environment:

- EUREKA\_SERVER\_ADDRESS=core-eureka

- DB\_INSTANCE=db-mysql

aps-workorder:

image: registry.cn-hangzhou.aliyuncs.com/imcloudep/aps-workorder

environment:

- EUREKA\_SERVER\_ADDRESS=core-eureka

- DB\_INSTANCE=db-mysql

wfs-common:

image: registry.cn-hangzhou.aliyuncs.com/imcloudep/wfs-common

environment:

- EUREKA\_SERVER\_ADDRESS=core-eureka

- DB\_INSTANCE=db-mysql

wf-special:

image: registry.cn-hangzhou.aliyuncs.com/imcloudep/wf-special

environment:

- EUREKA\_SERVER\_ADDRESS=core-eureka

- DB\_INSTANCE=db-mysql

wms-management:

image: registry.cn-hangzhou.aliyuncs.com/imcloudep/wms-management

environment:

- EUREKA\_SERVER\_ADDRESS=core-eureka

- DB\_INSTANCE=db-mysql

wms-materials:

image: registry.cn-hangzhou.aliyuncs.com/imcloudep/wms-materials

environment:

- EUREKA\_SERVER\_ADDRESS=core-eureka

- DB\_INSTANCE=db-mysql

wms-outbound:

image: registry.cn-hangzhou.aliyuncs.com/imcloudep/wms-outbound

environment:

- EUREKA\_SERVER\_ADDRESS=core-eureka

- DB\_INSTANCE=db-mysql

wms-stock:

image: registry.cn-hangzhou.aliyuncs.com/imcloudep/wms-stock

environment:

- EUREKA\_SERVER\_ADDRESS=core-eureka

- DB\_INSTANCE=db-mysql

wms-warehousing:

image: registry.cn-hangzhou.aliyuncs.com/imcloudep/wms-warehousing

environment:

- EUREKA\_SERVER\_ADDRESS=core-eureka

- DB\_INSTANCE=db-mysql

wms-statistics:

image: registry.cn-hangzhou.aliyuncs.com/imcloudep/wms-statistics

environment:

- EUREKA\_SERVER\_ADDRESS=core-eureka

- DB\_INSTANCE=db-mysql

service-crm:

image: registry.cn-hangzhou.aliyuncs.com/imcloudep/service-crm

environment:

- EUREKA\_SERVER\_ADDRESS=core-eureka

- DB\_INSTANCE=db-mysql

work-calendar:

image: registry.cn-hangzhou.aliyuncs.com/imcloudep/work-calendar

environment:

- EUREKA\_SERVER\_ADDRESS=core-eureka

- DB\_INSTANCE=db-mysql

networks:

default:

external:

name: imcloudep-external

[root@imcloud dockercompose-function]#

登录阿里云镜像仓库站点

[root@imcloud ~]# docker login registry.cn-hangzhou.aliyuncs.com

Username: jimaisi2017

Password:

Login Succeeded

[root@imcloud ~]#

镜像的启动顺序，基础类镜像容器-->核心类镜像容器-->功能类镜像容器

(基础类镜像容器先前已启动好了，下在分别启动核心类和功能类镜像容器)

[root@imcloud dockercompose-core]# pwd

/opt/imcloudep/dockercompose-core

//启动核心镜像容器

[root@imcloud dockercompose-core]# docker-compose up -d

Creating dockercomposecore\_core-eureka\_1 ... done

Creating dockercomposecore\_core-config\_1 ... done

[root@imcloud dockercompose-core]# cd ../dockercompose-function

[root@imcloud dockercompose-function]# pwd

/opt/imcloudep/dockercompose-function

//启动功能性镜像容器

[root@imcloud dockercompose-function]# docker-compose up -d

Creating dockercomposefunction\_core-apigateway\_1 ... done

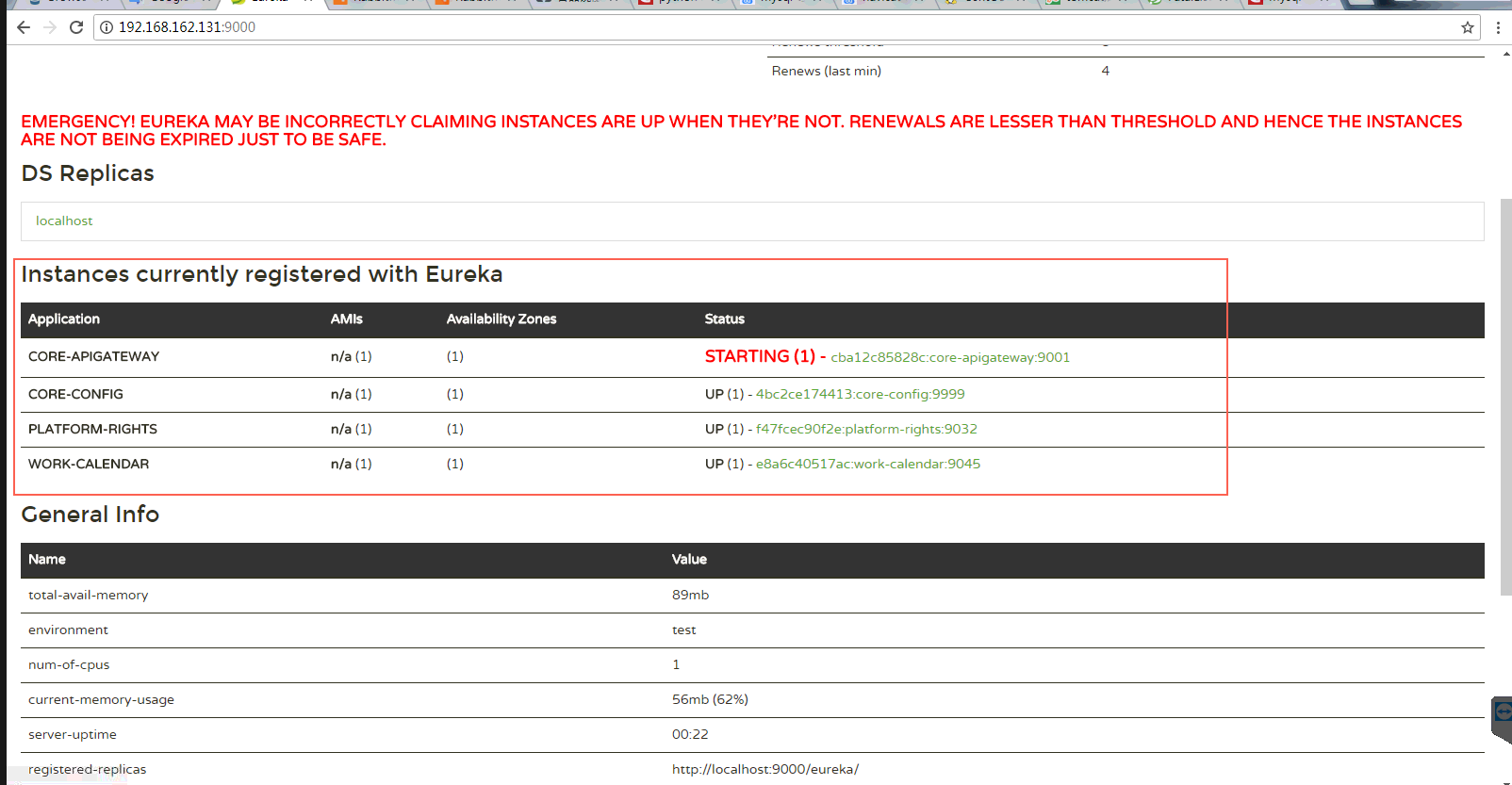
Creating dockercomposefunction\_platform-rights\_1 ... done

