

# 尚品汇商城复习

版本：V 1.0

## 自动化部署

### 一、环境准备

说明：如果已经安装过相关工具就忽略

#### 1 安装 JAVA 运行环境

第一步：上传或下载安装包

```
cd /usr/local
```

```
jdk-8u152-linux-x64.tar.gz
```

并删除原有的 JDK!

```
[root@localhost local]# rpm -qa | grep jdk
```

```
copy-jdk-configs-2.2-3.el7.noarch
```

```
java-1.8.0-openjdk-1.8.0.131-11.b12.el7.x86_64
```

```
java-1.8.0-openjdk-headless-1.8.0.131-11.b12.el7.x86_64
```

```
[root@localhost local]# rpm -e --nodeps java-1.8.0-openjdk-1.8.0.131-11.b12.el7.x86_64
```

```
[root@localhost local]# rpm -e --nodeps java-1.8.0-openjdk-headless-1.8.0.131-11.b12.el7.x86_64
```

第二步：解压安装包

```
tar -zxvf jdk-8u152-linux-x64.tar.gz
```

第三步：建立软连接

```
ln -s /usr/local/jdk1.8.0_152/ /usr/local/jdk
```

第四步：修改环境变量

```
vim /etc/profile
```

```
export JAVA_HOME=/usr/local/jdk
```

```
export JRE_HOME=$JAVA_HOME/jre
```

```
export CLASSPATH=.:$CLASSPATH:$JAVA_HOME/lib:$JRE_HOME/lib
```

```
export PATH=$PATH:$JAVA_HOME/bin:$JRE_HOME/bin
```

通过命令 `source /etc/profile` 让 profile 文件立即生效

```
source /etc/profile
```

第五步、测试是否安装成功

- ② 、使用 `javac` 命令，不会出现 `command not found` 错误
- ②、使用 `java -version`，出现版本为 `java version "1.8.0_152"`
- ③、`echo $PATH`，看看自己刚刚设置的的环境变量配置是否都正确

## 2 安装 maven

第一步：上传或下载安装包

```
cd /usr/local
```

```
apache-maven-3.5.4-bin.tar.gz
```

第二步：解压安装包

```
tar -zxvf apache-maven-3.5.4-bin.tar.gz
```

第三步：建立软连接

```
ln -s /usr/local/apache-maven-3.5.4/ /usr/local/maven
```

#### 第四步：修改环境变量

```
vim /etc/profile
```

```
export MAVEN_HOME=/usr/local/maven
```

```
export PATH=$PATH:$MAVEN_HOME/bin
```

通过命令 `source /etc/profile` 让 profile 文件立即生效

```
source /etc/profile
```

#### 第五步、测试是否安装成功

```
mvn -v
```

### 3 安装 git

```
yum -y install git
```

### 4 安装 docker

参考文档：

第一步：安装必要的一些系统工具

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

第二步：添加软件源信息

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

第三步：更新并安装 Docker-CE

```
yum makecache fast
```

```
yum -y install docker-ce
```

第四步：开启 Docker 服务

```
service docker start
```

第五步、测试是否安装成功

```
docker -v
```

## 5 安装 Jenkins

第一步：上传或下载安装包

```
cd /usr/local/jenkins
```

```
jenkins.war
```

第二步：启动、改端口、输出日志

```
nohup java -jar /opt/jenkins/jenkins.war --httpPort=8081 >/opt/jenkins/jenkins.out  
&
```

使用命令查看 jenkins 的运行信息：

```
ps -aux | grep jenkins.war
```

使用命令关闭 jenkins:

```
kill -9 pid
```

第三步：访问

<http://ip:8081>

## 二、初始化 Jenkins 插件和管理员用户

### 1 配置国内的镜像

[官方下载插件慢 更新下载地址](#)

```
cd {你的 Jenkins 工作目录}/updates #进入更新配置位置
```

```
sed -i 's/http:\\\\updates.jenkins-ci.org\\download/https:\\\\mirrors.tuna.tsinghua.edu.cn\\jenkins/g' default.json &&  
sed -i 's/http:\\\\www.google.com/https:\\\\www.baidu.com/g' default.json
```

