尚品汇商城

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-3.6.1-bin.tar.gz

第二步：解压安装包

tar -zxvf apache-maven-3.6.1-bin.tar.gz

第三步：建立软连接

ln -s /usr/local/apache-maven-3.6.1/ /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

在/usr/local 目录下创建一个文件夹

mkdir jenkins

cd /usr/local/jenkins

jenkins.war 传入到/usr/local/jenkins

第二步：启动

nohup java -jar /usr/local/jenkins/jenkins.war >/usr/local/jenkins/jenkins.out &

第二步：访问

防火墙必须关闭：systemctl stop/disable firewalld

http://ip:8080

如果访问不到，则杀死进程，重启！

ps -ef | grep jenkins

kill -9 -pid

nohup java -jar /usr/local/jenkins/jenkins.war >/usr/local/jenkins/jenkins.out &

二、初始化 Jenkins 插件和管理员用户

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

`cd /root/.jenkins 有 updates 文件夹`

进入 updates 文件夹 然后执行下面这个命令！

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{给权限}

重启 Jenkins，安装插件试试，简直超速！！

ps -ef | grep jenkins

使用命令杀死进程

Kill -9 pid

nohup java -jar /usr/local/jenkins/jenkins.war >/usr/local/jenkins/jenkins.out &

1 访问 jenkins

http://ip:8080

2 解锁 jenkins

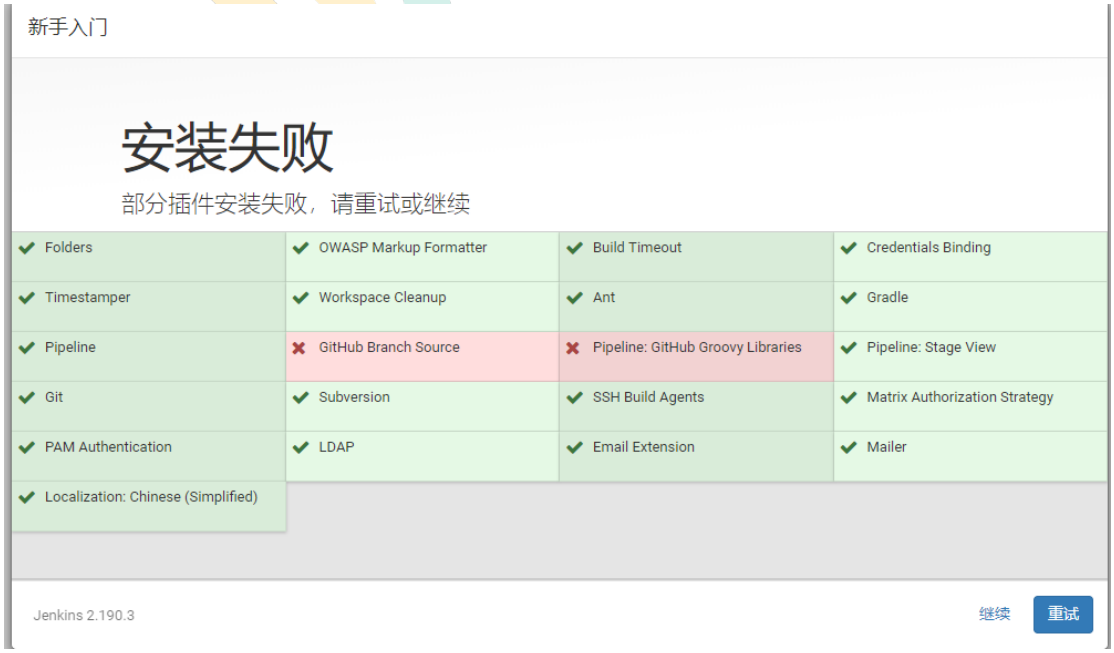
cat /var/jenkins_home/secrets/initialAdminPassword

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！

3 选择“继续”

4 选择“安装推荐插件”

