链滴

Jenkins 部署和踩坑总结

作者：CodingOX

原文链接：https://ld246.com/article/1612765490260

来源网站：链滴

基础安装

下载 Tomcat，目前不要选择 10+的版本，选择 9 比较好，下载地址：链接

下载 Jenkins，推荐下载Generic Java package (.war)，下载地址：链接

然后将下载后的 jar 包部署到 Tomcat 的 webapps目录下，然后重启。

Tomcat 的启动和关闭可能会涉及到权限问题，记得授权。

启动后，通过http://ip:8080/jenkins/进行访问，第一次速度有一些慢，然后就是获取密码，创建新户。

插件推荐

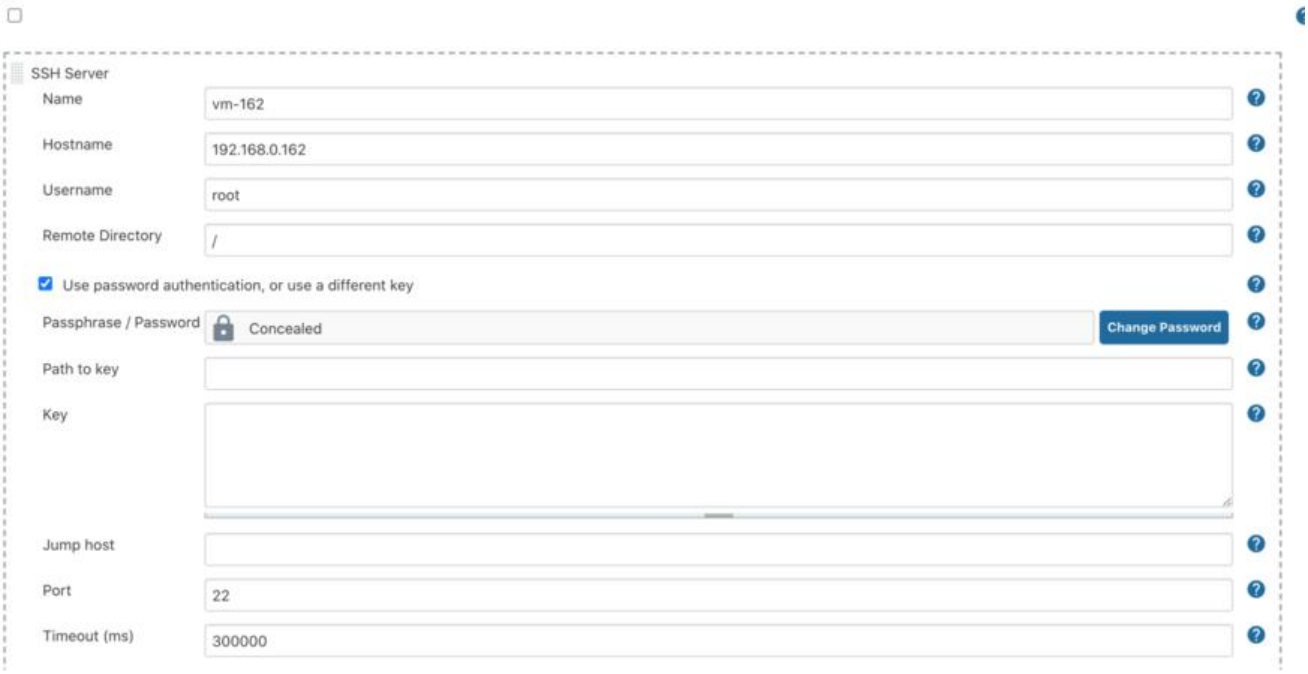
包括如下内容：

- Maven 相关插件： Maven Integration plugin
- Git插件： Git plugin
- 环境变量插件： Environment Injector
- 远程部署： Publish Over SSH
- 通用的 Webhook插件： Generic Webhook Trigger Plugin
- BlueOcean 服务编排： Blue Ocean 和Blue Ocean Pipeline Editor

配置相关

系统配置

- Jenkins Location这个地方建议修改为：http请求和补充自己的email
- Git glugin中的user.name和user.email修改
- 邮件通知建议使用QQ邮箱，163很容易识别为垃圾邮箱。
- Publish over ssh如果有用到，请在这里设置，通过 证书或者密码登录都是可以的，设置完成后，击Test Configuration测试下。



