

本文主要以图文的形式讲解mall在Linux环境下的部署，涉及在Docker容器中安装MySQL、Redis、Nginx、RabbitMQ、MongoDB、Elasticsearch、Logstash、Kibana，以及SpringBoot应用部署，基于[CenterOS7.6](#)。

## Docker环境安装

- 安装yum-utils:

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

- 为yum源添加docker仓库位置:

```
1 yum-config-manager --add-repo https://download.docker.com/linux/centos/docker-ce.repo
```

- 安装docker:

```
1 yum install docker-ce
```

- 启动docker:

```
1 systemctl start docker
```

## MySQL安装

- 下载MySQL 5.7的docker镜像:

```
1 docker pull mysql:5.7
```

- 使用如下命令启动MySQL服务:

```
1 docker run -p 3306:3306 --name mysql \
2 -v /mydata/mysql/log:/var/log/mysql \
3 -v /mydata/mysql/data:/var/lib/mysql \
4 -v /mydata/mysql/conf:/etc/mysql \
5 -e MYSQL_ROOT_PASSWORD=root \
6 -d mysql:5.7
```

- 参数说明

- -p 3306:3306: 将容器的3306端口映射到主机的3306端口
- -v /mydata/mysql/conf:/etc/mysql: 将配置文件夹挂在到主机
- -v /mydata/mysql/log:/var/log/mysql: 将日志文件夹挂载到主机
- -v /mydata/mysql/data:/var/lib/mysql: 将数据文件夹挂载到主机
- -e MYSQL\_ROOT\_PASSWORD=root: 初始化root用户的密码

- 进入运行MySQL的docker容器:

```
1 docker exec -it mysql /bin/bash
```

- 使用MySQL命令打开客户端:

```
1 mysql -uroot -proot --default-character-set=utf8
```

- 创建mall数据库:

```
1 create database mall character set utf8
```

- 安装上传下载插件，并将document/sql/mall.sql上传到Linux服务器上:

```
1 yum -y install lrzsz
```

- 将mall.sql文件拷贝到mysql容器的/目录下:

```
1 docker cp /mydata/mall.sql mysql:/
```

- 将sql文件导入到数据库:

```
1 use mall;
2 source /mall.sql;
```

- 创建一个reader:123456帐号并修改权限，使得任何ip都能访问:

```
1 grant all privileges on *.* to 'reader' '%' identified by '123456';
2
```

## Redis安装

- 下载Redis 5.0的docker镜像:

```
1 docker pull redis:5
```

- 使用如下命令启动Redis服务：

```
1 docker run -p 6379:6379 --name redis \  
2 -v /mydata/redis/data:/data \  
3 -d redis:5 redis-server --appendonly yes
```

- 进入Redis容器使用redis-cli命令进行连接：

```
1 docker exec -it redis redis-cli
```

```
[root@local-linux ~]# docker run -p 6379:6379 --name redis \  
> -v /mydata/redis/data:/data \  
> -d redis:3.2 redis-server --appendonly yes  
58d2e04a41fac479596f7bbda5e8890fa766b05ba1464bac6c44f5ba32347c2  
[root@local-linux ~]# docker exec -it redis redis-cli  
127.0.0.1:6379> set a 100  
OK  
127.0.0.1:6379> get a  
"100"
```

## Ngixn安装

- 下载Ngixn1.10的docker镜像：

```
1 docker pull nginx:1.10
```

- 先运行一次容器（为了拷贝配置文件）：

```
1 docker run -p 80:80 --name nginx \  
2 -v /mydata/nginx/html:/usr/share/nginx/html \  
3 -v /mydata/nginx/logs:/var/log/nginx \  
4 -d nginx:1.10
```

- 将容器内的配置文件拷贝到指定目录：

```
1 docker container cp nginx:/etc/nginx /mydata/nginx/
```

- 修改文件名称：

```
1 mv nginx conf
```

- 终止并删除容器：

```
1 docker stop nginx  
2 docker rm nginx
```

- 使用如下命令启动Ngixn服务：

```
1 docker run -p 80:80 --name nginx \  
2 -v /mydata/nginx/html:/usr/share/nginx/html \  
3 -v /mydata/nginx/logs:/var/log/nginx \  
4 -v /mydata/nginx/conf:/etc/nginx \  
5 -d nginx:1.10
```