如果你不配置国内的镜像，则这些下载速度非常慢！



如果出现错误：最好重试。

5 插件安装完成，创建管理员用户

6 保存并完成

7 进入完成页面

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

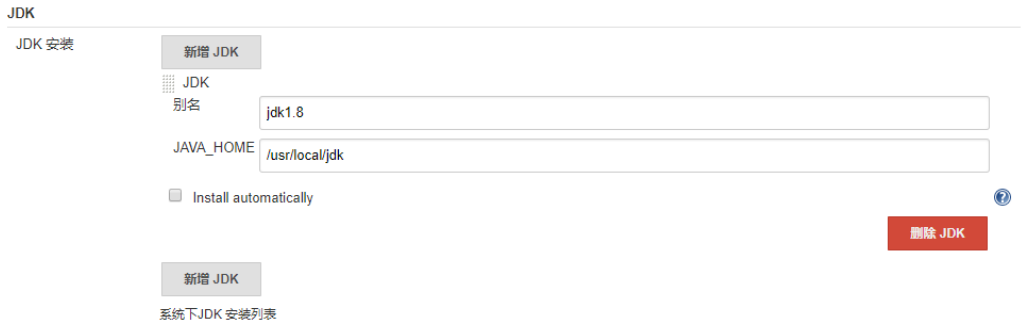
如果等待时间太长：那么我们可以将 jenkins 杀掉，重新启动！

三、配置 Jenkins 构建工具

全局配置

1 配置 jdk

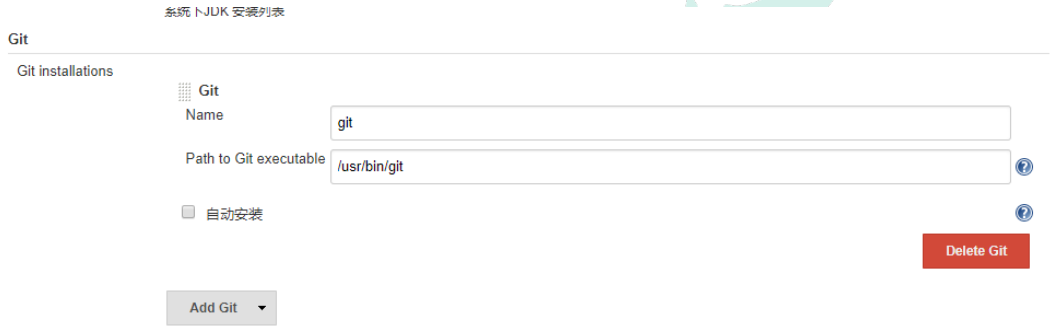
JAVA_HOME：/usr/local/jdk



2 配置 git