Creating dockercomposefunction\_work-calendar\_1 ... done

[root@imcloud dockercompose-function]#

说明：由于我的电脑内存资源不足，测试时并没有跑全部功能类微服务，只跑了3个。

登陆微服务注册和发现中心，查看服务注册运行情况(通过http://{IP}:9000来访问)



注：通过上图，可以看到微服务已经部署成功。

前端nginx布署：

安装nginx软件包

[root@imcloud yum.repos.d]# yum -y install nginx

Loaded plugins: fastestmirror

Loading mirror speeds from cached hostfile

\* base: mirrors.aliyun.com

\* centosplus: mirrors.aliyun.com

\* extras: mirrors.aliyun.com

\* updates: mirrors.aliyun.com

No package nginx available.

Error: Nothing to do

[root@imcloud yum.repos.d]#

注：安装时提示：Error: Nothing to do；说明没有配置epel 源。

[root@imcloud yum.repos.d]# yum list | grep epel-release

epel-release.noarch 7-9 extras

[root@imcloud yum.repos.d]#

[root@imcloud yum.repos.d]# yum -y install epel-release //安装epel源

[root@imcloud yum.repos.d]# yum install nginx -y //安装nginx

编辑/etc/nginx/nginx.conf配置文件

root@imcloud imcloudep]# vim /etc/nginx/nginx.conf

//配置如下：

[root@imcloud imcloudep]# cat /etc/nginx/nginx.conf

# For more information on configuration, see:

# \* Official English Documentation: http://nginx.org/en/docs/

# \* Official Russian Documentation: http://nginx.org/ru/docs/

user nginx;

worker\_processes auto;

error\_log /var/log/nginx/error.log;

pid /run/nginx.pid;

# Load dynamic modules. See /usr/share/nginx/README.dynamic.

include /usr/share/nginx/modules/\*.conf;

events {

worker\_connections 1024;

}

http {

log\_format main '$remote\_addr - $remote\_user [$time\_local] "$request" '

'$status $body\_bytes\_sent "$http\_referer" '

'"$http\_user\_agent" "$http\_x\_forwarded\_for"';

access\_log /var/log/nginx/access.log main;

sendfile on;

tcp\_nopush on;

tcp\_nodelay on;

keepalive\_timeout 65;

types\_hash\_max\_size 2048;

include /etc/nginx/mime.types;

default\_type application/octet-stream;

# Load modular configuration files from the /etc/nginx/conf.d directory.

# See http://nginx.org/en/docs/ngx\_core\_module.html#include

# for more information.

include /etc/nginx/conf.d/\*.conf;

server {

listen 80 default\_server; //服务监听端口

listen [::]:80 default\_server;

server\_name \_;

root /usr/share/nginx/html/nginx; //URL的根对应的文件系统路径

# Load configuration files for the default server block.

include /etc/nginx/default.d/\*.conf;

location /gateway/ {

proxy\_pass http://localhost:9001/; //匹配到gateway的，走本地微服务网关

}

location /platform {

}

location / {

}

error\_page 404 /404.html;

location = /40x.html {

}

error\_page 500 502 503 504 /50x.html;