这是直接修改的配置文件，如果前边 Jenkins 用 sudo 启动的话，那么这里的两个 sed 前均需要加上 sudo

重启 Jenkins，安装插件试试，简直超速！！

## 2 访问 jenkins

<http://ip:8080>

## 3 解锁 jenkins

```
cat /var/jenkins_home/secrets/initialAdminPassword
```

获取管理员密码 主要看箭头指向那个路径就 cat 那个路径就可以找到管理员密码了！



入门

### 解锁 Jenkins

为了确保管理员安全地安装 Jenkins，密码已写入到日志中（[不知道在哪里?](#)）该文件在服务器上：

`/var/jenkins_home/secrets/initialAdminPassword`

请从本地复制密码并粘贴到下面。

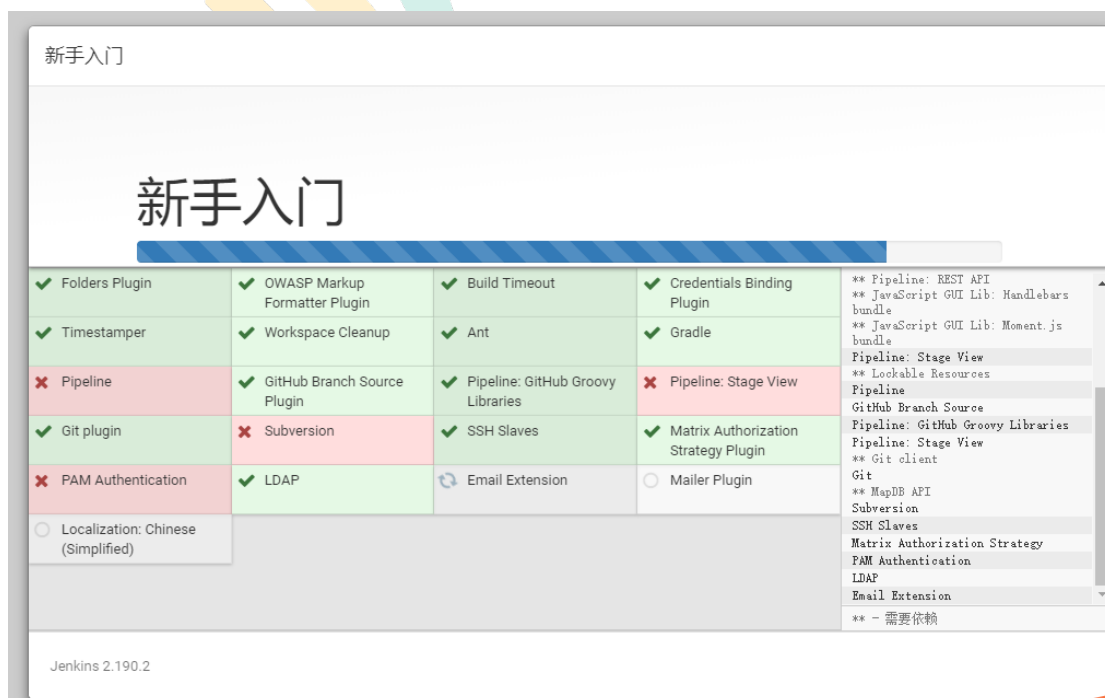
管理员密码

继续

## 4 选择“继续”



## 5 选择“安装推荐插件”



## 6 插件安装完成，创建管理员用户

新手入门

创建第一个管理员用户

Username:

test

Password:

....

Confirm password:

....