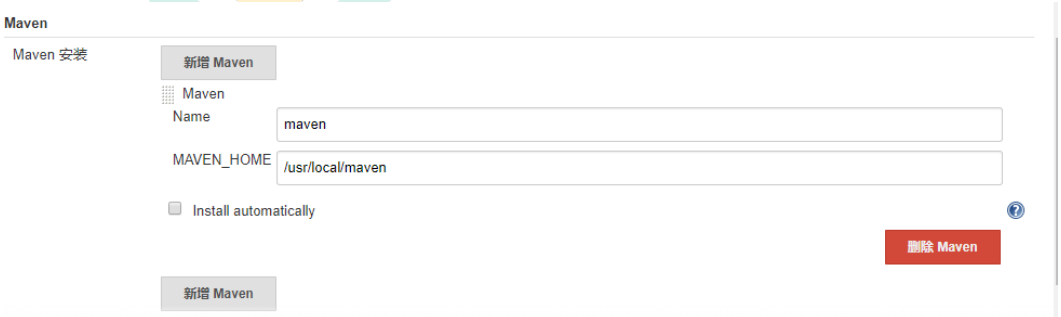
查看 git 安装路径：which git

/usr/bin/git



3 配置 maven

MAVEN_HOME：/usr/local/maven

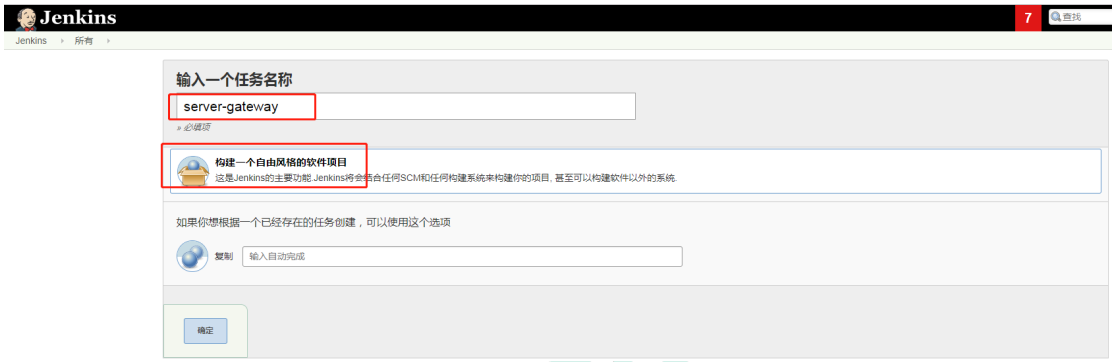


四、构建作业（server-gateway）

1 点击创建一个新任务，进入创建项目类型选择页面

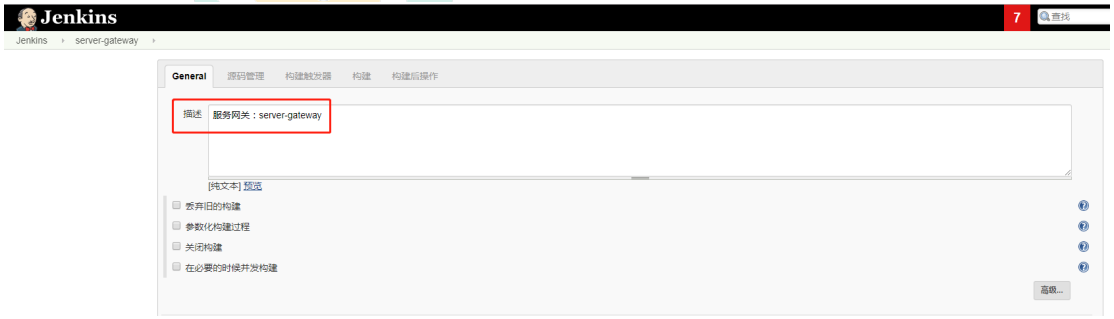
我的视图中找**新建任务**

server-gateway



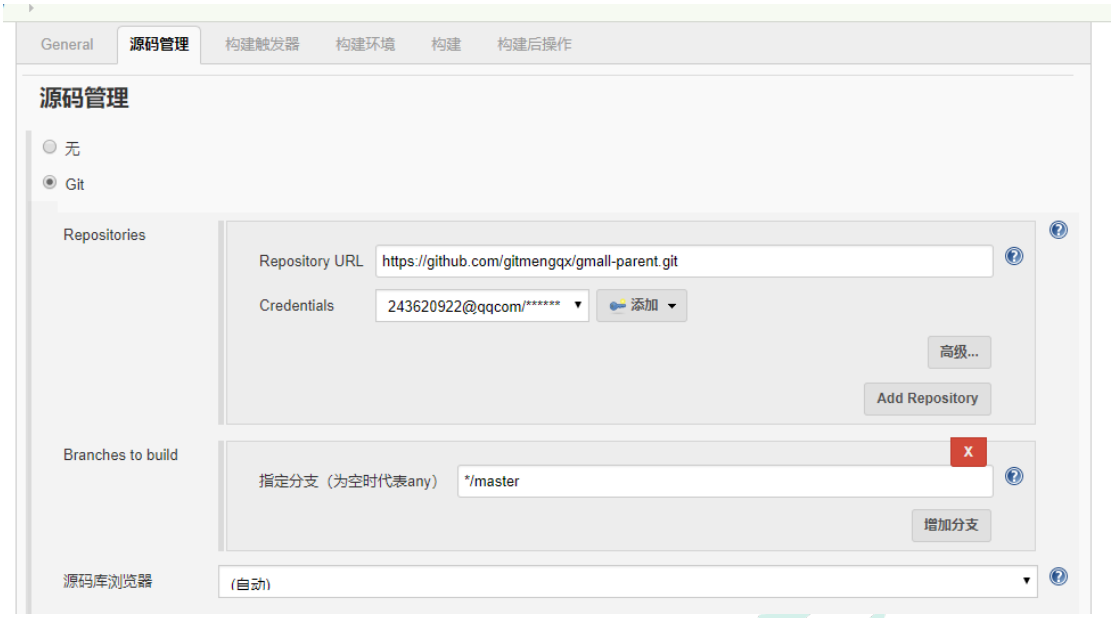
填好信息点击“确认”

