

1添加普通用户

```
adduser user
passwd user
#输入两次user的密码
su user
```

设置user到root的sudoers目录

```
su
visudo #另外的版本vim /etc/sudoers
```

```
## Allow root to run any commands anywhere
root    ALL=(ALL)    ALL
user    ALL=(ALL)    ALL
```

保存即可。

1.1pip无法找到

安装即可

```
easy_install pip
或者
pip3 install --upgrade pip
```

2安装docker

```
yum remove docker \
docker-client \
docker-client-latest \
docker-common \
docker-latest \
docker-latest-logrotate \
docker-logrotate \
docker-engine

yum install -y yum-utils

yum-config-manager \
--add-repo \
https://mirrors.aliyun.com/docker-ce/linux/centos/docker-ce.repo

yum makecache fast

yum install docker-ce docker-ce-cli containerd.io

systemctl start docker # 启动Docker
docker version # 查看当前版本号，是否启动成功
```

```
systemctl enable docker # 设置开机自启动
```

2.1 阿里云源

```
sudo mkdir -p /etc/docker
```

```
sudo tee /etc/docker/daemon.json <<-'EOF'
{
  "features": {"buildkit": false},
  "registry-mirrors": ["https://*****.mirror.aliyuncs.com"]
}
EOF
```

```
sudo systemctl daemon-reload
sudo systemctl restart docker
```

https://****.mirror.aliyuncs.com需要替换成自己的阿里云账号下的镜像加速链接

The screenshot shows the Alibaba Cloud Container Registry console. The left sidebar has a menu with '容器镜像服务' (Container Registry) and '镜像加速器' (Mirror Accelerator). The main content area shows the '镜像加速器' (Mirror Accelerator) page. A red box highlights the '加速器地址' (Accelerator Address) field, which contains a URL like 'https://****.mirror.aliyuncs.com'. A red arrow points to this field with the text '直接百度搜索阿里云,支付宝扫码验证登录即可 选择控制台,搜索镜像即可' (Directly search for Alibaba Cloud on Baidu, log in with Alipay QR code verification, select the console, and search for the mirror). Another red box highlights the '操作文档' (Operation Document) section, which contains instructions for installing/upgrading Docker and configuring the mirror accelerator. A red arrow points to the '操作文档' section with the text '每个人的连接都是不同的' (Everyone's connection is different). A third red box highlights the '操作文档' section, which contains the same configuration steps as shown in the code block above. A red arrow points to this box with the text '直接复制到远程控制器终端即可' (Directly copy to the remote controller terminal). The bottom right corner of the screenshot has the text 'CSDN @不教而诛'.

2.2 docker卸载

```
yum remove docker-ce docker-ce-cli containerd.io # 卸载依赖
rm -rf /var/lib/docker # 删除资源 . /var/lib/docker是docker的默认工作路径
```

3mysql

```
docker pull mysql:5.7
```

#创建mysql容器的命令

```
sudo docker run -d -p 3306:3306 -v /usr/local/mysql/conf:/etc/mysql/conf.d -v /usr/local/mysql/data:/var/lib/mysql -e MYSQL_ROOT_PASSWORD=123456 --name mysql mysql:5.7
```

#设置容器自启动

```
docker update --restart=always mysql
```

配置端口映射: -p 3306:3306 --name mysql, 将容器的3306端口映射到主机的3306端口

配置mysql数据卷挂载

1.-v /mydata/mysql/log:/var/log/mysql(日志文件挂载)

将容器中的日志文件夹/var/log/mysql挂载到主机对应的/mydata/mysql文件夹中

2.-v /mydata/mysql/data:/var/lib/mysql(数据文件挂载)

将容器中的数据文件夹/var/lib/mysql挂载到主机对应的/mydata/mysql/data文件夹中

3.-v /mydata/mysql/conf:/etc/mysql(配置文件挂载)

将容器的配置文件/etc/mysql挂载到主机对应的/mydata/mysql/conf文件夹中

注(这里所提的主机指的是当前的Linux主机)

配置用户

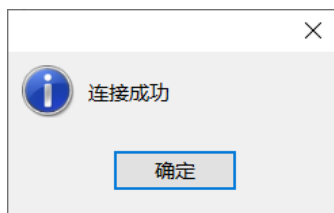
-e MYSQL_ROOT_PASSWORD=123456 设置初始化root用户的密码为123456

-d mysql:5.7 指定镜像资源

-d: 以后台方式运行实例

mysql:5.7: 指定用这个镜像来创建运行实例

navicat测试成功



3.1修改mysql的root密码

```
docker exec -it 容器ID /bin/bash
mysql -uroot -p
use mysql
SET PASSWORD FOR 'root' = PASSWORD('设置的密码');
exit
docker restart 容器ID
```

4Tomcat

```
cd /opt
mkdir docker_tomcat
cd docker_tomcat
mkdir webapps
mkdir logs
mkdir conf

docker pull tomcat:8
```

```
#第一次运行拷贝出配置文件
docker run -d -p 8080:8080 --name tomcat tomcat:8
docker ps
docker exec -it id /bin/bash
docker cp 01f6adb44435:/usr/local/tomcat/conf/server.xml
/opt/docker_tomcat/conf/server.xml
docker cp 01f6adb44435:/usr/local/tomcat/conf/tomcat-users.xml
/opt/docker_tomcat/conf/tomcat-users.xml
docker stop tomcat
docker rm tomcat

#第二次运行，挂载目录
docker run --name tomcat -p 8080:8080 \
-v /opt/docker_tomcat/webapps:/usr/local/tomcat/webapps \
-v /opt/docker_tomcat/conf/server.xml:/usr/local/tomcat/conf/server.xml \
-v /opt/docker_tomcat/conf/tomcat-users.xml:/usr/local/tomcat/conf/tomcat-
users.xml \
-v /opt/docker_tomcat/logs:/usr/local/tomcat/logs \
-d tomcat:8

docker cp cb4fff249ec5:/usr/local/tomcat/webapps.dist/* /opt/docker_tomcat


#设置容器自启动
docker update --restart=always tomcat
```