location = /50x.html {

}

}

# Settings for a TLS enabled server.

#

# server {

# listen 443 ssl http2 default\_server;

# listen [::]:443 ssl http2 default\_server;

# server\_name \_;

# root /usr/share/nginx/html;

#

# ssl\_certificate "/etc/pki/nginx/server.crt";

# ssl\_certificate\_key "/etc/pki/nginx/private/server.key";

# ssl\_session\_cache shared:SSL:1m;

# ssl\_session\_timeout 10m;

# ssl\_ciphers HIGH:!aNULL:!MD5;

# ssl\_prefer\_server\_ciphers on;

#

# # Load configuration files for the default server block.

# include /etc/nginx/default.d/\*.conf;

#

# location / {

# }

#

# error\_page 404 /404.html;

# location = /40x.html {

# }

#

# error\_page 500 502 503 504 /50x.html;

# location = /50x.html {

# }

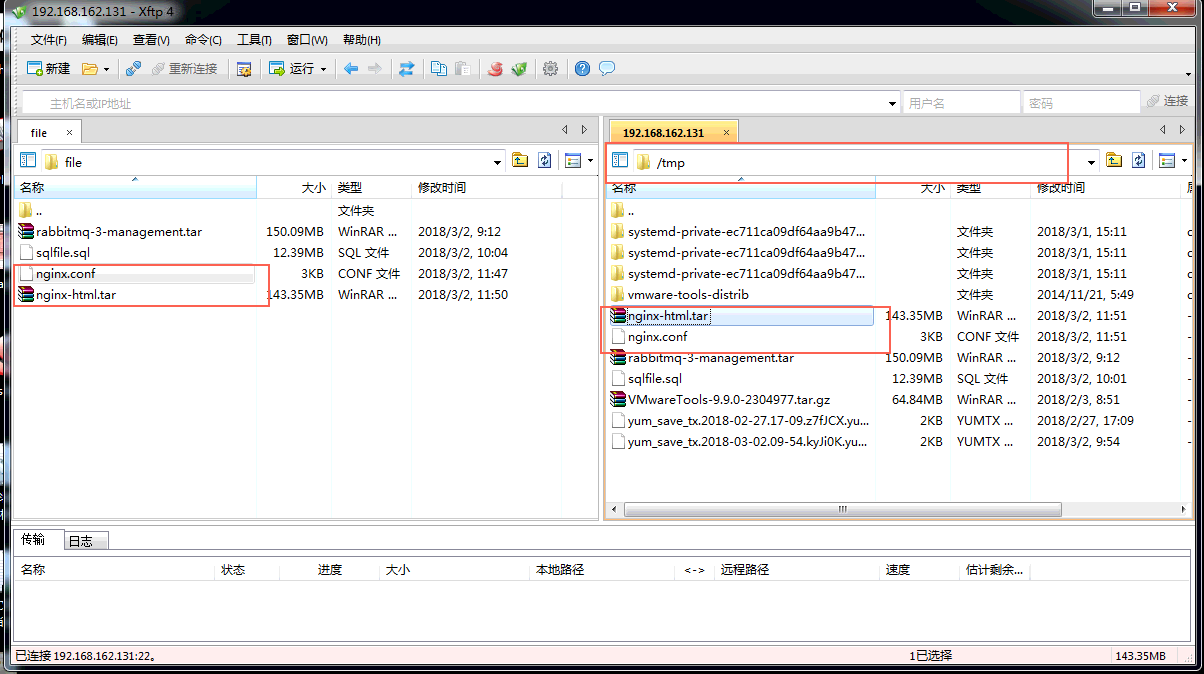
# }

}

[root@imcloud imcloudep]#

注：在nginx.conf配置文件中,一个location指令，对应一个服务目录。

上传项目前端代码到服务器/tmp目录下：



解压缩nginx-html.tar 到 /usr/share/nginx/html/ 目录下：

[root@imcloud imcloudep]# cd /usr/share/nginx/html/

[root@imcloud html]# tar xf /tmp/nginx-html.tar

[root@imcloud html]# ll

total 24

-rw-r--r-- 1 root root 3650 Oct 18 16:08 404.html

-rw-r--r-- 1 root root 3693 Oct 18 16:08 50x.html

-rw-r--r-- 1 root root 3700 Oct 18 16:08 index.html

drwxr-xr-x 28 root root 4096 Mar 2 10:17 nginx

-rw-r--r-- 1 root root 368 Oct 18 16:08 nginx-logo.png

-rw-r--r-- 1 root root 2811 Oct 18 16:08 poweredby.png

[root@imcloud html]#

启动nginx服务并设置开机自动启动：

[root@imcloud html]# systemctl status nginx && systemctl enable nginx

[root@imcloud html]# ss -antp | grep :80 //确认服务已监听在80端口

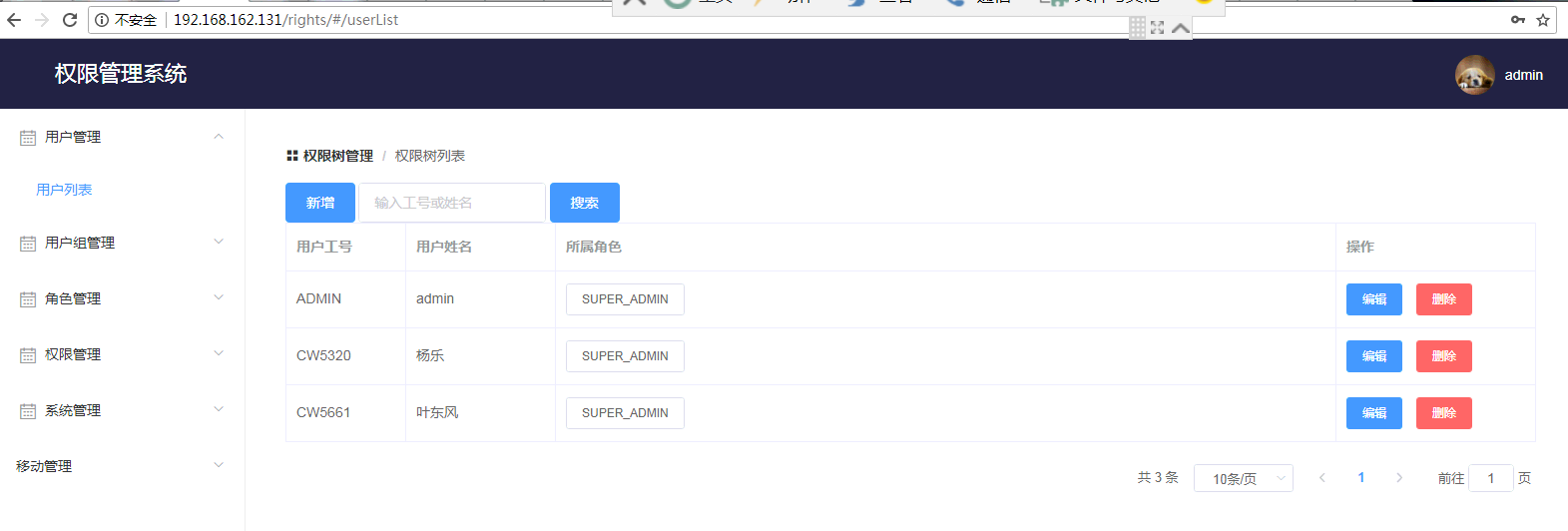
LISTEN 0 128 \*:80 \*:\* users:(("nginx",pid=57778,fd=6),("nginx",pid=57777,fd=6))

LISTEN 0 128 :::80 :::\* users:(("nginx",pid=57778,fd=7),("nginx",pid=57777,fd=7))

[root@imcloud html]#

通过浏览器访问验证服务：

登录权限管理系统：通过http://{IP}/rights/访问



登录认证中心后台管理系统：通过http://{IP}/platform/访问



至此，整个单服务器环境下IMCloud EP 搭建部署完毕。

多服务器（集群）环境搭建

需求

1. 两台节点主机

A : 192.168.168.131 （主机名：master.cluster.cc，作为管理节点）

B : 192.168.168.132 （主机名：slave.cluster.cc，作为工作节点 ）

2. 操作系统centos7.4(1708)

1．基础环境部署（参考前面系统环境准备）

2．安装Docker-CE、Docker-Compose （参考前面单服务器环境部署）

3．启动firewalld动态防火墙，并在主机之间打开以下协议和端口：

* TCP端口2377用于集群管理通信
* TCP和UDP端口7946用于节点之间的通信
* UDP端口4789用于覆盖网络通信
* TCP端口2375 用于docker engine的远程管理

3.1 开启docker engine的远程管理，分别在master 和 slave 节点执行下面命令

[root@master ~]# vim /etc/docker/daemon.json

//内容如下：

{"registry-mirrors": ["https://udodpz68.mirror.aliyuncs.com"],

"hosts": ["tcp://192.168.168.131:2375","unix:///var/run/docker.sock"]

}

[root@slave ~]# vim /etc/docker/daemon.json

//内容如下：

{"registry-mirrors": ["https://udodpz68.mirror.aliyuncs.com"],

"hosts": ["tcp://192.168.168.132:2375","unix:///var/run/docker.sock"]