Full name:

test

E-mail address:

493290402@qq.com

Jenkins 2.190.2

使用admin账户继续

保存并完成

## 7 保存并完成

新手入门

实例配置

Jenkins URL:

http://47.93.118.241:8989/

Jenkins URL 用于给各种Jenkins资源提供绝对路径链接的根地址。这意味着对于很多Jenkins特色是需要正确设置的，例如：邮件通知、PR状态更新以及提供给构建步骤的BUILD\_URL环境变量。

推荐的默认值显示在尚未保存，如果可能的话这是根据当前请求生成的。最佳实践是要设置这个值，用户可能会需要用到。这将会避免在分享或者查看链接时的困惑。

Jenkins 2.190.2

现在不要

保存并完成

## 8 进入完成页面



说明：登录忘记密码参考 <http://www.pianshen.com/article/8676256341/>

## 三、配置 Jenkins 构建工具

### 全局配置



### 1 配置 jdk

JAVA\_HOME: /usr/local/jdk



## 全局工具配置

### Maven 配置

默认 settings 提供

默认全局 settings 提供

### JDK

#### JDK 安装

新增 JDK

JDK

别名

jdk1.8

JAVA\_HOME

/home/wg/jdk1.8.0\_121

☐ 自动安装



删除 JDK

新增 JDK

## 2 配置 maven

MAVEN\_HOME: /usr/local/maven

### Maven

#### Maven 安装

新增 Maven

Maven

Name

mvn

MAVEN\_HOME

/usr/local/apache-maven-3.6.1

☐ 自动安装



删除 Maven

新增 Maven

系统下Maven 安装列表

## 3 配置 git

查看 git 安装路径: which git

系统下JDK 安装列表

### Git

#### Git installations

Git

Name

git

Path to Git executable

/usr/bin/git

☐ 自动安装



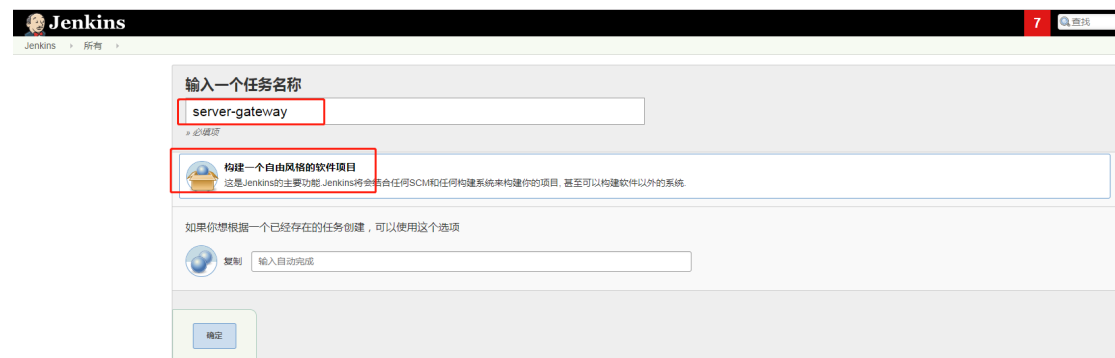
Delete Git

Add Git

## 四、构建作业 (service-product)

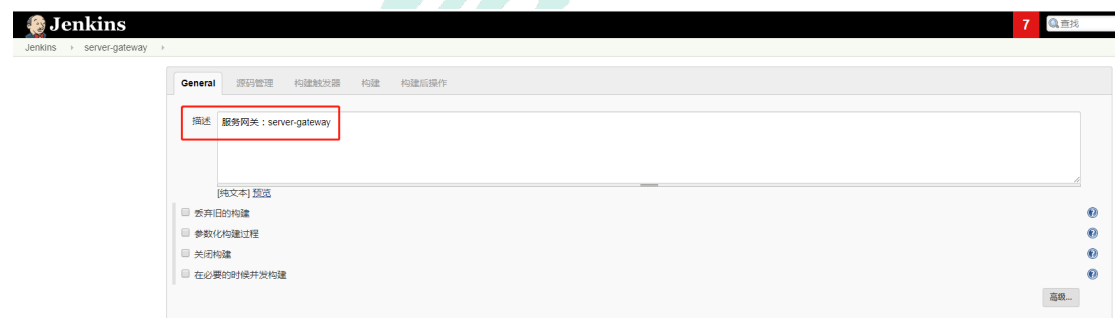
### 1 点击创建一个新任务，进入创建项目类型选择页面

我的视图中找新建任务



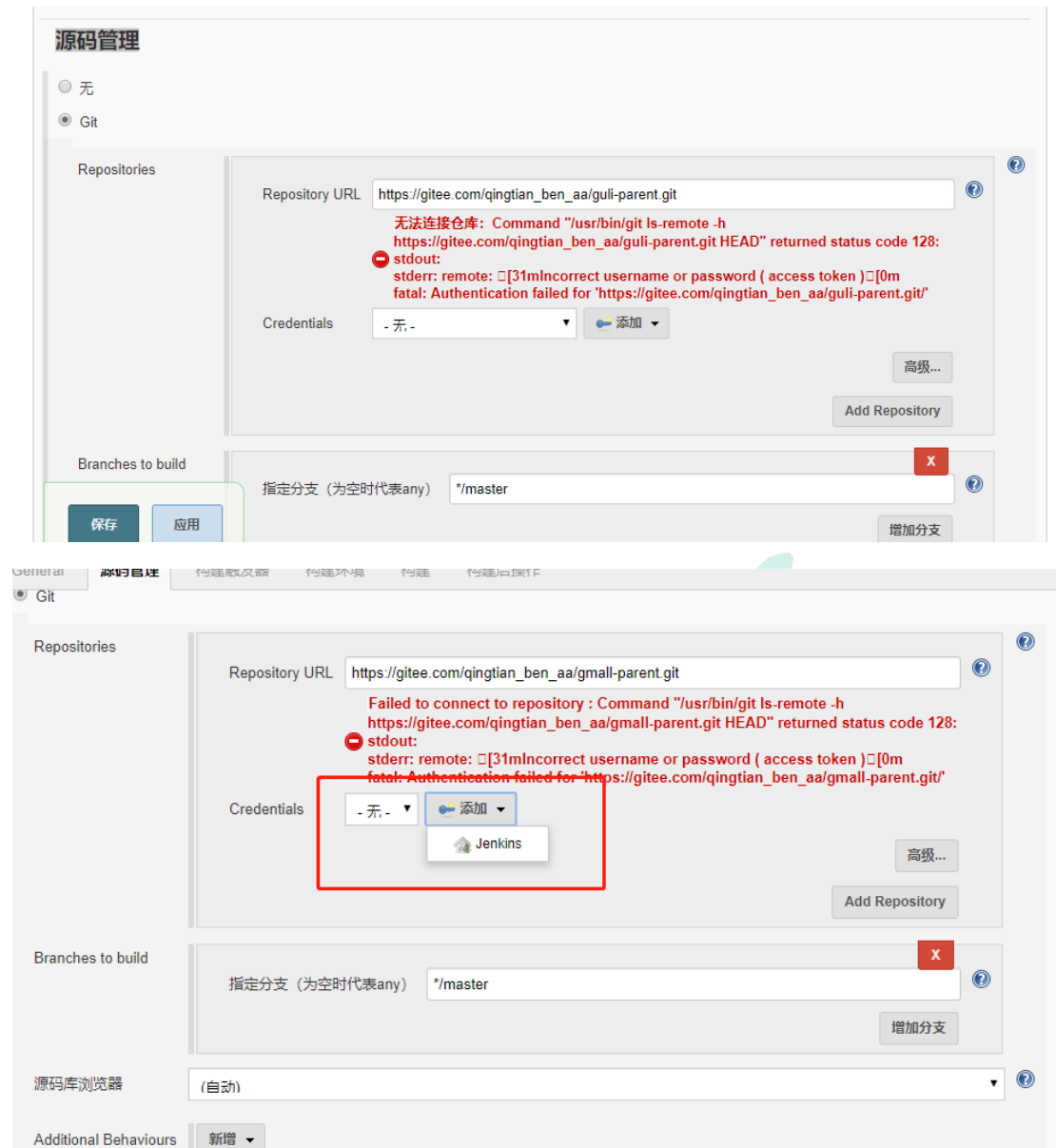
填好信息点击“确认”