通用配置

系统管理-> 全局通用配置 中需要配置：Maven、JDK、Docker 等。下面通过 JDK 举例，其他按照己的实际情况配置，但都请不要勾选“自动安装”。

其他的类似。

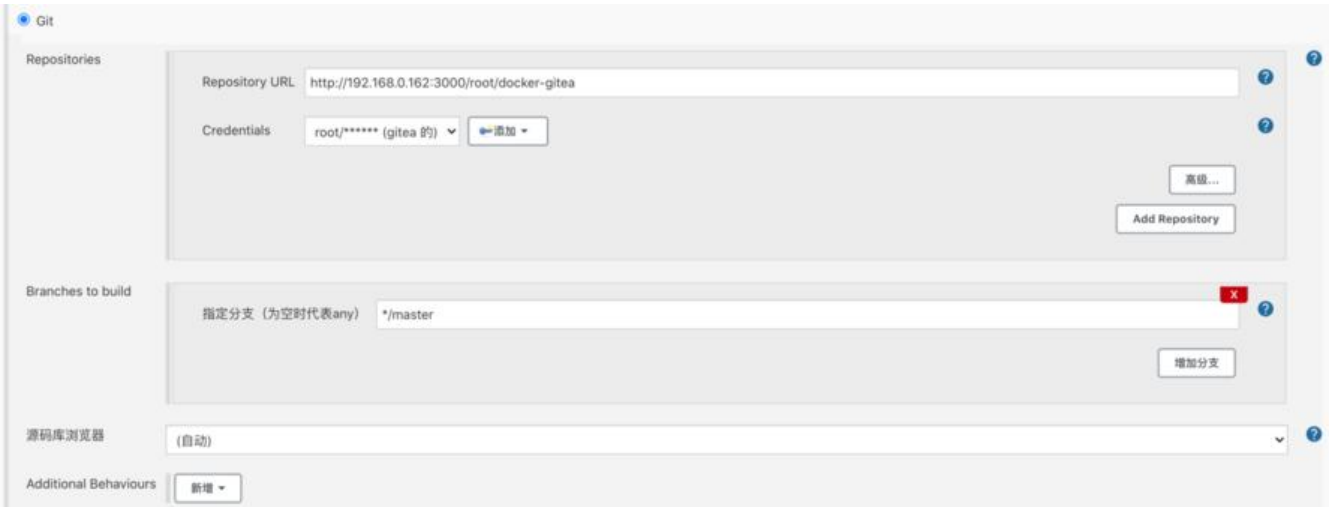
Maven项目

原文链接：Jenkins 部署和踩坑总结

本地部署

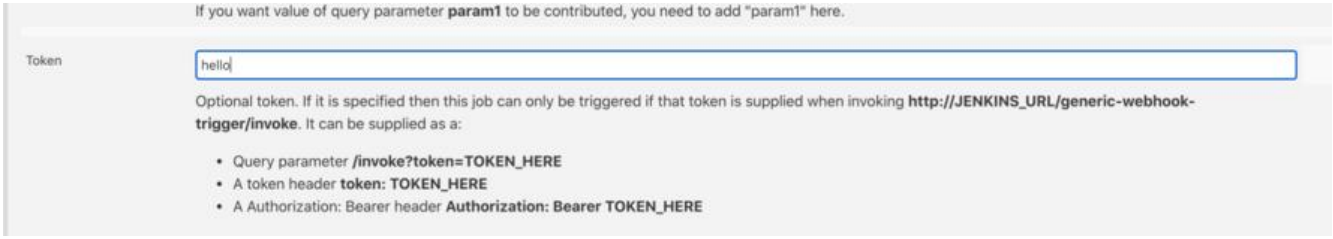
选择新建一个 maven 项目：

然后填写git的相关信息



那么这个流程何时开始编译，此时就需要通过 git项目的 webhook 了，下面这个选项是插件：Generi Webhook Trigger Plugin带来的。

