

- 一、Docker手动部署jar包
- 二、将镜像推送到Docker仓库，实现镜像共享
- 三、引入DockerCompose快速部署服务集群

## 五、引入Docker Compose快速搭建后端服务集群

**阶段目标：熟悉基于Docker的服务部署操作。快速搭建项目完整服务集群。**

分几个阶段逐步熟悉真实项目中的Docker操作。

开始Docker之前，先对项目中的配置信息进行调整。项目中相关配置参数，都统一整理到Nacos配置中心里。而server.port配置转移到项目本地，以便于后面操作。

补充修复一个小BUG。tl-live-im-server模块增加Dubbo服务后，Dubbo的协议实例也会往nacos上注册。这样，当tl-live-im-api模块给前端分配websocket地址时，就有可能选到dubbo注册的实例，从而造成前端建立websocket失败。

解决方法：在tl-live-im-api模块分配websocket服务地址时，去掉Dubbo协议注册的那些实例。

# 一、Docker手动部署jar包

这一阶段，还是将项目打包成Jar包。并通过Docker进行项目集群部署。通常用于程序员自检。

## 1、编译可执行Jar包

在pom.xml中增加引入spring-boot-maven-plugin，将项目编译成可执行jar包。

```
<build>
  <plugins>
    <plugin>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-maven-plugin</artifactId>
      <version>3.0.4</version>
      <executions>
        <execution>
          <goals>
            <goal>repackage</goal>
          </goals>
        </execution>
      </executions>
    </plugin>
  </plugins>
</build>
```

编译出的jar包可以使用java -jar 指令直接运行。运行前，只需要在系统的hosts文件中增加域名映射即可。在同一个服务器上进行多实例部署时，也可以通过指定-Dserver.port解决端口冲突的问题。

但是这种方式对资源占用比较多，应用每次重启都需要重复运行指令。如果引用Docker就可以更方便部署。

## 2、打包Docker镜像

编写Dockerfile，构建镜像。以tl-live-id-generate模块为例。

```
FROM openjdk:17
VOLUME /tmp
ADD tl-live-id-generate-provider-1.1.jar app.jar
ENV JAVA_OPTS="\
-server \
-Xmx1g \
-Xms1g \
-Xmn256m"
EXPOSE 8084
ENTRYPOINT java ${JAVA_OPTS} -jar app.jar
```

将Dockerfile上传到服务器，并将编译产生的tl-live-id-generate-provider-1.1.jar文件放到Dockerfile的同一个目录。接下来就可以使用Docker指令将jar包打包成image镜像。

```
# 打包docker镜像
docker build -t tl-live-id-generate-provider:1.1 .
# 查看镜像
docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
tl-live-id-generate-provider	1.1	7ed7824c2865	6 hours ago	527MB

### 3、创建Container，运行镜像

打包成镜像后，基于Docker的良好设计，可以快速启动多个节点，形成单机伪集群。

```
# 运行三个实例
docker run -d --add-host nacos.tllive.com:192.168.65.212 -p 8084:8084 --name tl-live-id-generate-node1 tl-live-id-generate-provider:1.1

docker run -d --add-host nacos.tllive.com:192.168.65.212 -p 8184:8084 --name tl-live-id-generate-node2 tl-live-id-generate-provider:1.1

docker run -d --add-host nacos.tllive.com:192.168.65.212 -p 8284:8084 --name tl-live-id-generate-node3 tl-live-id-generate-provider:1.1
# 查看服务状态
docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED
eac44b6cbe85	tl-live-id-generate-provider:1.1	"/bin/sh -c 'java \$..."	2 minutes ago
Up 2 minutes	0.0.0.0:8284->8084/tcp, :::8284->8084/tcp		tl-live-id-generate-node3
dcea8d0254c9	tl-live-id-generate-provider:1.1	"/bin/sh -c 'java \$..."	2 minutes ago
Up 2 minutes	0.0.0.0:8184->8084/tcp, :::8184->8084/tcp		tl-live-id-generate-node2
fee3b71e4d06	tl-live-id-generate-provider:1.1	"/bin/sh -c 'java \$..."	4 minutes ago
Up 4 minutes	0.0.0.0:8084->8084/tcp, :::8084->8084/tcp		tl-live-id-generate-node1

```
# 后续如果服务崩溃，需要重启服务，只要重新启动container就可以了。
# 查看日志使用 docker logs ${CONTAINER_ID} 如果需要采集日志，可以通过docker的-v参数，将日志文件挂载到宿主机的不同目录上。
```