-d: 以后台方式运行
-p 8080:8080: 指定端口，映射形式为：主机端口(容器外部端口): docker 容器端口(tomcat的端口)
tomcat:8: 镜像名称，与上述拉取名称一致
--name tomcat1: 自定义容器名称
如果是大写的 -P，则会给主机随机分配端口

测试外网

[Home](#)
[Documentation](#)
[Configuration](#)
[Examples](#)
[Wiki](#)
[Mailing Lists](#)
[Find Help](#)

Apache Tomcat/8.5.73



If you're seeing this, you've successfully installed Tomcat. Congratulations!

Recommended Reading:
[Security Considerations How-To](#)
[Manager Application How-To](#)
[Clustering/Session Replication How-To](#)

[Server Status](#)
[Manager App](#)
[Host Manager](#)

Developer Quick Start

[Tomcat Setup](#)
[Realms & AAA](#)
[Examples](#)
[Servlet Specifications](#)

[First Web Application](#)
[JDBC DataSources](#)
[Tomcat Versions](#)

[Managing Tomcat](#)
[Documentation](#)
[Getting Help](#)

4.1不能访问主页需要在配置文件中加入代码

```
<Host name="localhost" appBase="webapps"
    unpackWARs="true" autoDeploy="true">
<Context docBase="/usr/local/tomcat/webapps/ruoyi-boot" path="/" reloadable="true" source=""/>
```

```
vim /opt/docker_tomcat/conf/server.xml
<Context docBase="/usr/local/tomcat/webapps/ruoyi-admin" path="/"
reloadable="true" source=""/>
```

4.2不挂载目录的情况下

HTTP状态 404 - 未找到

类型 状态报告

描述 源服务器未能找到目标资源的表示或者是不愿公开一个已经存在的资源表示。

Apache Tomcat/8.5.73

因为docker中tomcat文件夹下，webapps为空，需要将webapps.dist内容复制进去。

```
docker exec -it 容器名 /bin/bash
```

```
[user@VM-0-8-centos ~]$ sudo docker exec -it e2c20118b7f8 /bin/bash
root@e2c20118b7f8:/usr/local/tomcat# ls
BUILDING.txt  LICENSE  README.md  RUNNING.txt  conf  logs  temp  webapps.dist
CONTRIBUTING.md  NOTICE  RELEASE-NOTES  bin  lib  native-jni-lib  webapps  work
root@e2c20118b7f8:/usr/local/tomcat# cd webapps
root@e2c20118b7f8:/usr/local/tomcat/webapps# ls
root@e2c20118b7f8:/usr/local/tomcat/webapps# ls -l
total 0
```

```
rm -rf webapps
mv webapps.dist webapps
```

再次测试即可。

5Redis

```
docker pull redis
```

以配置文件启动

```
mkdir /opt/docker_redis
```

```
cd /opt/docker_redis
```

```
wget http://download.redis.io/redis-stable/redis.conf
```

```
chmod 777 redis.conf
```

修改默认配置信息

```
vim /opt/docker_redis/redis.conf
```

```
sudo chmod 664 redis.conf
```

bind 127.0.0.1 通过#注释掉，解除本地连接限制，允许远程访问

```
67 # ~~~~~
68 #bind 127.0.0.1
69
```

protected-mode yes 默认no，保护模式，限制为本地访问，修改后解除保护模式

```
86 # are explicitly listed using the "bind" directive.
87 protected-mode no
88
```

daemonize yes 默认no 为不守护进程模式，修改为yes

```
222 # By default Redis does not run as a daemon. Use 'yes' if you need it.
223 # Note that Redis will write a pid file in /var/run/redis.pid when
224 daemonize yes
225
```

设置密码（建议设置，不设置有风险）

```
790 # requirepass foobared
791 requirepass 123456
792 # Command renaming (DEPRECATED).
```

持久化(可选)

```
1088
1089 appendonly yes
1090
```

```
docker run -p 6379:6379 --name redis -v /docker-
software/redis/redis.conf:/etc/redis/redis.conf -v /docker-
software/redis/data:/data -d redis redis-server /etc/redis/redis.conf --
appendonly yes
```