原文链接：Jenkins 部署和踩坑总结



只需要填写 token，然后webhook 的时候发送相关指令即可，发送格式写的很清楚

- 通过带参数的形式： http://192.168.0.161:8080/jenkins/generic-webhook-trigger/invoke?token=hello
- 通过 header的形式等

然后设置 maven 的构建指令以及运行

强调几点：

- maven的指令，就是上面的Goals and options的开头不能提交mvn
- Post Steps表示的构建完成后的步骤，通常是进行项目启动

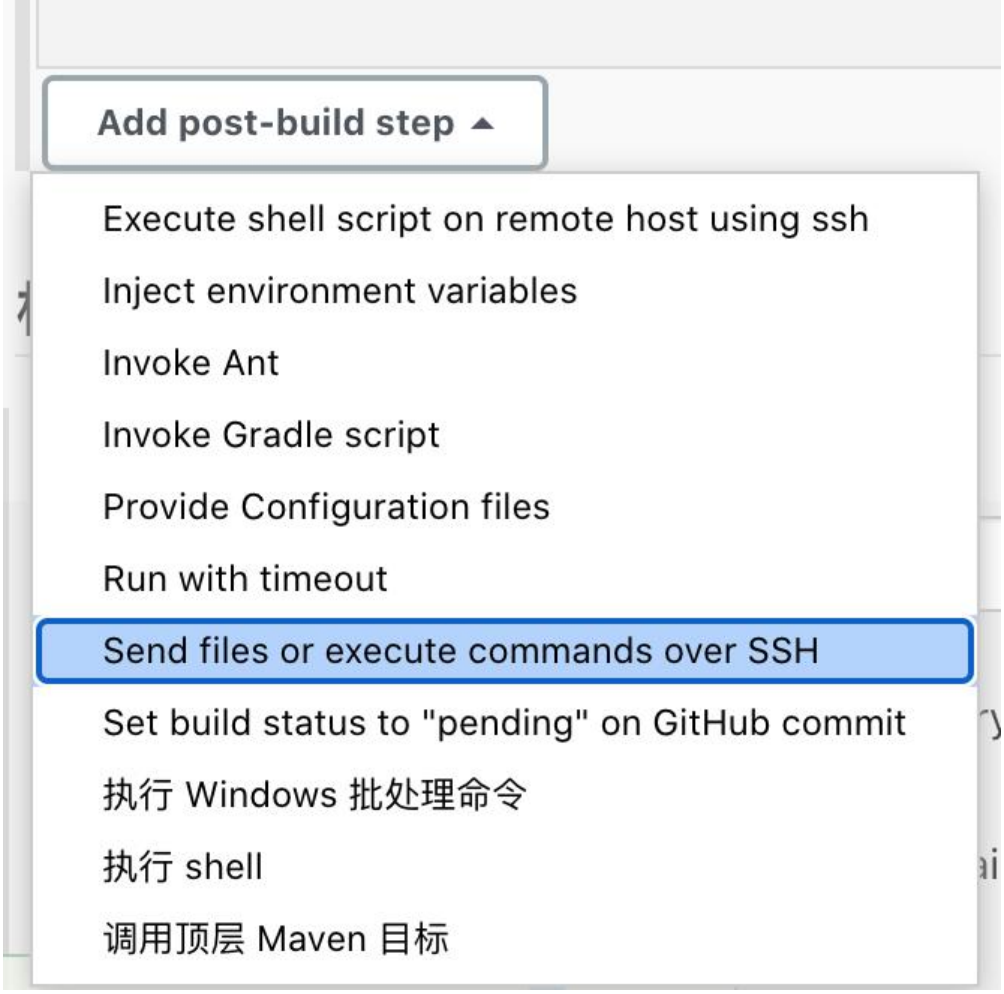
最后可以通过 email 进行消息通知。

远程部署

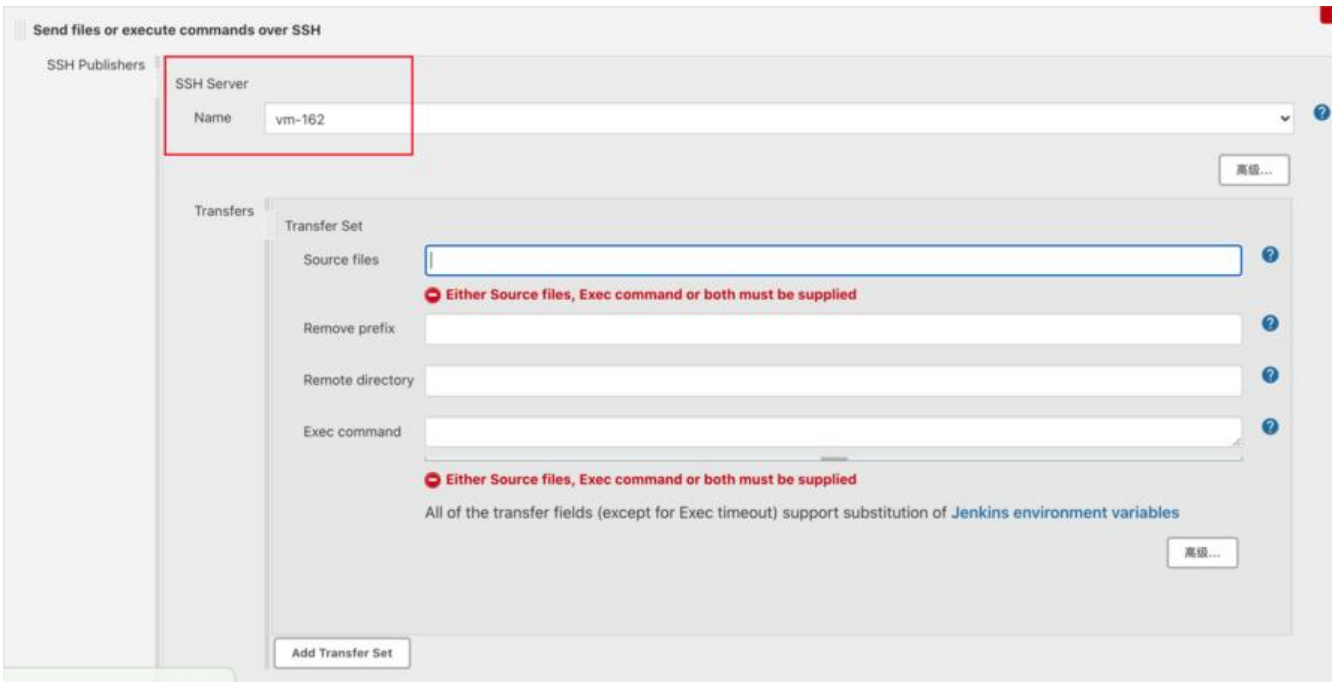
基本设置

通常项目不仅要部署到本机，还可能需要部署其他主机，此时有 2 种办法，一种是通过主机信任后通过scp传输，一种是通过插件Publish over ssh，我们先聊聊后者。