### 2 配置“General”

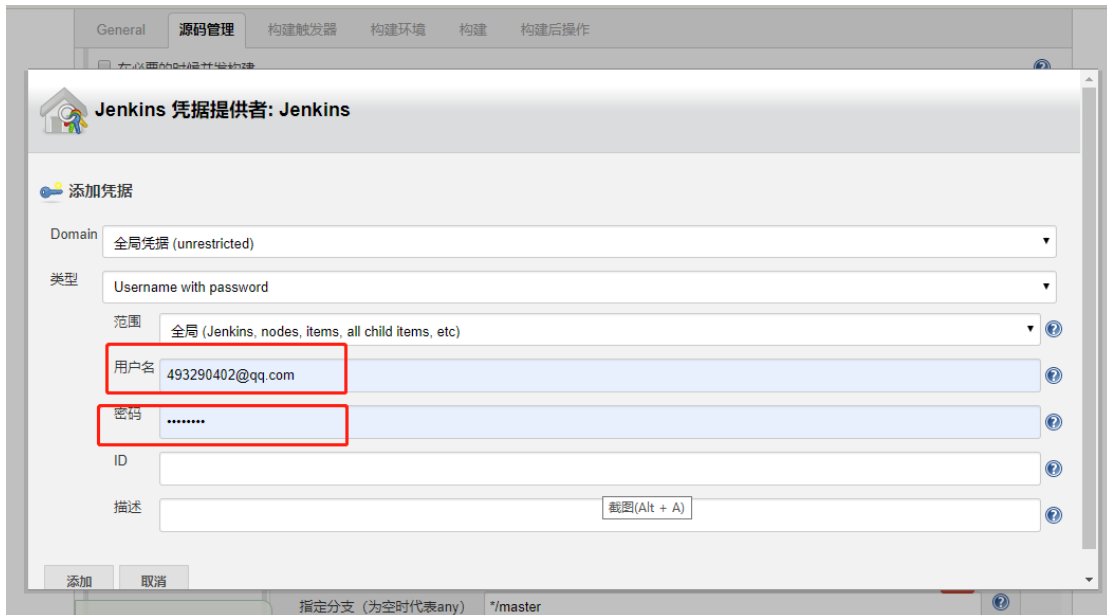


### 3 配置“源码管理”

填写源码的 git 地址 (small-parent 项目 git 地址)



点击添加 jenkins 添加 git 用户，git 的用户名与密码



General 源码管理 构建触发器 构建环境 构建 构建后操作

在必要的时候创建凭据

Jenkins 凭据提供者: Jenkins

添加凭据

Domain: 全局凭据 (unrestricted)

类型: Username with password

范围: 全局 (Jenkins, nodes, items, all child items, etc)

用户名: 493290402@qq.com

密码: \*\*\*\*\*

ID:

描述: 截图(Alt + A)

添加 取消

指定分支 (为空时代表any) \*/master

选择添加的用户，上面的红色提示信息消失，说明连接成功，如下图



源码管理

☐ 无

☒ Git

Repositories

Repository URL: https://gitee.com/qingtian\_ben\_aa/guli-parent.git

Credentials: 493290402@qq.com/\*\*\*\*\* 添加

高级...