}

[root@slave ~]#

为了便于协议和端口的管理，我们将上述自定义为firewalld的一个服务：

[root@master services]# cd /usr/lib/firewalld/services/

[root@master services]# cp -a ssh.xml swarm.xml

[root@master services]# cp -a ssh.xml swarm.xml

//配置如下：

<?xml version="1.0" encoding="utf-8"?>

<service>

<short>SWARM</short>

<description>Secure Shell (SSH) is a protocol for logging into and executing commands on remote machines. It provides secure encrypted communications. If you plan on accessing your machine remotely via SSH over a firewalled interface, enable this option. You need the openssh-server package installed for this option to be useful.</description>

<port protocol="tcp" port="2377"/>

<port protocol="tcp" port="7946"/>

<port protocol="udp" port="7946"/>

<port protocol="udp" port="4789"/>

<port protocol="tcp" port="50"/>

<port protocol="tcp" port="2375"/>

<port protocol="tcp" port="2376"/>

</service>

[root@master services]#

[root@master services]# scp - a swarm.xml 192.168.168.132:/usr/lib/firewalld/services/

[root@master services]# systemctl enable firewalld

[root@master services]# systemctl start firewalld

[root@master services]# firewall-cmd --permanent --add-service=swarm

[root@master services]# firewall-cmd --reload

[root@master services]#

[root@slave yum.repos.d]# systemctl enable firewalld

[root@slave yum.repos.d]# systemctl start firewalld.service

[root@slave yum.repos.d]# firewall-cmd --permanent --add-service=swarm

[root@slave yum.repos.d]# firewall-cmd –reload

4. 创建一个swarm群集

4．1 在管理节点上执行以下命令

[root@master dockercompose-base]# docker swarm init --advertise-addr 192.168.168.131

Swarm initialized: current node (ktn3i4fwibkkvj30x2htzg4l7) is now a manager.

To add a worker to this swarm, run the following command:

docker swarm join --token SWMTKN-1-2c2nbxlfxbppbpb33fdbx6cnhu2tj1fafbmacfeymf0zr2ze3g-8smu8qas0schdqto0syci0q0c 192.168.168.131:2377

To add a manager to this swarm, run 'docker swarm join-token manager' and follow the instructions.

[root@master dockercompose-base]#

注：该--advertise-addr标志配置管理器节点将其地址发布为192.168.168.131。群中的其他节点必须能够访问该IP地址。输出包括将新节点加入群集的命令

4．2 运行docker info 查看当前群集的当前状态

[root@master ~]# docker info

4．3 运行docker node ls命令查看有关节点的信息：

[root@master ~]# docker node ls

ID HOSTNAME STATUS AVAILABILITY MANAGER STATUS

ktn3i4fwibkkvj30x2htzg4l7 \* master.cluster.cc Ready Active Leader

[root@master ~]#

注：在\*旁边的节点ID表明当前连接此节点上。Docker Engine swarm模式会自动为机器主机名称命名节点。

5．添加节点到swarm群集

5．1 在slave节点上运行第4步骤 docker swarm init 输出产生的命令，以创建一个加入现有群集的工作节点：

[root@slave yum.repos.d]# docker swarm join --token SWMTKN-1-2c2nbxlfxbppbpb33fdbx6cnhu2tj1fafbmacfeymf0zr2ze3g-8smu8qas0schdqto0syci0q0c 192.168.168.131:2377

This node joined a swarm as a worker.

[root@slave yum.repos.d]#

如果您没有可用的命令，则可以在管理器节点上运行以下命令以检索工作节点的加入命令：

[root@master ~]# docker swarm join-token worker

To add a worker to this swarm, run the following command:

docker swarm join --token SWMTKN-1-2c2nbxlfxbppbpb33fdbx6cnhu2tj1fafbmacfeymf0zr2ze3g-8smu8qas0schdqto0syci0q0c 192.168.168.131:2377

[root@master ~]#

5．2 在master管理节点上运行 docker node ls 命令以查看工作节点：

[root@master ~]# docker node ls

ID HOSTNAME STATUS AVAILABILITY MANAGER STATUS

ktn3i4fwibkkvj30x2htzg4l7 \* master.cluster.cc Ready Active Leader

ieiqnk6onimgltkaoj156ofja slave.cluster.cc Ready Active

[root@master ~]#

5．为了实现跨主机的docker容器通信，创建overlay网络

[root@master ~]# docker network create \

--driver overlay \

--opt encrypted \

springcloud-overlay

nfxneb6tbastrg77m249gaz87

[root@master ~]#

6. 在mastert管理节点上构建core-config微服务镜像

core-config微服务在整个springCloud框架中，作为其它微服务的配置中心，其它微服务将从core-config获取相应的配置文件。由于不同的项目，配置文件不同，通过在core-config中的application.yml配置不同的git仓库地址，实现拉取不同的配置文件。因此每个项目需要对core-config微服务单独构建镜像。

6.1 创建用于构建core-config镜像的目录

[root@master ~]# mkdir -p /opt/imcloudep/core-config

6.2 根据项目需要编辑core-config.jar包中application.yml文件中git仓库地址

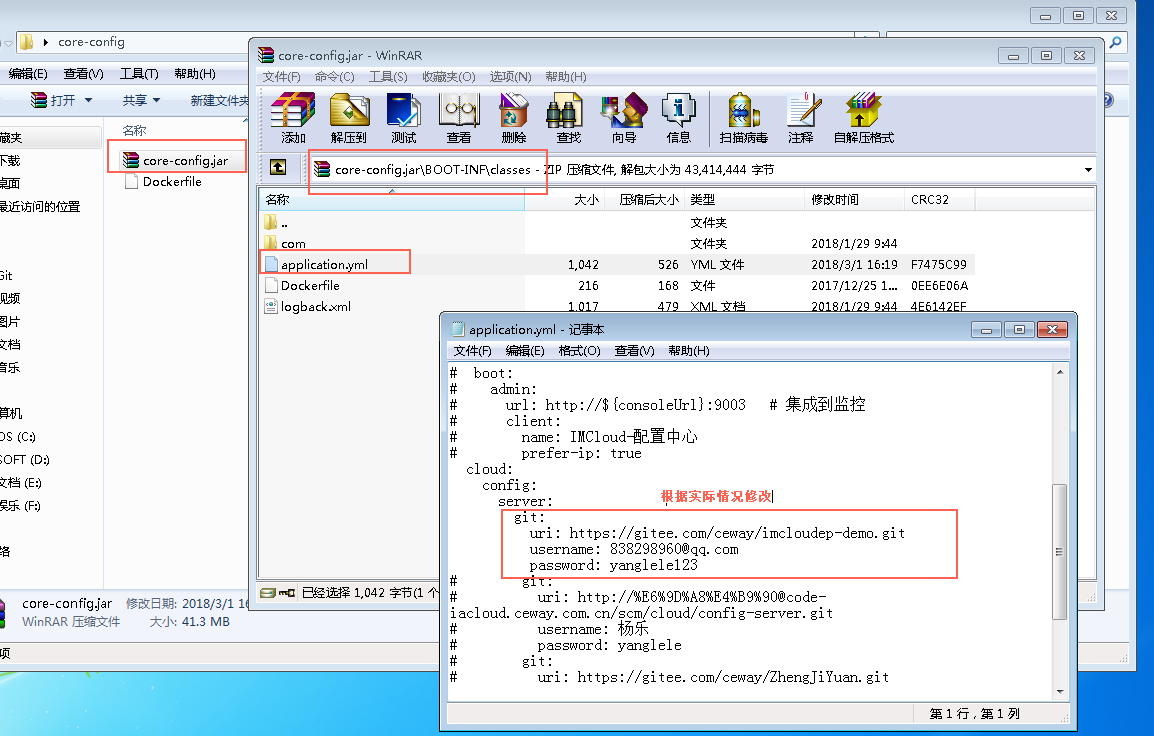
(1) 通过压缩软件（比如：winrar、7-Zip）打开core-config.jar包;