在Post Steps中新增Send files or execute commands over SSH，如下所示：



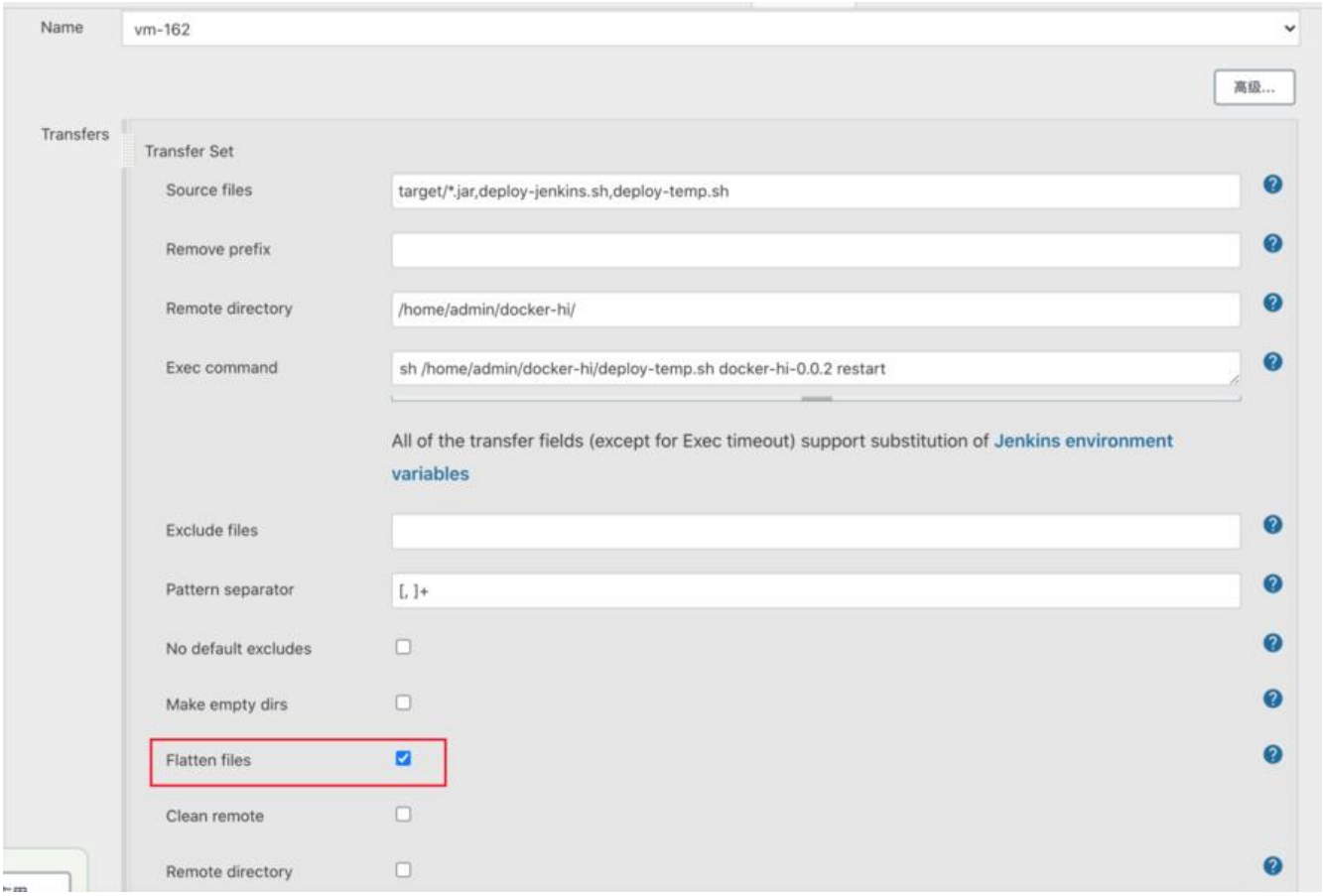
然后可以看到下图:



假如你没有 SSH Server 的设置，那是因为你没有在全局设置中进行配置，可以参考我上面插件推荐中的最后配置

原文链接：Jenkins 部署和踩坑总结

然后输入如下内容：



- Sourec File表示的是你要传输的文件，这个路径其实就是相对你项目的路径。Jenkins会从Git上拉取文件，然后在自己的 workspace中进行编译。
- Remove prefix表示的去除前缀，这里因为我勾选了Flattern files 所以不需要去除前缀
- Remote Directory表示的远程主机上的目录
- Exec command表示的是远程主机上执行的命令，通常是通过脚本进行启动、关闭、重启等
- Flattern files表示的去除前缀，就是将只传输文件到Remote Director，不带文件前缀

脚本文件

这个脚本文件，是最大的坑，查资料，卡了半天，总算再查另外一个问题的时候，查到了解决方案。

你可能会遇到问题：无法通过Jenkins 的这个插件使用脚本启动远程项目。同时你的日志中可能报错：RROR: Exception when publishing, exception message [Exec exit status not zero. Status [1]]

其实这个问题可能是因为：Jenkins 启动任务完成后，会关闭工作进程启动的子进程，所以远程任务关闭了【可能】

所以在写脚本的时候，需要注意 3 个细节：

- 文件开始加入： source /etc/profile
- 然后设置参数： export BUILD_ID=dontkillme
- 最后在获取进程的时候，排除到 Jenkins，比如我们 Java 应用这么写：$(ps -ef | grep -w "$SRVICE_NAME" | grep "java" | awk '{print $2}')