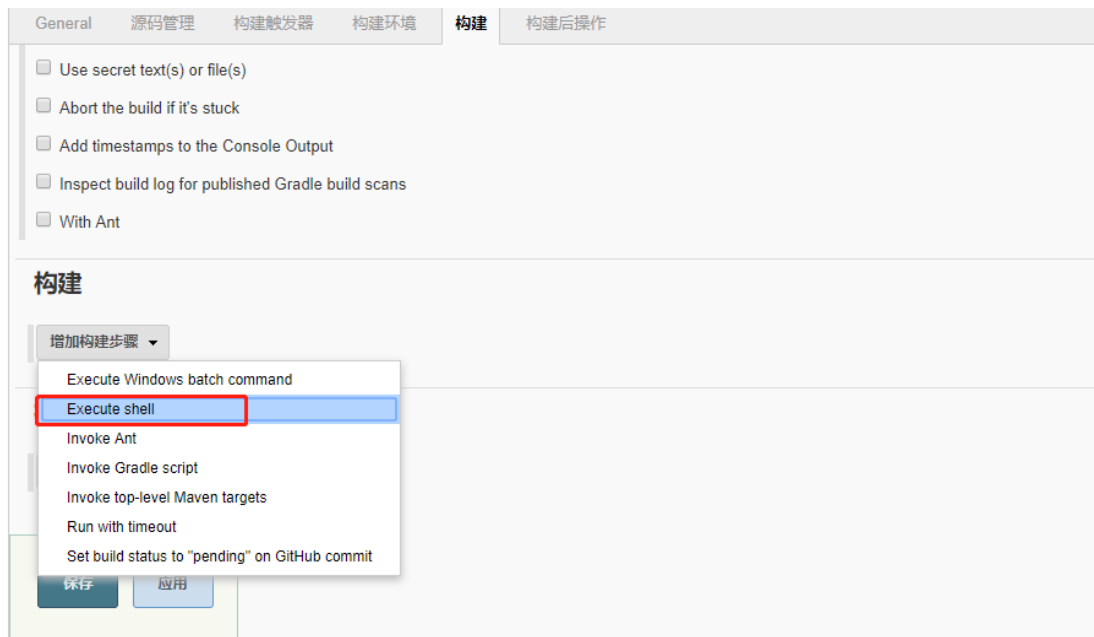
Add Repository

Branches to build

指定分支 (为空时代表any) \*/master

增加分支

## 4 构建作业



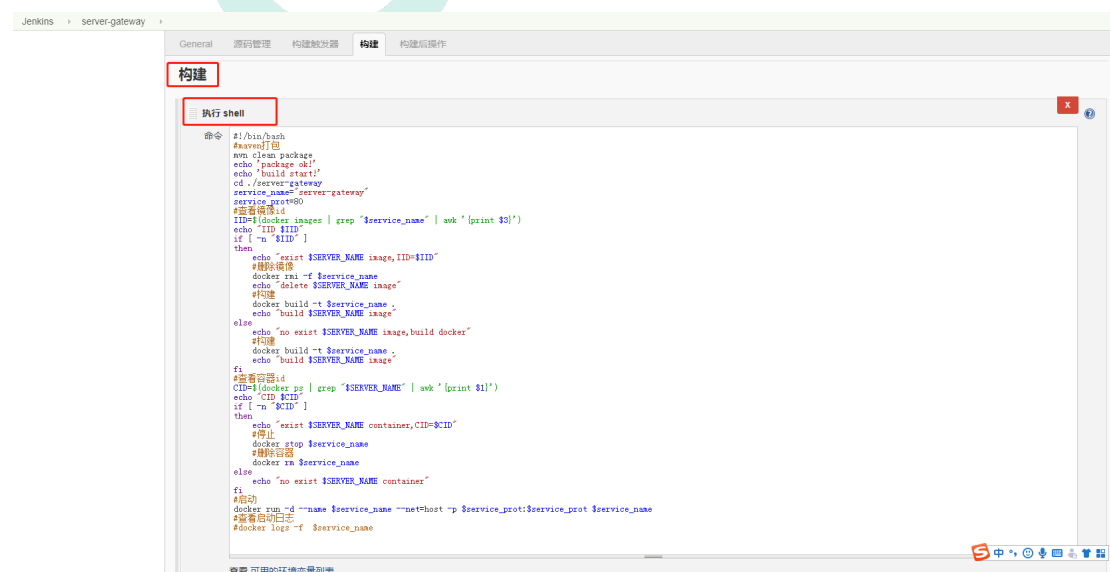
注：脚本文件 `gmall-parent\资料\sh\docker.sh`，其他服务都一样，只是调整以下三项配置：