(2) 定位到BOOT-INF\classes\下的application.yml文件;

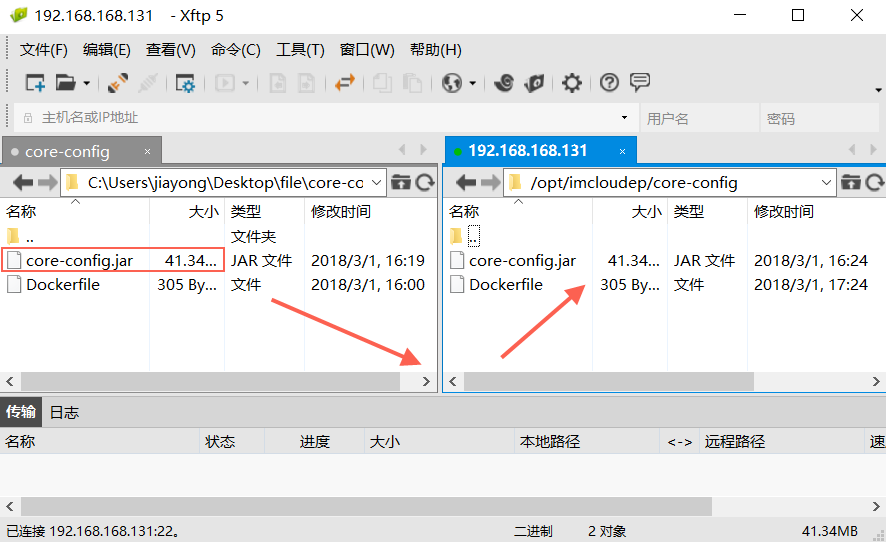
(3) 用记事本或其它文本编辑工具打开application.yml文件;

(4) 根据实际项目修改git仓库地址及用户名和密码.

操作如下图:



6.3 通过xftp软件将编辑好的core-config.jar包上传到master管理节点



6.4 编辑Dockerfile文件,构建imcloudep/core-config镜像

[root@master core-config]# vim Dockerfile

//文件内容如下:

FROM java:8

ENV TZ=Asia/Shanghai

RUN ln -snf /usr/share/zoneinfo/$TZ /etc/localtime && echo $TZ > /etc/timezone

VOLUME /tmp

ADD core-config.jar core-config.jar

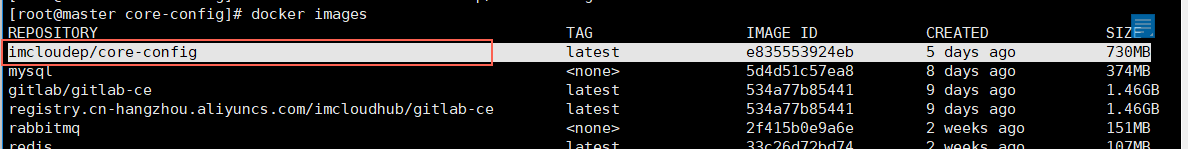
RUN bash -c 'touch /core-config.jar'

EXPOSE 9999

ENTRYPOINT ["java","-Djava.security.egd=file:/dev/./urandom","-jar","/core-config.jar"]

[root@master core-config]# docker build -t imcloudep/core-config .

查看构建镜像是否成功:



7. 在master管理节点上,从阿里云镜像仓库拉取其它微服务镜像

//登录阿里云镜像私有仓库

[root@master ~]# docker login registry.cn-hangzhou.aliyuncs.com/imcloudep

Username: jimaisi2017

Password:

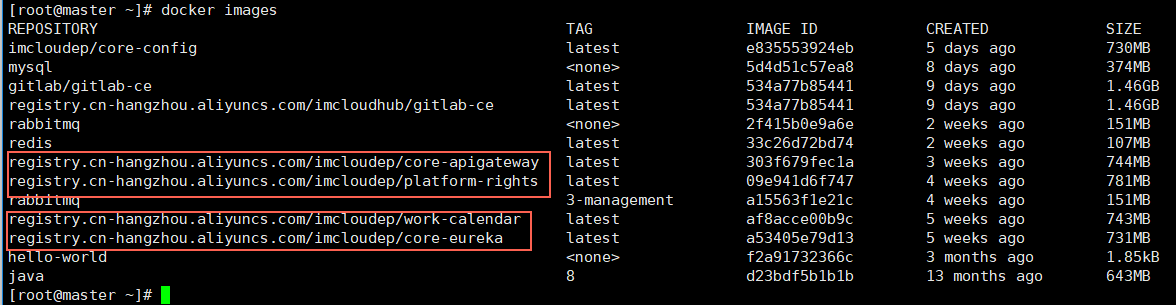
Login Succeeded

[root@master ~]#

//通过下面命令拉取微服务镜像

[root@master ~]# docker pull registry.cn-hangzhou.aliyuncs.com/imcloudep/core-eureka

由于我的测试机内存资源限制,这里仅拉取几个关键镜像,以验证服务:



8. 微服务的部署,可以通过docker service 或者 docker stack 的方式来部署

docker service 指令一次只能创建一个服务,示例代码如下:

//示例代码

[root@master ~]# docker service create \

--name portainer \

--publish 7000:9000 \

--replicas=1 \

--constraint 'node.role == manager' \

--mount type=bind,src=//var/run/docker.sock,dst=/var/run/docker.sock \

--mount type=bind,src=//opt/portainer,dst=/data \

portainer/portainer \

-H unix:///var/run/docker.sock

docker stack 指令根据 \*.yml 文件来部署一个服务栈,可以一次创建多个服务,方便服务的管理.示例如下:

[root@master ~]# docker stack deploy \

-c /opt/imcloudep/dockerswarm/dockercompose-base/docker-compose.yml \

imcloudep-base

[root@master ~]#

9. 在MASTER管理节点部署PORTAINER管理SWARM群集

[root@master dockercompose-base]# mkdir /opt/portainer

[root@master dockercompose-base]# docker service create \

--name portainer \

--publish 7000:9000 \

--replicas=1 \

--constraint 'node.role == manager' \

--mount type=bind,src=//var/run/docker.sock,dst=/var/run/docker.sock \

--mount type=bind,src=//opt/portainer,dst=/data \

portainer/portainer \

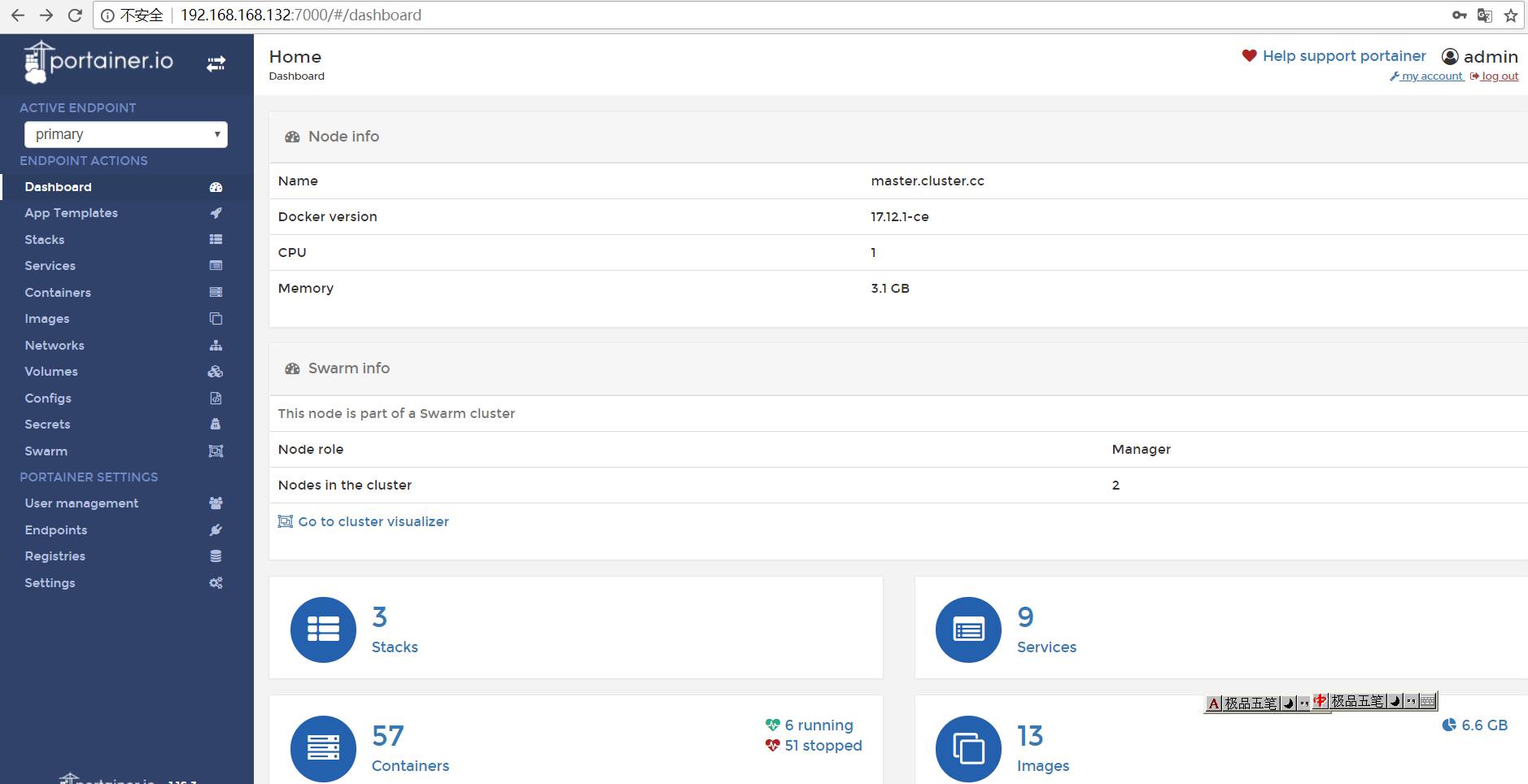
-H unix:///var/run/docker.sock

[root@master dockercompose-base]#

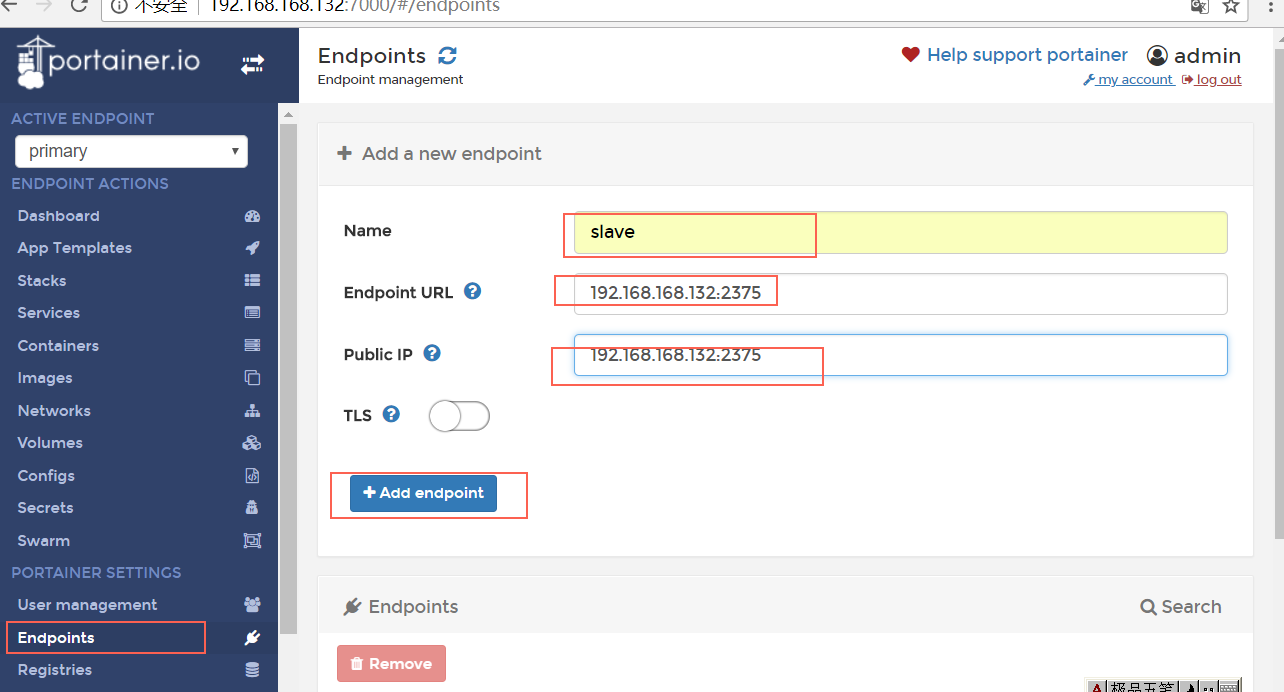
通过浏览器登录PORTAINER，访问地址：http://{IP}:7000



登录成功界面：



将slave工作节点加入PORTAINER管理



10. 编辑dockercompose.yml文件

[root@master ~]# mkdir -p /opt/imcloudep/dockerswarm/

[root@master ~]# cd /opt/imcloudep/dockerswarm/

[root@master dockerswarm]# mkdir dockercompose-base dockercompose-core

[root@master dockerswarm]# mkdir dockercompose-function

[root@master dockerswarm]# touch dockercompose-base/docker-compose.yml

[root@master dockerswarm]# touch dockercompose-core/docker-compose.yml

[root@master dockerswarm]# touch dockercompose-function/docker-compose.yml

[root@master dockerswarm]# vim dockercompose-base/docker-compose.yml

//配置内容如下：

version: '3'

services:

db-mysql:

image: mysql:latest #lkjlkfaj

volumes:

- /opt/mysql/data:/var/lib/mysql

- /opt/mysql/conf:/etc/mysql/conf.d

- /etc/localtime:/etc/localtime

environment:

- MYSQL\_ROOT\_PASSWORD=Hello!!!123

networks:

- springcloud-overlay

ports:

- "3306:3306"

command: mysqld --character-set-server=utf8 --collation-server=utf8\_unicode\_ci

deploy:

placement:

constraints: [ node.role == worker ]

cache-redis:

image: redis:latest

ports:

- "6379:6379"

volumes:

- /opt/redis/redis.conf:/usr/local/etc/redis/redis.conf

- /opt/redis/data:/data

networks:

- springcloud-overlay

command: redis-server /usr/local/etc/redis/redis.conf

deploy:

placement:

constraints: [ node.role == worker ]

imcloudep-rabbitmq:

image: rabbitmq:3-management

hostname: imcloudep-rabbitmq

environment:

- RABBITMQ\_DEFAULT\_USER=root

- RABBITMQ\_DEFAULT\_PASS=root

volumes:

- /opt/rabbitmq:/var/lib/rabbitmq

networks:

- springcloud-overlay

ports:

- "15672:15672"

deploy:

placement:

constraints: [ node.role == worker ]