命令分析

-p 6379:6379 端口映射：前表示主机部分，：后表示容器部分。

-name redis 指定该容器名称，查看和进行操作都比较方便。

-v 挂载文件或目录：前表示主机部分，：后表示容器部分。

-d redis 表示后台启动redis

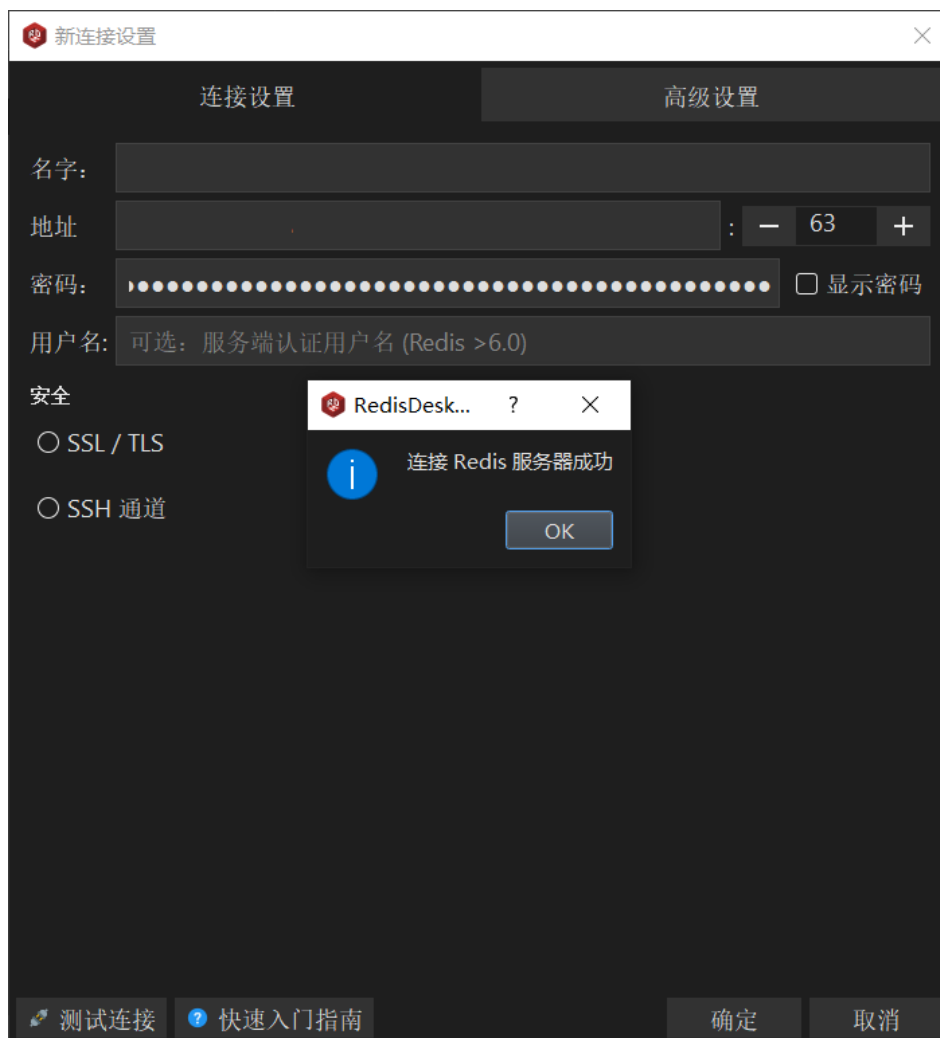
redis-server /etc/redis/redis.conf

以配置文件启动redis，加载容器内的conf文件，最终找到的是挂载的目录/usr/local/docker/redis.conf

-appendonly yes 开启redis 持久化

-requirepass 123456 设置密码为123456

用RDM测试



```
#设置容器自启动
docker update --restart=always redis
```

6Nginx

```
docker pull nginx
#先运行一次容器（为了拷贝配置文件）：
docker run -p 80:80 --name nginx -d nginx
docker exec -it nginx /bin/bash
```

将要复制出去的配置文件

```
d6fa12e96baf450471c554ec0e977b07cc4212e107b7140500505175005500c
[user@VM-0-8-centos docker_redis]$ sudo docker exec -it nginx /bin/bash
root@d6fa12e96baf:/# cd /etc/nginx/
root@d6fa12e96baf:/etc/nginx# ls
conf.d fastcgi_params mime.types modules nginx.conf scgi_params uwsgi_params
root@d6fa12e96baf:/etc/nginx#
```

拷贝到外面

```
mkdir /opt/docker_nginx
mkdir /opt/docker_nginx/conf
docker cp d6fa12e96baf:/etc/nginx/nginx.conf /opt/docker_nginx/conf/

docker stop nginx
docker rm nginx
```

```
docker run --name nginx -p 80:80 \
-v /opt/docker_nginx/conf/nginx.conf:/etc/nginx/nginx.conf \
-v /opt/docker_nginx/html:/usr/share/nginx/html \
-v /opt/docker_nginx/logs:/var/log/nginx \
-d nginx

#设置容器自启动
docker update --restart=always nginx
```

在外网访问。

403 Forbidden

nginx/1.21.5

7RabbitMQ

```
docker pull rabbitmq
```

7.1端口开放

如果在云服务上部署需在安全组开通一下端口：15672、5672、25672、61613、1883。

- 15672(UI页面通信口)
- 5672(client端通信口)
- 25672(server间内部通信口)
- 61613(stomp 消息传输)
- 1883(MQTT消息队列遥测传输)

7.2启动MQ安装management

```
docker run -d --name rabbit -e RABBITMQ_DEFAULT_USER=admin -e
RABBITMQ_DEFAULT_PASS=admin -p 15672:15672 -p 5672:5672 -p 25672:25672 -p
61613:61613 -p 1883:1883 rabbitmq:management
```

参数解释：本条命令包括安装web页面管理的 rabbitmq:management组件，账号和密码都为 admin；-p 后面参数表示公网IP地址的端口号对应容器内部的端口号。

```
#设置容器自启动
docker update --restart=always rabbit
```

rabbitmq默认**账号和密码**是guest，默认情况只能在localhost访问，所以我们需要通过刚才创建的admin用户进行登录。输入 <http://IP地址:15672> 即可完成访问，账号密码都为admin。



Username: *

Password: *

Login

RabbitMQ 3.9.11 Erlang 24.2

Refreshed 2023-03-08 08:47:12 Refresh every 5 seconds Virtual host All Cluster rabbit@413bd6df266e User admin Log out

OverviewConnectionsChannelsExchangesQueuesAdmin

Overview

Totals

Queued messages last minute ?