## RabbitMQ安装

- 下载rabbitmq3.7.15的docker镜像：

```
1 docker pull rabbitmq:3.7.15
```

- 使用如下命令启动RabbitMQ服务：

```
1 docker run -p 5672:5672 -p 15672:15672 --name rabbitmq \  
2 -d rabbitmq:3.7.15
```

- 进入容器并开启管理功能：

```
1 docker exec -it rabbitmq /bin/bash  
2 rabbitmq-plugins enable rabbitmq_management
```

```
[root@local-linux nginx]# docker exec -it rabbitmq /bin/bash
root@dd5fafeb8847:/# rabbitmq-plugins enable rabbitmq_management
Enabling plugins on node rabbit@dd5fafeb8847:
rabbitmq_management
The following plugins have been configured:
  rabbitmq_management
  rabbitmq_management_agent
  rabbitmq_web_dispatch
Applying plugin configuration to rabbit@dd5fafeb8847...
The following plugins have been enabled:
  rabbitmq_management
  rabbitmq_management_agent
  rabbitmq_web_dispatch
started 3 plugins.
```

- 开启防火墙:

```
1 firewall-cmd --zone=public --add-port=15672/tcp --permanent
2 firewall-cmd --reload
```

- 访问地址查看是否安装成功: <http://192.168.3.101:15672>



Username:  \*

Password:  \*

- 输入账号密码并登录: guest guest
- 创建帐号并设置其角色为管理员: mall mall



3.7.15 Erlang 22.0.2

Overview Connections Channels Exchanges Queues **Admin**

## Users

▼ All users

Filter:  ☐ Regex ? 1 it

Name	Tags	Can access virtual hosts	Has password
guest	administrator	/	•

?

▼ Add a user

Username:  \*

Password:  \*  \* (confirm)

Tags:  ?

Set **Admin** | Monitoring | Policymaker  
Management | Impersonator | None

- 创建一个新的虚拟host为: /mall

RabbitMQ 3.7.15 Erlang 22.0.2 Refreshed 2019-06-09 16:01:07 Refresh every 5 seconds Virtual host All Cluster rabbit@dd5f4feb8847 User guest Log out

Overview Connections Channels Exchanges Queues Admin

### Virtual Hosts

▼ All virtual hosts

Filter:  ☐ Regex 1 item, page size up to 100

Overview				Messages		Network		Message rates	
Name	Users	State	Ready	Unacked	Total	From client	To client	publish	deliver / get
/	guest	running	NaN	NaN	NaN				

▼ Add a new virtual host

Name:

Add virtual host

Users

Virtual Hosts

Policies

Limits

Cluster

- 点击mall用户进入用户配置页面

Overview Connections Channels Exchanges Queues Admin

### Users

▼ All users

Filter:  ☐ Regex 2 items, page size up to 100

Name	Tags	Can access virtual hosts	Has password
quest	administrator	/, /mall	•
mall	administrator	No access	•

Users

Virtual Hosts

Policies

Limits

Cluster

- 给mall用户配置该虚拟host的权限

## User: mall

This user does not have permission to access any virtual host. Use "Set Permission" below to grant permission to access virtual hosts.

### Overview

Tags administrator

Can log in with password •

### Permissions

Current permissions

... no permissions ...

Set permission

Virtual Host:

Configure regexp:

Write regexp:

Read regexp:

Set permission

## Elasticsearch安装

- 下载Elasticsearch 7.6.2的docker镜像:

```
1 docker pull elasticsearch:7.6.2
```

- 修改虚拟内存区域大小, 否则会因为过小而无法启动:

```
1 sysctl -w vm.max_map_count=262144
```

- 使用如下命令启动Elasticsearch服务:

```
1 docker run -p 9200:9200 -p 9300:9300 --name elasticsearch \
2 -e "discovery.type=single-node" \
3 -e "cluster.name=elasticsearch" \
4 -v /mydata/elasticsearch/plugins:/usr/share/elasticsearch/plugins \
5 -v /mydata/elasticsearch/data:/usr/share/elasticsearch/data \
6 -d elasticsearch:7.6.2
```