# gitlab-ce:

# image: 'gitlab/gitlab-ce:latest'

# restart: always

# hostname: 'gitlab.test.com'

# environment:

# GITLAB\_OMNIBUS\_CONFIG: |

# external\_url 'http://gitlab.test.com:9090'

# gitlab\_rails['gitlab\_shell\_ssh\_port'] = 2224

# networks:

# - springcloud-overlay

# ports:

# - '9090:9090'

# - '2224:22'

# volumes:

# - '/opt/gitlab/config:/etc/gitlab'

# - '/opt/gitlab/logs:/var/log/gitlab'

# - '/opt/gitlab/data:/var/opt/gitlab'

# deploy:

# placement:

# constraints: [ node.hostname == slave.cluster.cc ]

networks:

springcloud-overlay:

external:

name: springcloud-overlay

[root@master dockercompose-base]# vim dockercompose-core/docker-compose.yml

//配置内容如下：

version: '3'

services:

core-eureka:

image: registry.cn-hangzhou.aliyuncs.com/imcloudep/core-eureka

networks:

- springcloud-overlay

ports:

- "9000:9000"

core-config:

image: imcloudep/core-config

networks:

- springcloud-overlay

ports:

- "9999:9999"

environment:

- EUREKA\_SERVER\_ADDRESS=core-eureka

networks:

springcloud-overlay:

external:

name: springcloud-overlay

[root@master dockerswarm]# vim dockercompose-function/docker-compose.yml

//配置内容如下：

version: '3'

services:

core-apigateway:

image: registry.cn-hangzhou.aliyuncs.com/imcloudep/core-apigateway

networks:

- springcloud-overlay

ports:

- "9001:9001"

environment:

- EUREKA\_SERVER\_ADDRESS=core-eureka

- DB\_INSTANCE=db-mysql

# base-push:

# image: registry.cn-hangzhou.aliyuncs.com/imcloudep/base-push

# networks:

# - springcloud-overlay

# ports:

# - "9008:9008"

# environment:

# - EUREKA\_SERVER\_ADDRESS=core-eureka

#

# ehr-profile:

# image: registry.cn-hangzhou.aliyuncs.com/imcloudep/ehr-profile

# networks:

# - springcloud-overlay

# environment:

# - EUREKA\_SERVER\_ADDRESS=core-eureka

# - DB\_INSTANCE=db-mysql

# ehr-signin:

# image: registry.cn-hangzhou.aliyuncs.com/imcloudep/ehr-signin

# networks:

# - springcloud-overlay

# environment:

# - EUREKA\_SERVER\_ADDRESS=core-eureka

# - DB\_INSTANCE=db-mysql

#

# mms-materialsweb:

# image: registry.cn-hangzhou.aliyuncs.com/imcloudep/mms-materialsweb

# networks:

# - springcloud-overlay

# environment:

# - EUREKA\_SERVER\_ADDRESS=core-eureka

# - DB\_INSTANCE=db-mysql

platform-rights:

image: registry.cn-hangzhou.aliyuncs.com/imcloudep/platform-rights

networks:

- springcloud-overlay

environment:

- EUREKA\_SERVER\_ADDRESS=core-eureka

- DB\_INSTANCE=db-mysql

# pms-purchase:

# image: registry.cn-hangzhou.aliyuncs.com/imcloudep/pms-purchase

# networks:

# - springcloud-overlay

# environment:

# - EUREKA\_SERVER\_ADDRESS=core-eureka

# - DB\_INSTANCE=db-mysql

# pms-supplier:

# image: registry.cn-hangzhou.aliyuncs.com/imcloudep/pms-supplier

# networks:

# - springcloud-overlay

# environment:

# - EUREKA\_SERVER\_ADDRESS=core-eureka

# - DB\_INSTANCE=db-mysql

# pms-app:

# image: registry.cn-hangzhou.aliyuncs.com/imcloudep/pms-app

# networks:

# - springcloud-overlay

# environment:

# - EUREKA\_SERVER\_ADDRESS=core-eureka

# - DB\_INSTANCE=db-mysql

#

# aps-plan:

# image: registry.cn-hangzhou.aliyuncs.com/imcloudep/aps-plan

# networks:

# - springcloud-overlay

# environment:

# - EUREKA\_SERVER\_ADDRESS=core-eureka

# - DB\_INSTANCE=db-mysql

# aps-order:

# image: registry.cn-hangzhou.aliyuncs.com/imcloudep/aps-order

# networks:

# - springcloud-overlay

# environment:

# - EUREKA\_SERVER\_ADDRESS=core-eureka

# - DB\_INSTANCE=db-mysql

# aps-technology:

# image: registry.cn-hangzhou.aliyuncs.com/imcloudep/aps-technology

# networks:

# - springcloud-overlay

# environment:

# - EUREKA\_SERVER\_ADDRESS=core-eureka

# - DB\_INSTANCE=db-mysql

# aps-workorder:

# image: registry.cn-hangzhou.aliyuncs.com/imcloudep/aps-workorder

# networks:

# - springcloud-overlay

# environment:

# - EUREKA\_SERVER\_ADDRESS=core-eureka

# - DB\_INSTANCE=db-mysql

#

# wfs-common:

# image: registry.cn-hangzhou.aliyuncs.com/imcloudep/wfs-common

# networks:

# - springcloud-overlay

# environment:

# - EUREKA\_SERVER\_ADDRESS=core-eureka

# - DB\_INSTANCE=db-mysql

# wf-special:

# image: registry.cn-hangzhou.aliyuncs.com/imcloudep/wf-special

# networks:

# - springcloud-overlay

# environment:

# - EUREKA\_SERVER\_ADDRESS=core-eureka

# - DB\_INSTANCE=db-mysql

#

# wms-management:

# image: registry.cn-hangzhou.aliyuncs.com/imcloudep/wms-management

# networks:

# - springcloud-overlay

# environment:

# - EUREKA\_SERVER\_ADDRESS=core-eureka

# - DB\_INSTANCE=db-mysql

# wms-materials:

# image: registry.cn-hangzhou.aliyuncs.com/imcloudep/wms-materials

# networks:

# - springcloud-overlay

# environment:

# - EUREKA\_SERVER\_ADDRESS=core-eureka

# - DB\_INSTANCE=db-mysql

# wms-outbound:

# image: registry.cn-hangzhou.aliyuncs.com/imcloudep/wms-outbound

# networks:

# - springcloud-overlay

# environment:

# - EUREKA\_SERVER\_ADDRESS=core-eureka

# - DB\_INSTANCE=db-mysql

# wms-stock:

# image: registry.cn-hangzhou.aliyuncs.com/imcloudep/wms-stock

# networks:

# - springcloud-overlay

# environment:

# - EUREKA\_SERVER\_ADDRESS=core-eureka

# - DB\_INSTANCE=db-mysql

# wms-warehousing:

# image: registry.cn-hangzhou.aliyuncs.com/imcloudep/wms-warehousing

# networks:

# - springcloud-overlay

# environment:

# - EUREKA\_SERVER\_ADDRESS=core-eureka

# - DB\_INSTANCE=db-mysql

# wms-statistics:

# image: registry.cn-hangzhou.aliyuncs.com/imcloudep/wms-statistics

# networks:

# - springcloud-overlay

# environment:

# - EUREKA\_SERVER\_ADDRESS=core-eureka

# - DB\_INSTANCE=db-mysql

#

# service-crm:

# image: registry.cn-hangzhou.aliyuncs.com/imcloudep/service-crm

# networks:

# - springcloud-overlay

# environment:

# - EUREKA\_SERVER\_ADDRESS=core-eureka

# - DB\_INSTANCE=db-mysql

work-calendar:

image: registry.cn-hangzhou.aliyuncs.com/imcloudep/work-calendar

networks:

- springcloud-overlay

environment:

- EUREKA\_SERVER\_ADDRESS=core-eureka

- DB\_INSTANCE=db-mysql

networks:

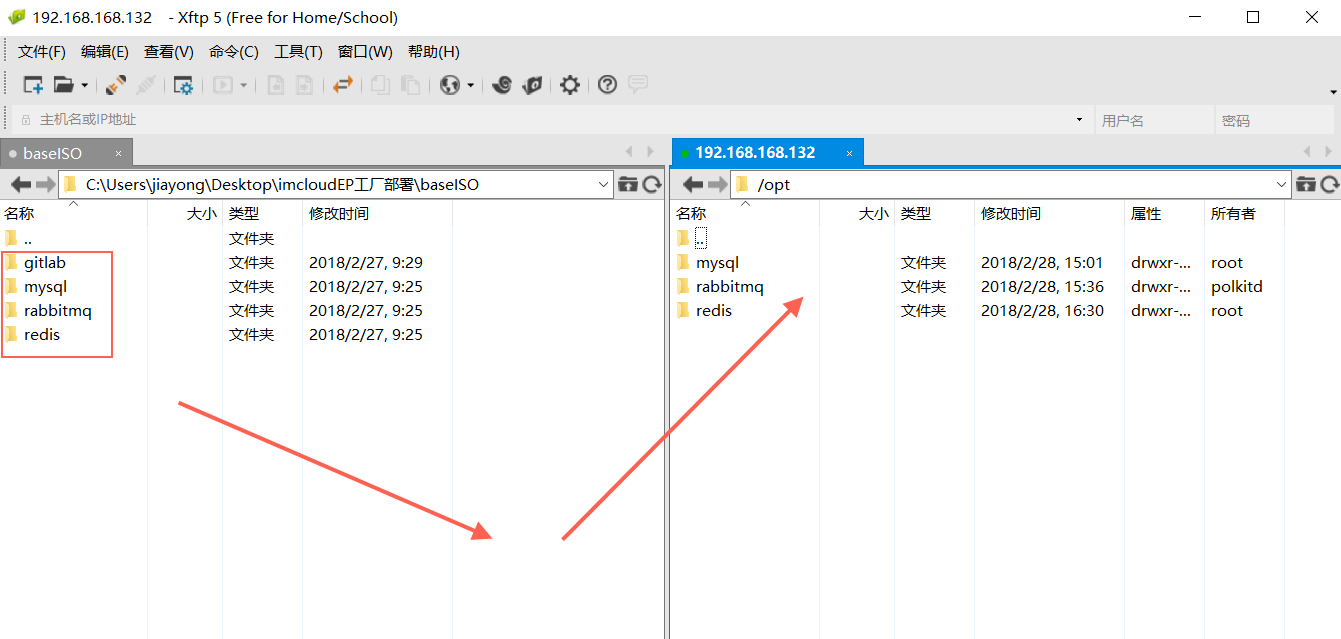
springcloud-overlay:

external:

name: springcloud-overlay

[root@master dockerswarm]#

11. 在slave工作节点上创建运行基础容器的挂载目录



12. 通过docker stack 来部署容器