Currently idle

Message rates last minute ?

Currently idle

Global counts ?

Connections: 0Channels: 0Exchanges: 7Queues: 0Consumers: 0

Nodes

Name	File descriptors ?	Socket descriptors ?	Erlang processes	Memory ?	Disk space	Uptime	Info	Reset stats
rabbit@413bd6df266e	42 1048576 available	0 943629 available	394 1048576 available	129 MiB 799 MiB high watermark 31 MiB low watermark	31 GiB	4m 16s	basic disc 2 rss	This node All nodes

Churn statistics

Ports and contexts

Export definitions

Import definitions

HTTP API Server Docs Tutorials Community Support Community Slack Commercial Support Plugins GitHub Changelog

8ElasticSearch

```
docker network create es-net
```

```
docker pull elasticsearch:7.12.1
docker pull kibana:7.12.1
```

```
mkdir -p /opt/docker_es/config
mkdir -p /opt/docker_es/data
su
echo "http.host: 0.0.0.0" >> /opt/docker_es/config/elasticsearch.yml

chmod -R 777 /opt/docker_es/

docker run --name elasticsearch -p 9200:9200 -p 9300:9300 \
-e "discovery.type=single-node" \
-e ES_JAVA_OPTS="-Xms64m -Xmx128m" \
--privileged \
--network es-net \
-v
/opt/docker_es/config/elasticsearch.yml:/usr/share/elasticsearch/config/elasticsearch.yml \
-v /opt/docker_es/data:/usr/share/elasticsearch/data \
-v /opt/docker_es/plugins:/usr/share/elasticsearch/plugins \
-d elasticsearch:7.12.1

docker start 容器id
#设置容器自启动
```

```
docker update --restart=always elasticsearch
```

访问9200端口

```
{
  "name" : "0c96f19dc4d6",
  "cluster_name" : "elasticsearch",
  "cluster_uuid" : "YxG_oNGoStyoLzPSDZ_xcA",
  "version" : {
    "number" : "7.4.2",
    "build_flavor" : "default",
    "build_type" : "docker",
    "build_hash" : "2f90bbf7b93631e52bafb59b3b049cb44ec25e96",
    "build_date" : "2019-10-28T20:40:44.881551Z",
    "build_snapshot" : false,
    "lucene_version" : "8.2.0",
    "minimum_wire_compatibility_version" : "6.8.0",
    "minimum_index_compatibility_version" : "6.0.0-beta1"
  },
  "tagline" : "You Know, for Search"
}
```

8.1 kibana可视化

```
docker run --name kibana -e ELASTICSEARCH_HOSTS=http://43.136.129.168:9200 -p 5601:5601 -d kibana:7.4.2
```

访问IP:5601即可

9Nacos

9.1单机模式

```
docker pull nacos/nacos-server
```

```
mkdir -p /opt/docker_nacos/init.d /opt/docker_nacos/logs /opt/docker_nacos/data
touch /opt/docker_nacos/init.d/custom.properties
```

```
management.endpoints.web.exposure.include=*
```

查看版本

```
docker image inspect nacos/nacos-server:latest|grep -i version
```

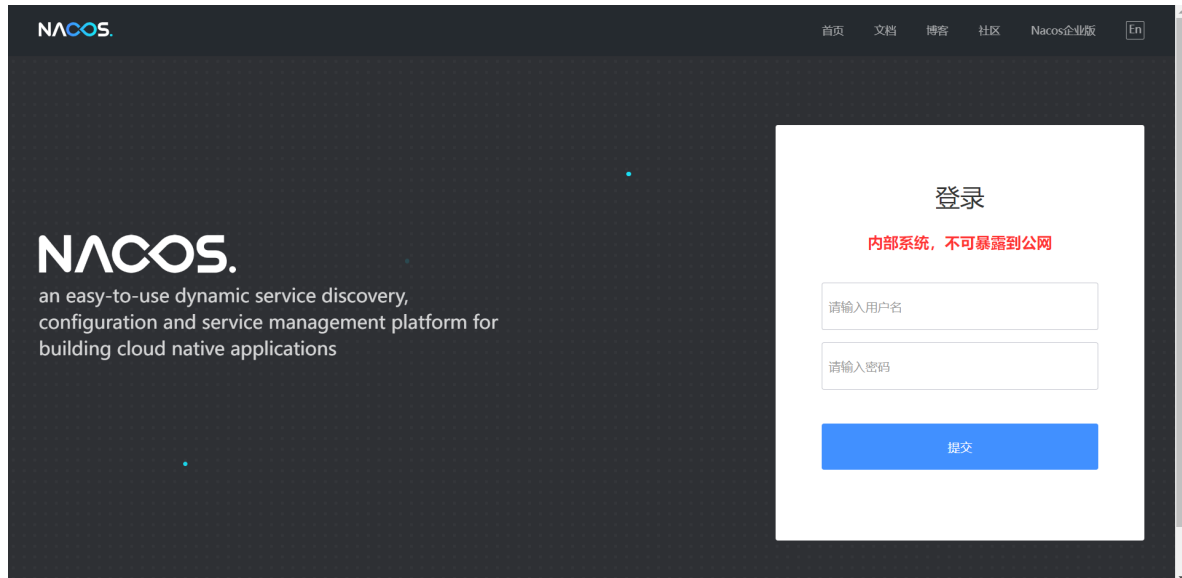
mysql新建nacos的数据库，并执行脚本 sql脚本地址如下：

```
https://github.com/alibaba/nacos/blob/master/config/src/main/resources/META-INF/nacos-db.sql
```

```
docker run -d -p 8848:8848 \
-p 9848:9848 \
-p 9849:9849 \
-e MODE=standalone \
```

```
-e JVM_XMS=128m \  
-e JVM_XMX=128m \  
-e JVM_XMN=128m \  
--name nacos nacos/nacos-server  
  
#设置容器自启动  
docker update --restart=always nacos
```