- 启动时会发现/usr/share/elasticsearch/data目录没有访问权限, 只需要修改/mydata/elasticsearch/data目录的权限, 再重新启动即可;

```
1 chmod 777 /mydata/elasticsearch/data/
```

- 安装中文分词器IKAnalyzer，并重新启动：

```
1 docker exec -it elasticsearch /bin/bash
```

```
2 #此命令需要在容器中运行
```

```
3 elasticsearch-plugin install https://github.com/medcl/elasticsearch-analysis-ik/releases/download/v7.6.2/elasticsearch-analysis-ik-7.6.2.zip
```

```
4 docker restart elasticsearch
```

- 开启防火墙：

```
1 firewall-cmd --zone=public --add-port=9200/tcp --permanent
```

```
2 firewall-cmd --reload
```

- 访问会返回版本信息：<http://192.168.3.101:9200>

```
{
  "name": "DESKTOP-5N1MJ19",
  "cluster_name": "elasticsearch",
  "cluster_uuid": "HNvVRirdQW-2Q4xEpXdpsA",
  "version": {
    "number": "7.6.2",
    "build_flavor": "default",
    "build_type": "zip",
    "build_hash": "ef48eb35cf30adf4db14086e8aabd07ef6fb113f",
    "build_date": "2020-03-26T06:34:37.794943Z",
    "build_snapshot": false,
    "lucene_version": "8.4.0",
    "minimum_wire_compatibility_version": "6.8.0",
    "minimum_index_compatibility_version": "6.0.0-beta1"
  },
  "tagline": "You Know, for Search"
}
```

## Logstash安装

- 下载Logstash7.6.2的docker镜像：

```
1 docker pull logstash:7.6.2
```

- 修改Logstash的配置文件logstash.conf中output节点下的Elasticsearch连接地址为es:9200，配置文件地址：\_

```
1 input {
2   tcp {
3     mode => "server"
4     host => "0.0.0.0"
5     port => 4560
6     codec => json_lines
7     type => "debug"
8   }
9   tcp {
10    mode => "server"
11    host => "0.0.0.0"
12    port => 4561
13    codec => json_lines
14    type => "error"
15  }
16  tcp {
17    mode => "server"
18    host => "0.0.0.0"
19    port => 4562
20    codec => json_lines
21    type => "business"
22  }
23  tcp {
24    mode => "server"
25    host => "0.0.0.0"
```

```

26 port => 4563
27 codec => json_lines
28 type => "record"
29 }
30 }
31 filter{
32   if [type] == "record" {
33     mutate {
34       remove_field => "port"
35       remove_field => "host"
36       remove_field => "@version"
37     }
38   }
39   json {
40     source => "message"
41     remove_field => ["message"]
42   }
43 }
44 output {
45   elasticsearch {
46     hosts => "192.168.50.66:9200"
47     index => "mall-%{type}-%{+YYYY.MM.dd}"
48   }
49 }

```

```

1 output {
2   elasticsearch {
3     hosts => "es:9200"
4     index => "mall-%{type}-%{+YYYY.MM.dd}"
5   }
6 }

```

- 创建 `/mydata/logstash` 目录，并将Logstash的配置文件 `logstash.conf` 拷贝到该目录；

```
1 mkdir /mydata/logstash
```

- 使用如下命令启动Logstash服务；

```

1 docker run --name logstash -p 4560:4560 -p 4561:4561 -p 4562:4562 -p 4563:4563 \
2 --link elasticsearch:es \
3 -v /mydata/logstash/logstash.conf:/usr/share/logstash/pipeline/logstash.conf \
4 -d logstash:7.6.2

```

- 进入容器内部，安装 `json_lines` 插件。

```
1 logstash-plugin install logstash-codec-json_lines
```

## Kibana安装

- 下载Kibana7.6.2的docker镜像：

```
1 docker pull kibana:7.6.2
```

- 使用如下命令启动Kibana服务：

```

1 docker run --name kibana -p 5601:5601 \
2 --link elasticsearch:es \
3 -e "elasticsearch.hosts=http://es:9200" \
4 -d kibana:7.6.2