2 配置“General”

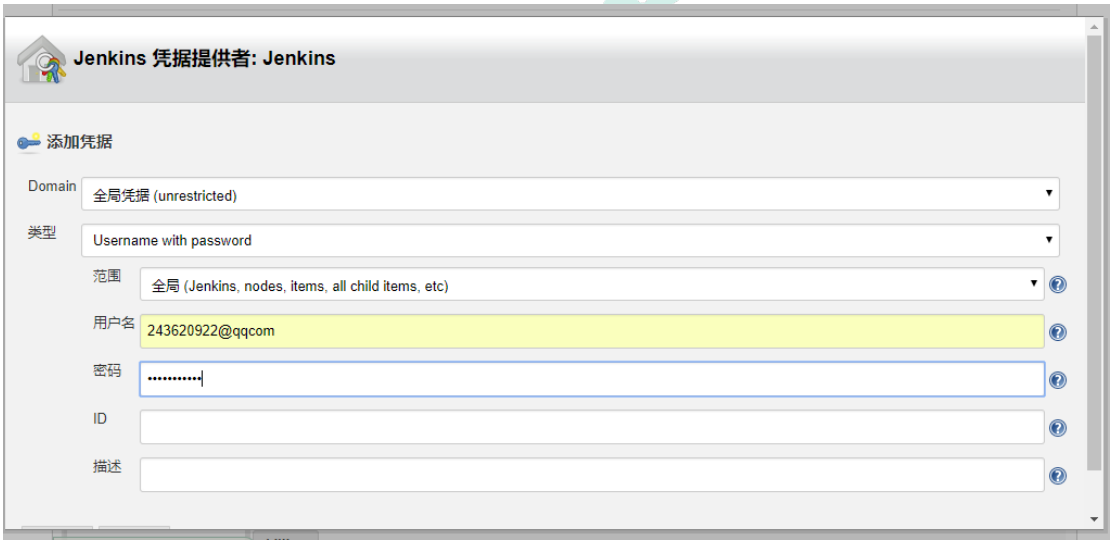


3 配置“源码管理”