访问即可



其他版本

```
docker run -d -p 8848:8848 \  
-p 9848:9848 \  
-p 9849:9849 \  
-e JVM_XMS=128m \  
-e JVM_XMX=128m \  
--restart=always \  
-e MODE=standalone \  
-e MYSQL_SERVICE_PORT=3306 \  
-e MYSQL_SERVICE_USER=root \  
-e MYSQL_SERVICE_PASSWORD=123456 \  
-e SPRING_DATASOURCE_PLATFORM=mysql \  
-e MYSQL_SERVICE_HOST=192.168.100.100 \  
--name nacos nacos/nacos-server
```

```
docker run \  
--name nacos -d \  
-p 8848:8848 \  
-p 9848:9848 \  
-p 9849:9849 \  
--privileged=true \  
--restart=always \  
-e JVM_XMS=64m \  
-e JVM_XMX=64m \  
-e MODE=standalone \  
-e PREFER_HOST_MODE=hostname \  

```

```
-e SPRING_DATASOURCE_PLATFORM=mysql \
-e MYSQL_SERVICE_HOST=43.136.129.168 \
-e MYSQL_SERVICE_PORT=3306 \
-e MYSQL_SERVICE_DB_NAME=nacos \
-e MYSQL_SERVICE_USER=root \
-e MYSQL_SERVICE_PASSWORD='Jz.4c8aow|P]7r~*!u[WvCO(Q_G%=Lgd?
sE2hAxRB6}jM0$+feS@/>{T)Yl-m3' \
-v /opt/docker_nacos/logs:/home/nacos/logs \
-v /opt/docker_nacos/init.d/custom.properties:/etc/nacos/init.d/custom.properties \
-v /opt/docker_nacos/data:/home/nacos/data \
nacos/nacos-server
```

创建容器：使用standalone模式并开放8848端口，并映射配置文件和日志目录，数据库默认使用Derby

```
docker run -d -p 8848:8848 -e MODE=standalone -e PREFER_HOST_MODE=hostname -v
/opt/docker_nacos/init.d/custom.properties:/home/nacos/init.d/custom.properties -v
/opt/docker_nacos/logs:/home/nacos/logs --restart always --name nacos nacos/nacos-
server
```

```
docker run --name nacos-quick -e MODE=standalone -p 8848:8848 -d nacos/nacos-
server:latest
```

docker 启动容器

```
docker run \
```

容器名称叫nacos -d后台运行

```
--name nacos -d \
```

nacos默认端口8848 映射到外部端口8848

```
-p 8848:8848 \
```

naocs 应该是2.0版本以后就需要一下的两个端口 所以也需要开放

```
-p 9848:9848
```

```
-p 9849:9849
```

```
--privileged=true \
```

docker重启时 nacos也一并重启

```
--restart=always \
```

-e 配置 启动参数

配置 jvm

```
-e JVM_XMS=256m
```

```
-e JVM_XMX=256m \
```

单机模式

```
-e MODE=standalone
```

```
-e PREFER_HOST_MODE=hostname \
```

数据库是mysql 配置持久化 不使用nacos自带的数据库

```
-e SPRING_DATASOURCE_PLATFORM=mysql \
```

写自己的数据库地址

```
-e MYSQL_SERVICE_HOST=##### \
```

数据库端口号

```
-e MYSQL_SERVICE_PORT=3306 \
```

```
mysql的数据库名称
-e MYSQL_SERVICE_DB_NAME=nacos \

mysql的账号密码
-e MYSQL_SERVICE_USER=root
-e MYSQL_SERVICE_PASSWORD=root \

-v 映射docker内部的文件到docker外部 我这里将nacos的日志 数据 以及配置文件 映射出来
映射日志
-v /root/apply/docker/apply/nacos/logs:/home/nacos/logs \

映射配置文件 (应该没用了 因为前面已经配置参数了)
-v
/root/apply/docker/apply/nacos/init.d/custom.properties:/etc/nacos/init.d/custom.properties \

映射nacos的本地数据 也没啥用因为使用了mysql
-v /root/apply/docker/apply/nacos/data:/home/nacos/data \

启动镜像名称
nacos/nacos-server
```