```

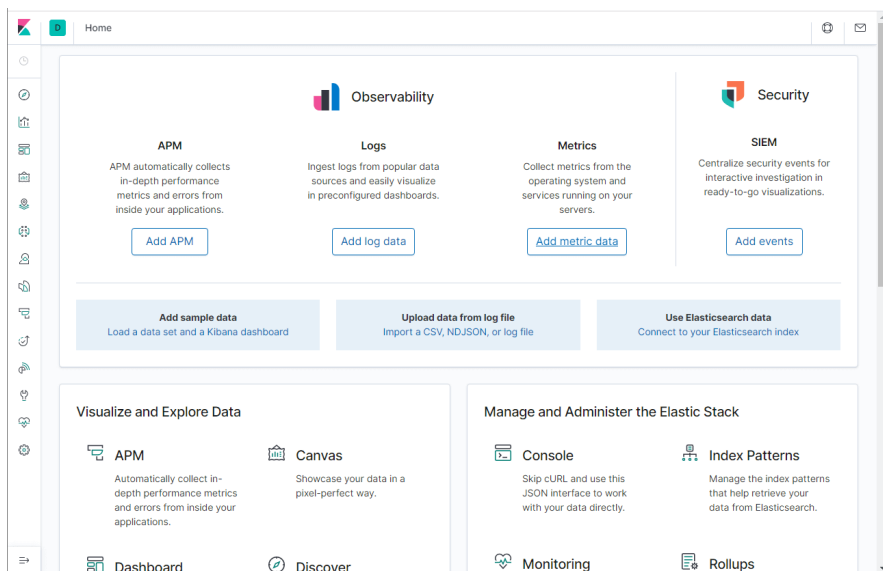
- 开启防火墙：

```

1 firewall-cmd --zone=public --add-port=5601/tcp --permanent
2 firewall-cmd --reload

```

- 访问地址进行测试：<http://虚拟机IP:5601>



## MongoDB安装

- 下载MongoDB4.2.5的docker镜像：

```
1 docker pull mongo:4.2.5
```

- 使用docker命令启动：

```
1 docker run -p 27017:27017 --name mongo \
2 -v /mydata/mongo/db:/data/db \
3 -d mongo:4.2.5
```

## Docker全部环境安装完成

- 所有下载镜像文件：

```
1 REPOSITORY TAG IMAGE ID CREATED SIZE
2 redis 5 071538dbbd71 2 weeks ago 98.3MB
3 mongo 4.2.5 fddee5bccba3 3 months ago 388MB
4 logstash 7.6.2 fa5b3b1e9757 4 months ago 813MB
5 kibana 7.6.2 f70986bc5191 4 months ago 1.01GB
6 elasticsearch 7.6.2 f29a1ee41030 4 months ago 791MB
7 rabbitmq 3.7.15-management 6ffc11daa8d0 13 months ago 186MB
8 mysql 5.7 7faa3c53e6d6 15 months ago 373MB
9 registry 2 f32a97de94e1 17 months ago 25.8MB
10 nginx 1.10 0346349a1a64 3 years ago 182MB
11 java 8 d23bdf5b1b1b 3 years ago 643MB
```

