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。

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'
{
"registry-mirrors": ["https://********.mirror.aliyuncs.com"]
}
EOF

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

2.2docker卸载

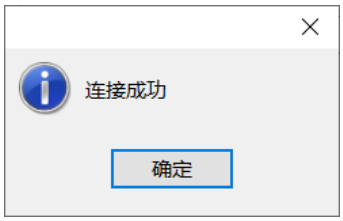
```
yum remove docker-ce docker-ce-cli containerd.io # 卸载依赖
rm -rf /var/lib/docker # 删除资源 . /var/lib/docker是docker的默认工作路径
```

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)
> 将容器的配置文件夹/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

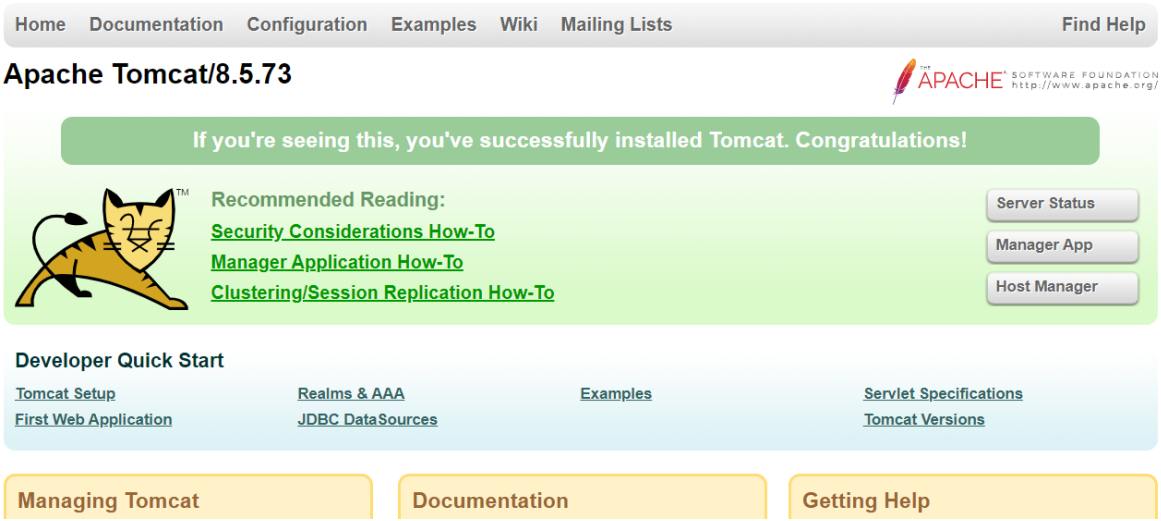
```
docker exec -it 容器ID /bin/bash
mysql -uroot -p
use mysql
SET PASSWORD FOR 'root' = PASSWORD('设置的密码');
exit
docker restart 容器ID
```

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



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

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
```

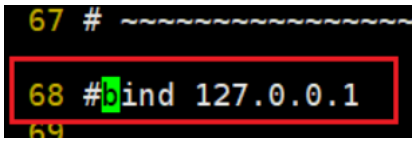
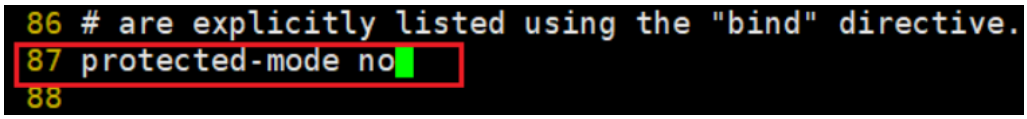
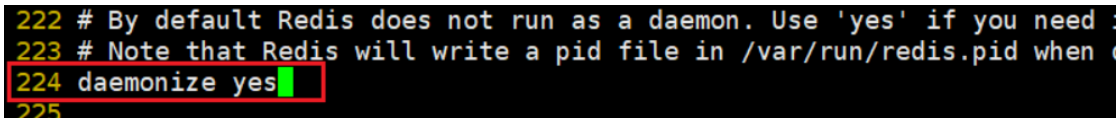
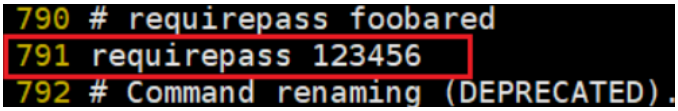
```
[user@VM-0-8-centos ~]$ sudo docker exec -it e2c20118b7f8 /bin/bash
root@e2c20118b7f8:/usr/local/tomcat# ls
BUILDING.txt     LICENSE  README.md      RUNNING.txt  conf  logs            temp     webapps.dist
CONTRIBUTING.md  NOTICE   RELEASE-NOTES  bin          lib   native-jni-lib  webapps  work
root@e2c20118b7f8:/usr/local/tomcat# 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
>
> 
>
>
> 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
>
> 
>
>
> daemonize yes 默认no 为不守护进程模式，修改为yes
>
> 
>
>
> 设置密码（建议设置，不设置有风险）
>
> 
>
>
> 持久化(可选)
>
> 
>