10JDK

10.1准备

1下载的 jdk-8u161-linux-x64.tar.gz 安装包，并将其直接放在了 root 目录下

2卸载已有JDK

```
rpm -qa | grep java
```

```
[root@localhost ~]# rpm -qa | grep java
java-1.8.0-openjdk-headless-1.8.0.131-11.b12.el7.x86_64 ✓
python-javapackages-3.4.1-11.el7.noarch
libvirt-java-0.4.9-4.el7.noarch
java-1.6.0-openjdk-1.6.0.41-1.13.13.1.el7_3.x86_64 ✓
javapackages-tools-3.4.1-11.el7.noarch
java-1.7.0-openjdk-headless-1.7.0.141-2.6.10.5.el7.x86_64 ✓
java-1.8.0-openjdk-devel-1.8.0.131-11.b12.el7.x86_64 ✓
libvirt-java-devel-0.4.9-4.el7.noarch
java-1.7.0-openjdk-devel-1.7.0.141-2.6.10.5.el7.x86_64 ✓
java-1.7.0-openjdk-1.7.0.141-2.6.10.5.el7.x86_64 ✓
java-1.6.0-openjdk-devel-1.6.0.41-1.13.13.1.el7_3.x86_64 ✓
java-1.8.0-openjdk-1.8.0.131-11.b12.el7.x86_64 ✓
tzdata-java-2017b-1.el7.noarch
```

3卸载java开头的版本

```
yum -y remove java-1.7.0-openjdk-1.7.0.141-2.6.10.5.el7.x86_64
yum -y remove java-1.8.0-openjdk-1.8.0.131-11.b12.el7.x86_64
.....
```

10.2安装

1创建文件夹

```
mkdir /opt/java  
cd /opt/java
```

2将上面准备好的JDK 安装包解压到 /opt/java 中即可

```
tar -zxvf /root/jdk-8u161-linux-x64.tar.gz -C ./
```

解压完之后， /opt/java 目录中会出现一个jdk1.8.0_161 的目录

```
drwxr-xr-x 8 10143 10143 4096 Apr 26 2022 jdk1.8.0_333
```

10.3配置环境变量

1编辑 /etc/profile 文件，在文件尾部加入如下JDK 环境配置即可

```
JAVA_HOME=/opt/java/jdk1.8.0_333  
CLASSPATH=$JAVA_HOME/lib/  
PATH=$PATH:$JAVA_HOME/bin  
export PATH JAVA_
```

2让环境变量生效

```
source /etc/profile
```

10.4验证JDK

```
java -version  
javac
```

```
[user@VM-0-8-centos java]$ java -version  
java version "1.8.0_333"  
Java(TM) SE Runtime Environment (build 1.8.0_333-b02)  
Java HotSpot(TM) 64-Bit Server VM (build 25.333-b02, mixed mode)
```

11Linux命令行部署微服务（占用内存大，不推荐）

11.1为Linux中Java文件夹添加可执行权限

```
chmod 777 java
```

为项目所有的微服务模块添加打包可执行jar包pom配置，除了公共feign外

```
<packaging>jar</packaging>  
  
<build>  
  <plugins>  
    <plugin>
```

```

<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-maven-plugin</artifactId>
<configuration>
    <fork>true</fork>
    <mainClass>com.atguigu.AdminApplication</mainClass>
</configuration>
<executions>
    <execution>
        <goals>
            <goal>repackage</goal>
        </goals>
    </execution>
</executions>
</plugin>
</plugins>
</build>

```

在本地通过测试即可上传服务器

```
java -jar xxx.jar
```

11.2服务器配置

注意可以手动启动 `nohup java -Xms64m -Xmx64m -jar /opt/springcloud/store-gateway-1.0.0.jar > gateway.log 2>&1 &`，不需要脚本一键启动

1新建jar包自启脚本（位置随意，文件名随意）xxx.sh

```
vim /opt/java_springcloud/startJar.sh
```

2写入配置文件

```
sudo rm -rf *.log
```

```

#!/bin/bash
##JDK start 配置自己服务器的JDK环境(jdk1.8还是11)
## JDK环境变量配置（如果不清楚本机的环境变量，可以通过more /etc/profile命令查看）
export JAVA_HOME=/opt/java/jdk1.8.0_333
export
CLASSPATH=.:$JAVA_HOME/lib/dt.jar:$JAVA_HOME/lib/tools.jar:$JAVA_HOME/jre/lib/rt.jar
export PATH=$PATH:$JAVA_HOME/bin

## JDK end
# 有多少个jar包就在后面加上去(可参考该jar包的jenkins启动脚本)
nohup java -Xms64m -Xmx64m -jar /opt/springcloud/store-gateway-1.0.0.jar > gateway.log 2>&1 &
nohup java -Xms32m -Xmx32m -jar /opt/springcloud/store-admin-1.0.0.jar > admin.log 2>&1 &
nohup java -Xms32m -Xmx32m -jar /opt/springcloud/store-front-carousel-1.0.0.jar > carousel.log 2>&1 &
nohup java -Xms32m -Xmx32m -jar /opt/springcloud/store-front-cart-1.0.0.jar > cart.log 2>&1 &
nohup java -Xms32m -Xmx32m -jar /opt/springcloud/store-front-category-1.0.0.jar > category.log 2>&1 &