- 所有运行在容器里面的应用：

CONTAINER ID	IMAGE	COMMAND	NAMES	CREATED	STATUS	PORTS
18fbf8a832a2	logstash:7.6.2	"/usr/local/bin/dock..."	logstash	6 days ago	Up About a minute	5044/tcp, 0.0.0.0:4560-4563->456
0-4563/tcp, 9600/tcp	kibana:7.6.2	"/usr/local/bin/dumb..."	kibana	13 days ago	Up About a minute	0.0.0.0:5601->5601/tcp
61beaa6f12b9	elasticsearch:7.6.2	"/usr/local/bin/dock..."	elasticsearch	13 days ago	Up About a minute	0.0.0.0:9200->9200/tcp, 0.0.0.0:
167aeb634e52	mongo:4.2.5	"docker-entrypoint.s..."	mongo	13 days ago	Up About a minute	0.0.0.0:27017->27017/tcp
9300->9300/tcp	redis:5	"docker-entrypoint.s..."	redis	13 days ago	Up About a minute	0.0.0.0:6379->6379/tcp
4d7ce609621d	rabbitmq:3.7.15-management	"docker-entrypoint.s..."	rabbitmq	13 months ago	Up About a minute	4369/tcp, 5671/tcp, 0.0.0.0:5672
4f9fee5e4b70	nginx:1.10	"nginx -g 'daemon of..."	nginx	13 months ago	Up About a minute	0.0.0.0:80->80/tcp, 443/tcp
c1fae9ea0016	mysql:5.7	"docker-entrypoint.s..."	mysql	13 months ago	Up 5 minutes	0.0.0.0:3306->3306/tcp, 33060/tc
->5672/tcp, 15671/tcp, 25672/tcp, 0.0.0.0:15672->15672/tcp	registry:2	"/entrypoint.sh /etc..."	registry2	14 months ago	Up 5 minutes	0.0.0.0:5000->5000/tcp
116a45d0ea2c						
8457cc4c8988						
p						
997175ddfa40						

## SpringBoot应用部署

### 构建所有Docker镜像并上传

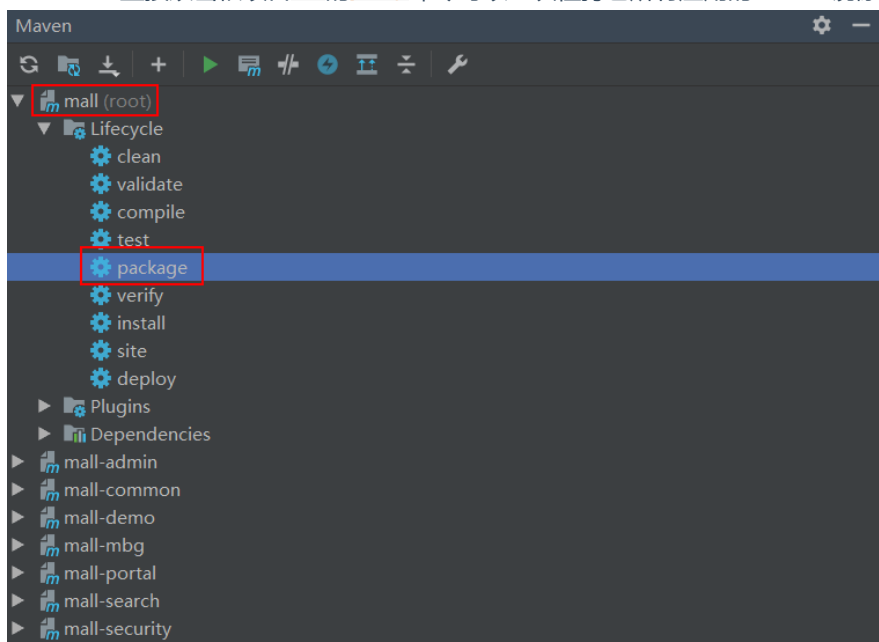
- 修改项目根目录下的pom.xml中的docker.host属性：

```
1 <properties>
2 <docker.host>http://192.168.3.101:2375</docker.host>
3 </properties>
```

- 如果项目根目录的pom.xml中docker-maven-plugin的<executions>节点被注释掉了就打开注释，使项目在打包时直接构建Docker镜像；

```
197 <version>${minio.version}</version>
198 </dependency>
199 </dependencies>
200 </dependencyManagement>
201
202 <build>
203 <pluginManagement>
204 <plugins>
205 <plugin>
206 <groupId>org.springframework.boot</groupId>
207 <artifactId>spring-boot-maven-plugin</artifactId>
208 </plugin>
209 <plugin>
210 <groupId>com.spotify</groupId>
211 <artifactId>docker-maven-plugin</artifactId>
212 <version>${docker.maven.plugin.version}</version>
213 <!--<executions>-->
214 <!--<execution>-->
215 <!--<id>build-image</id>-->
216 <!--<phase>package</phase>-->
217 <!--<goals>-->
218 <!--<goal>build</goal>-->
219 <!--</goals>-->
220 <!--</execution>-->
221 <!--</executions>-->
222 <configuration>...</configuration>
223 </plugin>
224 </plugins>
225 </pluginManagement>
226 </build>
227
228 </project>
```