`cd ./server-gateway` //当前目录为：gmall-parent

`service_name="server-gateway"` //当前服务名称：server-gateway

`service_prot=80` //当前服务端口：80

选择“执行 shell”



shell 命令如下：（该文件：gmall-parent\sh\docker.sh）

```
#!/bin/bash
#maven 打包
mvn clean package
echo 'package ok!'
echo 'build start!'
cd ./server-gateway
service_name="server-gateway"
service_prot=80
#查看镜像 id
IID=$(docker images | grep "$service_name" | awk '{print $3}')
echo "IID $IID"
if [ -n "$IID" ]
then
    echo "exist $SERVER_NAME image,IID=$IID"
    #删除镜像
    docker rmi -f $service_name
    echo "delete $SERVER_NAME image"
    #构建
    docker build -t $service_name .
    echo "build $SERVER_NAME image"
else
    echo "no exist $SERVER_NAME image,build docker"
    #构建
    docker build -t $service_name .
    echo "build $SERVER_NAME image"
fi
#查看容器 id
CID=$(docker ps | grep "$SERVER_NAME" | awk '{print $1}')
echo "CID $CID"
if [ -n "$CID" ]
then
    echo "exist $SERVER_NAME container,CID=$CID"
    #停止
    docker stop $service_name
    #删除容器
    docker rm $service_name
```

```
else
    echo "no exist $SERVER_NAME container"
fi
#启动
docker run -d --name $service_name --net=host -p
$service_prot:$service_prot $service_name
#查看启动日志
#docker logs -f $service_name
```

保存上面的构建作业



注：--net=host docker 的 4 种网络模式之一

host 模式，使用--net=host 指定。

container 模式，使用--net=container:NAME\_or\_ID 指定。

none 模式，使用--net=none 指定。

bridge 模式，使用--net=bridge 指定，默认设置。

参考文档：<https://www.cnblogs.com/gispathfinder/p/5871043.html>

## 5 构建

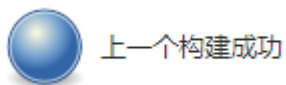
构建作业之后，就可以执行构建过程了。

## 5.1 执行构建过程

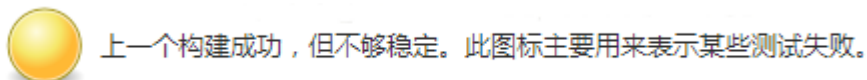


## 5.2 构建结构

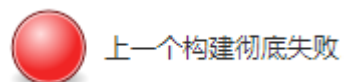
第一列是“上次构建状态显示”，是一个圆形图标，一般分为四种：



蓝色：构建成功；



黄色：不确定，可能构建成功，但包含错误；



红色：构建失败；





项目从未构建，或者被禁用

灰色：项目从未构建过，或者被禁用；

如上显示蓝色，表示构建成功。

注意：手动触发构建的时间与自动定时构建的时间互不影响。

## 5.3 查看控制台输出



日志内容：



注：其他模块构建方式一样，可选择“复制”上一个模块构建，修改配置即可，如图



```
#!/bin/bash
cd ./gmall-parent/service/service-product
service_name="service-product"
service_prot=8206
#查看镜像 id
IID=$(docker images | grep "$service_name" | awk '{print $3}')
echo "IID $IID"
if [ -n "$IID" ]
then
    echo "exist $SERVER_NAME image, IID=$IID"
    #删除镜像
    docker rmi -f $service_name
```

```
    echo "delete $SERVER_NAME image"

    #构建

    docker build -t $service_name .

    echo "build $SERVER_NAME image"
else
    echo "no exist $SERVER_NAME image,build docker"

    #构建

    docker build -t $service_name .

    echo "build $SERVER_NAME image"
fi

#查看容器 id
CID=$(docker ps | grep "$SERVER_NAME" | awk '{print $1}')
echo "CID $CID"
if [ -n "$CID" ]
then
    echo "exist $SERVER_NAME container,CID=$CID"

    #停止

    docker stop $service_name

    #删除容器

    docker rm $service_name
else
    echo "no exist $SERVER_NAME container"
fi

#启动

docker run -d --name $service_name --net=host -p
$service_prot:$service_prot $service_name

#查看启动日志
```

```
#docker logs -f $service_name
```