```

```
nohup java -Xms32m -Xmx32m -jar /opt/springcloud/store-front-collect-1.0.0.jar > collect 2>&1 &
nohup java -Xms64m -Xmx64m -jar /opt/springcloud/store-front-order-1.0.0.jar > order.log 2>&1 &
nohup java -Xms64m -Xmx64m -jar /opt/springcloud/store-front-product-1.0.0.jar > product.log 2>&1 &
nohup java -Xms32m -Xmx32m -jar /opt/springcloud/store-front-user-1.0.0.jar > user.log 2>&1 &
nohup java -Xms64m -Xmx64m -jar /opt/springcloud/store-static-oss-1.0.0.jar > oss.log 2>&1 &
nohup java -Xms32m -Xmx32m -jar /opt/springcloud/store-search-1.0.0.jar > search.log 2>&1 &
```

nohup命令可以让java在后台运行，即使关闭窗口也会继续运行

3编辑服务器开机时自动执行的文件

```
sudo vi /etc/rc.d/rc.local
```

在此文件的touch /var/lock/subsys/local底下
添加以下内容：

```
#开机启动jar脚本
/opt/java_springcloud/startJar.sh
```

```
touch /var/lock/subsys/local
/usr/local/qcloud/irq/net_smp_affinity.sh >/tmp/net_affinity.log 2>&1
/usr/local/qcloud/cpuidle/cpuidle_support.sh &> /tmp/cpuidle_support.log
/usr/local/qcloud/rps/set_rps.sh >/tmp/setRps.log 2>&1
/usr/local/qcloud/irq/virtio_blk_smp_affinity.sh > /tmp/virtio_blk_affinity.log 2>&1
/usr/local/qcloud/gpu/nv_gpu_conf.sh >/tmp/nv_gpu_conf.log 2>&1
/usr/local/qcloud/scripts/disable_rt_runtime_share.sh >/tmp/disable_rt_runtime_share.log 2>&1
#开机启动jar脚本
/opt/java_springcloud/startJar.sh
```

保存即可。

4分配可运行权限，两个脚本文件都变成绿色即实现

```
chmod 755 startJar.sh
```

```
-rwxr-xr-x 1 root root 972 Mar 23 22:28 rc.local
-rwxr-xr-x 1 root root 1508 Mar 23 22:24 startJar.sh
```

5启动sh，

```
sh startJar.sh
```

6查看服务

```
ps -ef|grep java
```

7停止服务

通过上一步骤查出的pid，kill掉


```
sudo kill 15169
```

12 Docker Compose部署微服务（推荐）

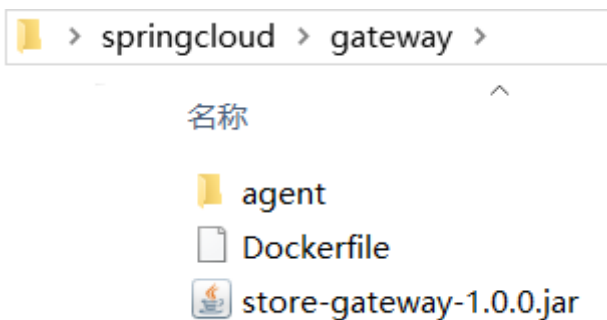
注意，可以手动启动docker镜像。不需要docker compose编排部署

12.1 Docker Compose安装

```
pip install docker-compose
```

12.2 准备材料

jar包生成在11.1中有说明。



空的agent文件夹，带有内容的Dockerfile，以及jar包，放在同一文件夹下。

Dockerfile内容：

```
#基于哪个镜像
From java:8
#复制文件到容器
ADD store-gateway-1.0.0.jar /app.jar
ADD agent /agent
#配置容器启动后执行的命令
ENTRYPOINT ["java","-jar","/app.jar"]

EXPOSE 3000
```

12.3 构建镜像

```
docker build -t emall-gateway:0.0.1 .
```

```
[root@iZf8zi2bp8l1lbuvxxt8vkZ gateway]# docker build -t emall-gateway:0.0.1 .
[+] Building 45.2s (8/8) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 246B
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [internal] load metadata for docker.io/library/java:8
=> [1/3] FROM docker.io/library/java:8@sha256:c1ff613e8ba25833d2e1940da0940c3824f03f802c449f3d1815a66b7f8c0e9d
=> => resolve docker.io/library/java:8@sha256:c1ff613e8ba25833d2e1940da0940c3824f03f802c449f3d1815a66b7f8c0e9d
=> => sha256:c1ff613e8ba25833d2e1940da0940c3824f03f802c449f3d1815a66b7f8c0e9d 2.00kB / 2.00kB
=> => sha256:5040bd2983909aa8896b9932438c3f1479d25ae837a5f6220242a264d0221f2d 51.36MB / 51.36MB
=> => sha256:fce5728aad85a763fe3c419db16885eb6f7a670a42824ea618414b8fb309ccde 18.54MB / 18.54MB
=> => sha256:d23bd5b1b1b1afce5f1d0fd33e7ed8afbc084b594b9ccf742a5b27080d8a4a8 4.73kB / 4.73kB
=> => sha256:76610ec20bf5892e24cebd4153c7668284aa1d1151b7c3b0c7d50c579aa5ce75 42.50MB / 42.50MB
=> => sha256:60170fec2151d2108ed1420625c5113843ba4e0223d3023353d3f32ffe3cfc2 593.15kB / 593.15kB
=> => sha256:e98f73de8f0d2ef292f58b004d67bc6e9ee779dcfaff7ebb3964649d4787b872 214B / 214B
=> => sha256:11f7af24ed9cf47597dd6cf9963bb3e9109c963f0135e869a9e9b4999fdc12a3 242B / 242B
=> => sha256:49e2d6393f32abb1de7c9395c04c822ceb2287383d5a90998f7bd8dbfd43d48c 130.10MB / 130.10MB
=> => sha256:bb9cdec9c7f337940f7d872274353b66e118412cbfd433c711361bcf7922aea4 289.05kB / 289.05kB
=> => extracting sha256:5040bd2983909aa8896b9932438c3f1479d25ae837a5f6220242a264d0221f2d
=> => extracting sha256:fce5728aad85a763fe3c419db16885eb6f7a670a42824ea618414b8fb309ccde
=> => extracting sha256:76610ec20bf5892e24cebd4153c7668284aa1d1151b7c3b0c7d50c579aa5ce75
=> => extracting sha256:60170fec2151d2108ed1420625c5113843ba4e0223d3023353d3f32ffe3cfc2
=> => extracting sha256:e98f73de8f0d2ef292f58b004d67bc6e9ee779dcfaff7ebb3964649d4787b872
=> => extracting sha256:11f7af24ed9cf47597dd6cf9963bb3e9109c963f0135e869a9e9b4999fdc12a3
=> => extracting sha256:49e2d6393f32abb1de7c9395c04c822ceb2287383d5a90998f7bd8dbfd43d48c
=> => extracting sha256:bb9cdec9c7f337940f7d872274353b66e118412cbfd433c711361bcf7922aea4
=> [internal] load build context
=> => transferring context: 72B
=> [2/3] ADD store-gateway-1.0.0.jar /app.jar
=> [3/3] ADD agent /agent
=> exporting to image
=> => exporting layers
=> => writing image sha256:03dc3df6dcab5a772d16234e92de35857b592252d167331a6e25aa0fbad181f6
=> => naming to docker.io/library/emall-gateway:0.0.1
[root@iZf8zi2bp8l1lbuvxxt8vkZ gateway]#
```