docker镜像的方式启动服务，只能在本地进行测试。并且，需要多次重复控制单个镜像。对镜像的启动顺序也完全需要手动控制，不灵活。这种方式通常适用于程序员自检的阶段。

## 二、将镜像推送到Docker仓库，实现镜像共享

程序员自检完成后，要将镜像推送到正式服务器上，这时就需要基于镜像仓库进行共享。

这里演示使用阿里云的免费Docker仓库的过程。

### 1、需要在阿里云创建一个个人镜像仓库

阿里云地址：<https://www.aliyun.com/>

登录阿里云，找到容器服务，实例列表中可以创建一个免费的个人实例。



进入个人实例，在下面创建镜像仓库，并配置访问凭证



### 2、将本地镜像推送到镜像仓库中

进入进项仓库的管理页面，就可以参照页面上的操作指南，将本地Docker镜像推送到镜像仓库中。

一个仓库存放同一个docker镜像，只是可以包含多个不同的版本。

## ← tl-live-id-generate

### 基本信息

### 触发器

### 镜像版本

### 基本信息

### 编辑

仓库名称	tl-live-id-generate <a href="#">复制</a>
仓库地域	华东1（杭州）
仓库类型	私有
代码仓库	无

公网地址	registry.cn-hangzhou.aliyuncs.com/tearwind/tl-live-id-generate <a href="#">复制</a>
专有网络	registry-vpc.cn-hangzhou.aliyuncs.com/tearwind/tl-live-id-generate <a href="#">复制</a>
经典网络	registry-internal.cn-hangzhou.aliyuncs.com/tearwind/tl-live-id-generate <a href="#">复制</a>
摘要	仿抖音直播项目ID生成模块

### 操作指南

### 制品描述

#### 1. 登录阿里云Docker Registry

```
$ docker login --username=royk***@163.com registry.cn-hangzhou.aliyuncs.com
```

用于登录的用户名为阿里云账号全名，密码为开通服务时设置的密码。

您可以在访问凭证页面修改凭证密码。

#### 2. 从Registry中拉取镜像

```
$ docker pull registry.cn-hangzhou.aliyuncs.com/tearwind/tl-live-id-generate:[镜像版本号]
```

#### 3. 将镜像推送到Registry

```
$ docker login --username=royk***@163.com registry.cn-hangzhou.aliyuncs.com
$ docker tag [ImageId] registry.cn-hangzhou.aliyuncs.com/tearwind/tl-live-id-generate:[镜像版本号]
$ docker push registry.cn-hangzhou.aliyuncs.com/tearwind/tl-live-id-generate:[镜像版本号]
```

请根据实际镜像信息替换示例中的[ImageId]和[镜像版本号]参数。

#### 4. 选择合适的镜像仓库地址

这里输入的密码不是阿里云账号的密码，而是在个人Docker镜像中设定的访问凭证

### # docker登录

```
[root@192-168-65-212 docker]# docker login --username=roykingw@163.com registry.cn-hangzhou.aliyuncs.com
```

Password:

WARNING! Your password will be stored unencrypted in /root/.docker/config.json.  
Configure a credential helper to remove this warning. See  
<https://docs.docker.com/engine/reference/commandline/login/#credentials-store>

Login Succeeded

### # 推送镜像

```
[root@192-168-65-212 docker]# docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
tl-live-id-generate-provider	1.1	7ed7824c2865	2 days ago	527MB