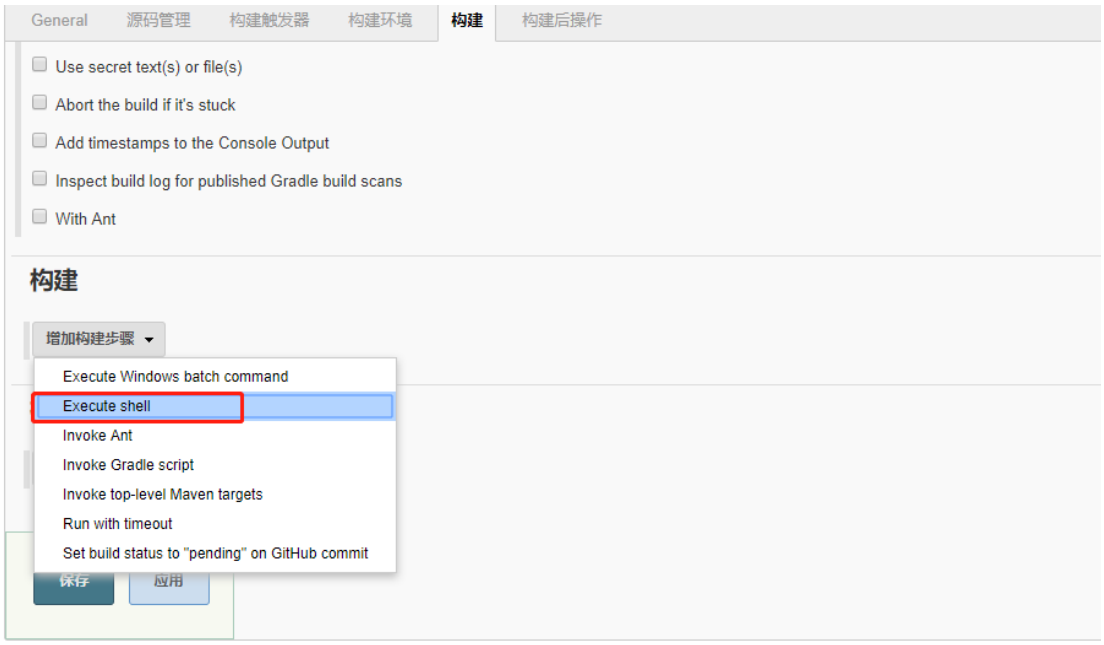
填写源码的 git 地址（gmall-parent 项目 git 地址）



点击添加 jenkins 添加 git 用户，git 的用户名与密码



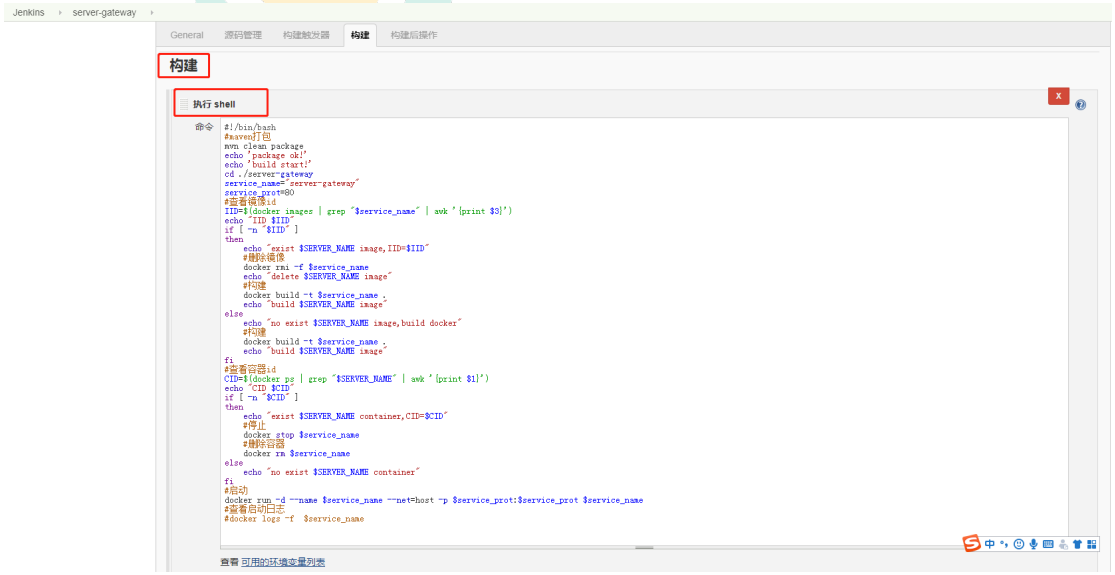
4 构建作业



注：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：

```
#!/bin/bash
#maven 打包
mvn clean package
echo 'package ok!'
echo 'build start!'
cd ./server-gateway
service_name="server-gateway"
service_port=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 $service_port:$service_port $service_name
#查看启动日志
#docker logs -f  $service_name
```

```
service-product:
#!/bin/bash
#maven 打包
mvn clean package
echo 'package ok!'
echo 'build start!'
cd ./service/service-product
service_name="service-product"