docker build -t emall-admin:0.0.1 .

docker build -t emall-carousel:0.0.1 .

docker build -t emall-cart:0.0.1 .

docker build -t emall-category:0.0.1 .

docker build -t emall-collect:0.0.1 .

docker build -t emall-order:0.0.1 .

docker build -t emall-oss:0.0.1 .

docker build -t emall-product:0.0.1 .

docker build -t emall-search:0.0.1 .

docker build -t emall-user:0.0.1 .

12.4查看镜像

docker images

```
[root@iZf8zi2bp8l1lbuvxxt8vkZ gateway]# docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
emall-gateway	0.0.1	03dc3df6dcab	5 minutes ago	683MB

12.5运行镜像

服务器 1.7g swap内存2g

docker run -d -p 3000:3000 emall-gateway:0.0.1

docker run -d -p 3009:3009 -m 200M emall-cart:0.0.1

```
docker run -d -p 3005:3005 -m 200M email-category:0.0.1
docker run -d -p 3008:3008 -m 200M email-collect:0.0.1
docker run -d -p 3011:3011 -m 200M email-admin:0.0.1
docker run -d -p 3010:3010 email-order:0.0.1
docker run -d -p 3002:3002 -m 200M email-oss:0.0.1
docker run -d -p 3004:3004 -m 200M email-product:0.0.1
docker run -d -p 3001:3001 -m 200M email-user:0.0.1
docker run -d -p 3003:3003 -m 200M email-carousel:0.0.1
docker run -d -p 3007:3007 -m 200M email-search:0.0.1
```

12.6添加swap交换分区空间

1.查看当前内存和swap空间大小

```
free -mh
```

2.创建swap交换分区文件/swap/swapfile, 大小为8G

```
sudo mkdir /swap
sudo dd if=/dev/zero of=/swap/swapfile bs=1G count=8
```

3.格式化swap分区:

```
sudo mkswap /swap/swapfile
```

4.设置交换分区:

```
sudo mkswap -f /swap/swapfile
```

5.修改权限:

```
sudo chmod 600 /swap/swapfile
```

6.激活swap分区:

```
sudo swapon /swap/swapfile
```

6.设为开机自动启用:

```
sudo vi /etc/fstab
```

在该文件底部添加如下内容:

```
/swap/swapfile swap swap default 0 0
```

注意: 要执行格式化swap分区这一步骤, 否则出现无法激活的问题。

12.7删除swap交换分区

1.查询swap分区文件:

```
swapon -s
```

Filename	Type	Size	Used	Priority
/dev/dm-1	partition	2097148	685964	-1

1.停止正在使用的swap分区:

```
swapoff /dev/dm-1
```

2.删除swap分区文件:

```
rm /dev/dm-1
```

3.删除或注释在/etc/fstab文件中的以下开机自动挂载内容:

```
/dev/mapper/centos-swap swap swap default 0 0
```

彻底关闭swap分区方法:

```
swapoff -a
```

13部署前端项目

将前端项目打包, 会生成dist文件夹, 上传到nginx中html目录下即可。

记得修改配置文件的接口地址

```
http {
    server {
        listen 80;
        server_name 43 168;
        #charset koi8-r;
        #access_log logs/host.access.log main;
        location / {
            root /usr/share/nginx/html/dist;
            index index.html index.htm;
        }
        location /prod-api/ {
            proxy_set_header Host $http_host;
            proxy_set_header X-Real-IP $remote_addr;
            proxy_set_header REMOTE-HOST $remote_addr;
            proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
            proxy_pass http://192.168.1.100:8081/;
        }
    }
}

include /etc/nginx/mime.types;
default_type application/octet-stream;
```