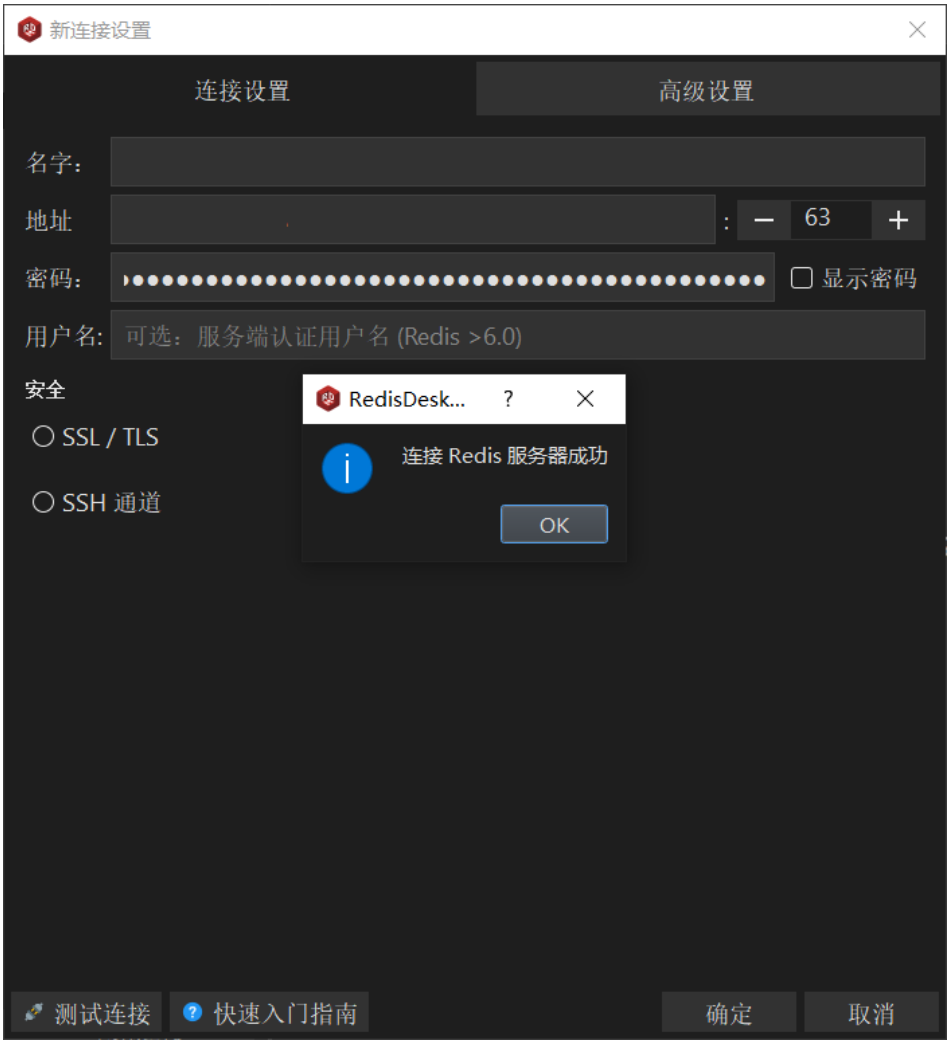
```
docker run -p 6379:6379 --name redis -v /docker-
software/redis/redis.conf:/etc/redis/redis.conf  -v /docker-
software/redis/data:/data -d redis redis-server /etc/redis/redis.conf --
appendonly yes
```

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

```
[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。

7RabbitMq

```
docker pull rabbitmq
```

7.1端口开放

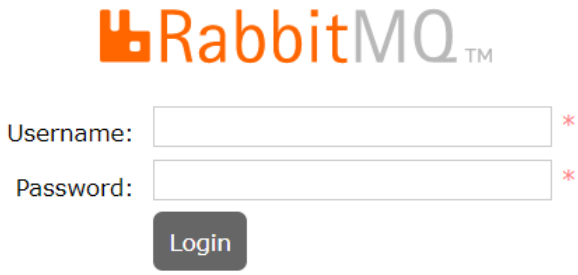
如果在云服务上部署需在安全组开通一下端口：15672、5672、25672、61613、1883。

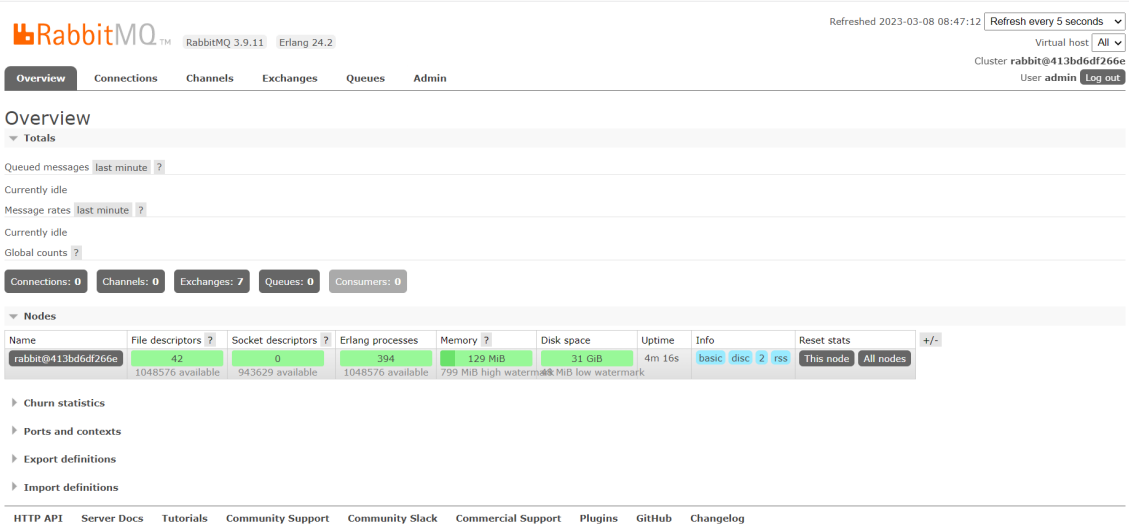
> 15672(UI页面通信口)
>
> 5672(client端通信口)
>
> 25672(server间内部通信口)
>
> 61613(stomp 消息传输)
>
> 1883(MQTT消息队列遥测传输)

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。*





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/elastics
earch.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"
}
```


8.1kibana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