service_port=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 $service_port:$service_port $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。

使用宿主机的 ip 和端口

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：


上一个构建成功

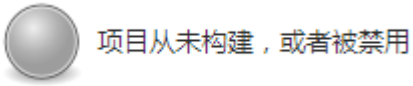
蓝色：构建成功；


上一个构建成功，但不够稳定。此图标主要用来表示某些测试失败。

黄色：不确定，可能构建成功，但包含错误；


上一个构建彻底失败

红色：构建失败；



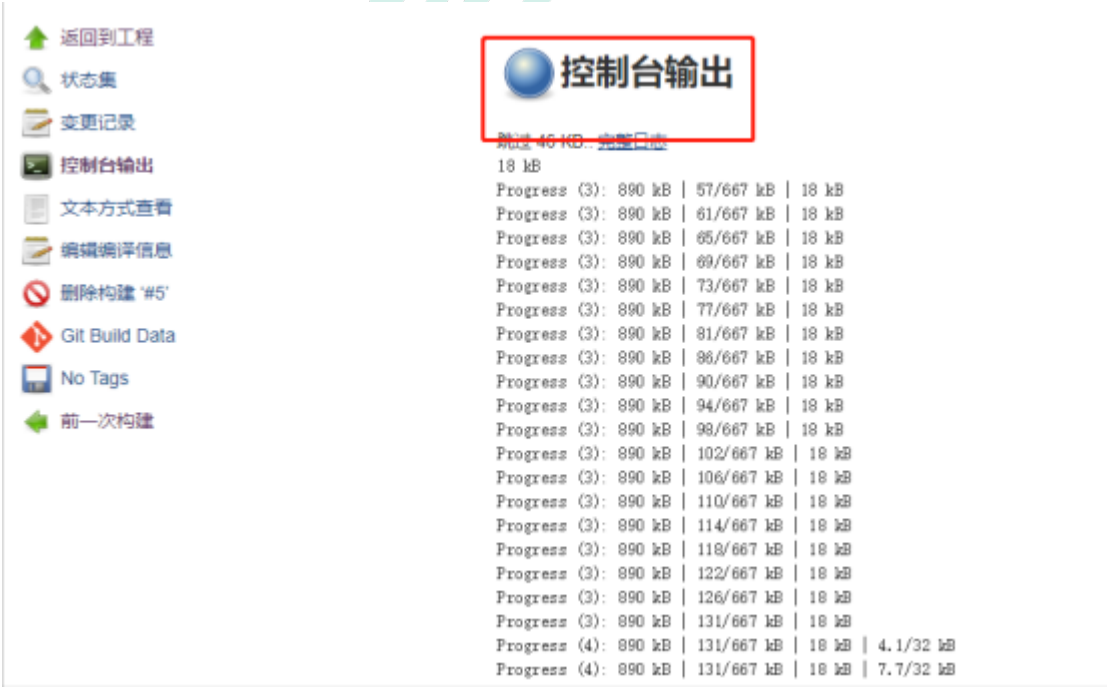
灰色：项目从未构建过，或者被禁用；

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


构建一个多配置项目
适用于多配置项目,例如多环境测试,平台指定构建,等等.
文件夹
创建一个可以嵌套存储的容器。利用它可以进行分组。 视图仅仅是一个过滤器，而文件夹则是一个独立的命名空间， 因此你可以
的内容，只要它们在不同的文件 夹里即可。
GitHub 组织
扫描一个 GitHub 组织（或者个人账户）的所有仓库来匹配已定义的标记。
多分支流水线
根据一个SCM仓库中检测到的分支创建一系列流水线。

如果你想根据一个已经存在的任务创建，可以使用这个选项
复制
server-gateway
确定