- 直接双击根项目mall的package命令可以一次性打包所有应用的Docker镜像；



```
1 REPOSITORY TAG IMAGE ID CREATED SIZE
2 mall/mall-portal 1.0-SNAPSHOT 70e0f76416a0 21 seconds ago 705MB
3 mall/mall-search 1.0-SNAPSHOT f3290bd1d0c7 41 seconds ago 725MB
4 mall/mall-admin 1.0-SNAPSHOT 26557b93a106 About a minute ago 705MB
```

## 部署mall-admin

```
1 docker run -p 8080:8080 --name mall-admin \
2 --link mysql:db \
3 --link redis:redis \
4 -v /etc/localtime:/etc/localtime \
5 -v /mydata/app/admin/logs:/var/logs \
6 -d mall/mall-admin:1.0-SNAPSHOT
```

注意：如果想使用Logstash收集日志的话，需要将应用容器连接到Logstash，添加如下配置即可；

```
1 --link logstash:logstash \
```



## 部署mall-search

```
1 docker run -p 8081:8081 --name mall-search \  
2 --link elasticsearch:es \  
3 --link mysql:db \  
4 -v /etc/localtime:/etc/localtime \  
5 -v /mydata/app/search/logs:/var/logs \  
6 -d mall/mall-search:1.0-SNAPSHOT
```

## 部署mall-port

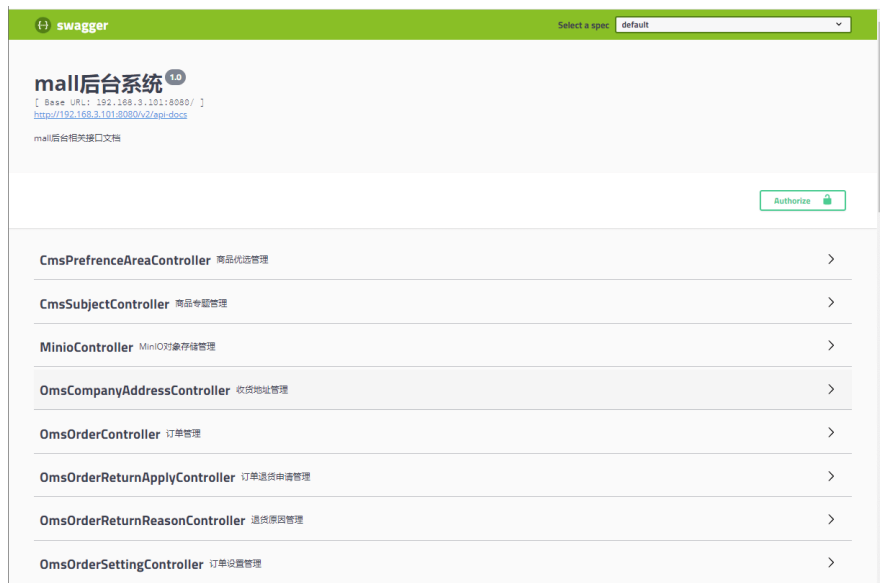
```
1 docker run -p 8085:8085 --name mall-portal \  
2 --link mysql:db \  
3 --link redis:redis \  
4 --link mongo:mongo \  
5 --link rabbitmq:rabbitmq \  
6 -v /etc/localtime:/etc/localtime \  
7 -v /mydata/app/portal/logs:/var/logs \  
8 -d mall/mall-portal:1.0-SNAPSHOT
```

## 开启防火墙

```
1 firewall-cmd --zone=public --add-port=8080/tcp --permanent \  
2 firewall-cmd --zone=public --add-port=8081/tcp --permanent \  
3 firewall-cmd --zone=public --add-port=8085/tcp --permanent \  
4 firewall-cmd --reload
```

## 访问接口进行测试

- mall-admin的api接口文档地址: <http://虚拟机IP:8080/swagger-ui.html>



- mall-search的api接口文档地址: <http://虚拟机IP:8081/swagger-ui.html>



- mall-portal的api接口文档地址：<http://虚拟机IP:8085/swagger-ui.html>