```
[root@192-168-65-212 docker]# docker tag 7ed7824c2865 registry.cn-hangzhou.aliyuncs.com/tearwind/tl-live-id-generate:1.1
```

```
[root@192-168-65-212 docker]# docker push registry.cn-hangzhou.aliyuncs.com/tearwind/tl-live-id-generate:1.1
```

The push refers to repository [registry.cn-hangzhou.aliyuncs.com/tearwind/tl-live-id-generate]

```
[root@192-168-65-212 docker]# docker images
```

REPOSITORY	TAG	IMAGE ID
tl-live-id-generate-provider	1.1	7ed7824c2865
2 days ago	527MB	
registry.cn-hangzhou.aliyuncs.com/tearwind/tl-live-id-generate	1.1	7ed7824c2865
2 days ago	527MB	

推送完成后，在阿里云的镜像版本中可以查到刚推送的镜像版本。

## 三、引入DockerCompose快速部署服务集群

镜像推送到docker仓库后，就可以在服务器上同样通过docker pull指令拉取镜像。

但是，当我们把仿抖音项目的各个模块都上传后，在服务器上就会需要同时部署多个docker镜像。如果全都用docker pull的指令一个个部署，会非常麻烦。并且镜像启动的前后顺序也无法保证。这时，就可以使用DockerCompose组件来对Docker服务进行编排，让多个Docker镜像按照一定的顺序一起启动。

docker compose安装参见：docker compose官方地址：

<https://docs.docker.com/compose/install/linux/>

在之前安装docker时，我们已经一起安装了docker compose。

```
[root@192-168-65-212 docker]# docker compose version
Docker Compose version v2.27.0
```

在服务端，编辑docker-compose.yml

```
#version: '3'
services:
  tl-live-id-generate-1:
    container_name: tl-live-id-generate-1
    image: 'registry.cn-hangzhou.aliyuncs.com/tearwind/tl-live-id-generate:1.1'
    ports:
      - "8084:8084"
    volumes:
      - /tmp/logs/tl-live-id-generate-provider:/tmp/logs/tl-live-id-generate-provider
    environment:
      - server.port=8081
      - JAVA_OPTS=-XX:MetaspaceSize=128m -XX:MaxMetaspaceSize=128m -Xms512m -Xmx512m -
Xmn128m -Xss256k
    extra_hosts:
      - 'nacos.tllive.com:192.168.65.212'
  tl-live-id-generate-2:
    container_name: tl-live-id-generate-2
    image: 'registry.cn-hangzhou.aliyuncs.com/tearwind/tl-live-id-generate:1.1'
    ports:
      - "8094:8094"
    volumes:
      - /tmp/logs/tl-live-id-generate-provider:/tmp/logs/tl-live-id-generate-provider
    environment:
      - server.port=8094
      - JAVA_OPTS=-XX:MetaspaceSize=128m -XX:MaxMetaspaceSize=128m -Xms512m -Xmx512m -
Xmn128m -Xss256k
    extra_hosts:
      - 'nacos.tllive.com:192.168.65.212'
```

然后在docker-compose.yml文件的同目录，启动docker compose服务，就可以拉取镜像并按照顺序启动对应的docker 服务。

```
[root@192-168-65-212 docker]# docker compose up -d
[+] Running 2/2
 ✓ Container tl-live-id-generate-1 Started
3.6s
 ✓ Container tl-live-id-generate-2 Started
3.5s
```

这样就完成了tl-live-id-generate模块的集群化部署。

之后，将所有的模块都按照这个方式进行整合，就可以实现一套基于Docker的一键快速部署方案了。

程序员开发完成后，不需要提交jar包，直接提交一个新版本的Docker镜像就可以了。

而服务端可以一键完成整个集群的部署。

docker compose只能操作当前docker上的镜像，也就是说，docker只能实现单服务器上的镜像部署。如果需要将多个模块分别部署到不同宿主机上，那么用docker就只能拆分docker-compose服务了。如果需要更复杂的镜像编排，就需要使用k8s等更大型的镜像管理工具了。