原文链接：Jenkins 部署和踩坑总结

下面给出一个案例脚本：

```
#!/bin/bash
# 搭配使用才行!
source /etc/profile
# jenkins 使用
export BUILD_ID=dontkillme


## exec shell name
EXEC_SHELL_NAME=$1\.sh
## service name
SERVICE_NAME=$1
SERVICE_DIR=/home/admin
JAR_NAME=$SERVICE_NAME\.jar
PID=$SERVICE_NAME\.pid
WORK_DIR=$SERVICE_DIR/docker-hi

mkdir -p $WORK_DIR

#function start
start() {
  cd $WORK_DIR
  if [ ! -d "log" ]; then
    mkdir log
  fi
#  nohup java  -Xms256m -Xmx512m -jar $JAR_NAME >log/$SERVICE_NAME.out 2>&1 &
  nohup java -Xms256m -Xmx512m -jar $JAR_NAME >log/$SERVICE_NAME.out 2>&1 &
  echo $! >$WORK_DIR/$PID
  echo "#### start $SERVICE_NAME"
}

# function stop x
stop() {
  cd $WORK_DIR
  if [ -f "$WORK_DIR/$PID" ]; then
    kill $(cat $WORK_DIR/$PID)
    rm -rf $WORK_DIR/$PID
  fi
  echo "#### stop $SERVICE_NAME"
  sleep 6
  TEMP_PID=$(ps -ef | grep -w "$SERVICE_NAME" | grep "java" | awk '{print $2}')
  if [ "$TEMP_PID" == "" ]; then
    echo "#### $SERVICE_NAME process not exists or stop success"
  else
    echo "#### $SERVICE_NAME process pid is:$TEMP_PID ."
    kill -9 $TEMP_PID
  fi
}

# function clean
clean() {
  cd $WORK_DIR
  if [ ! -d "lastDeploy" ]; then
```

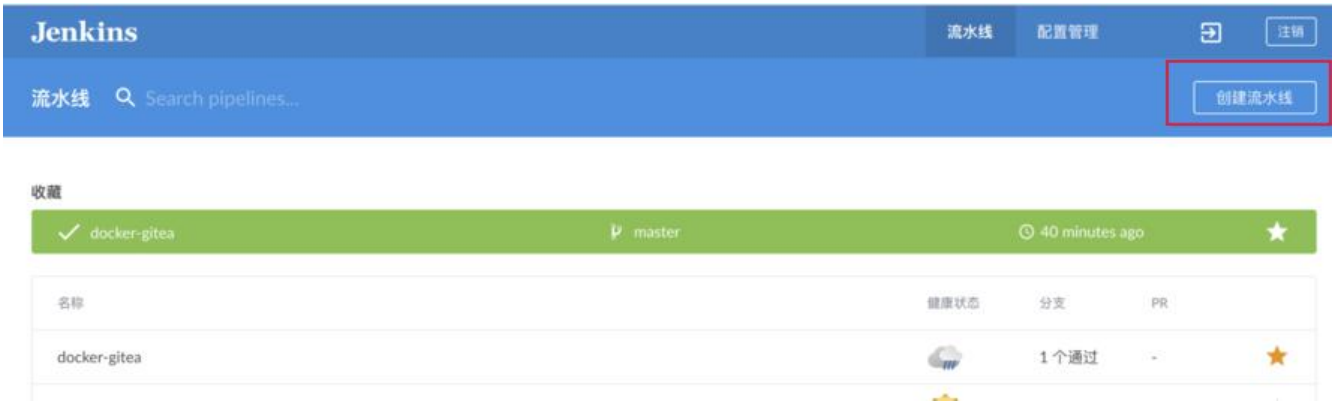
```
    mkdir lastDeploy
  else
    rm lastDeploy/$SERVICE_NAME*
  fi
  if [ -f "$JAR_NAME" ]; then
    mv $JAR_NAME lastDeploy
  fi
}

case "$2" in

start)
  start
  ;;

stop)
  stop
  ;;

restart)
  stop
  sleep 2
  start
  echo "#### restart $SERVICE_NAME"
  ;;

clean)
  stop
  sleep 2
  clean
  echo "#### clean $SERVICE_NAME"
  ;;

*)
  ## restart
  stop
  sleep 2
  start
  ;;

esac
exit 0
```

Blue Ocean

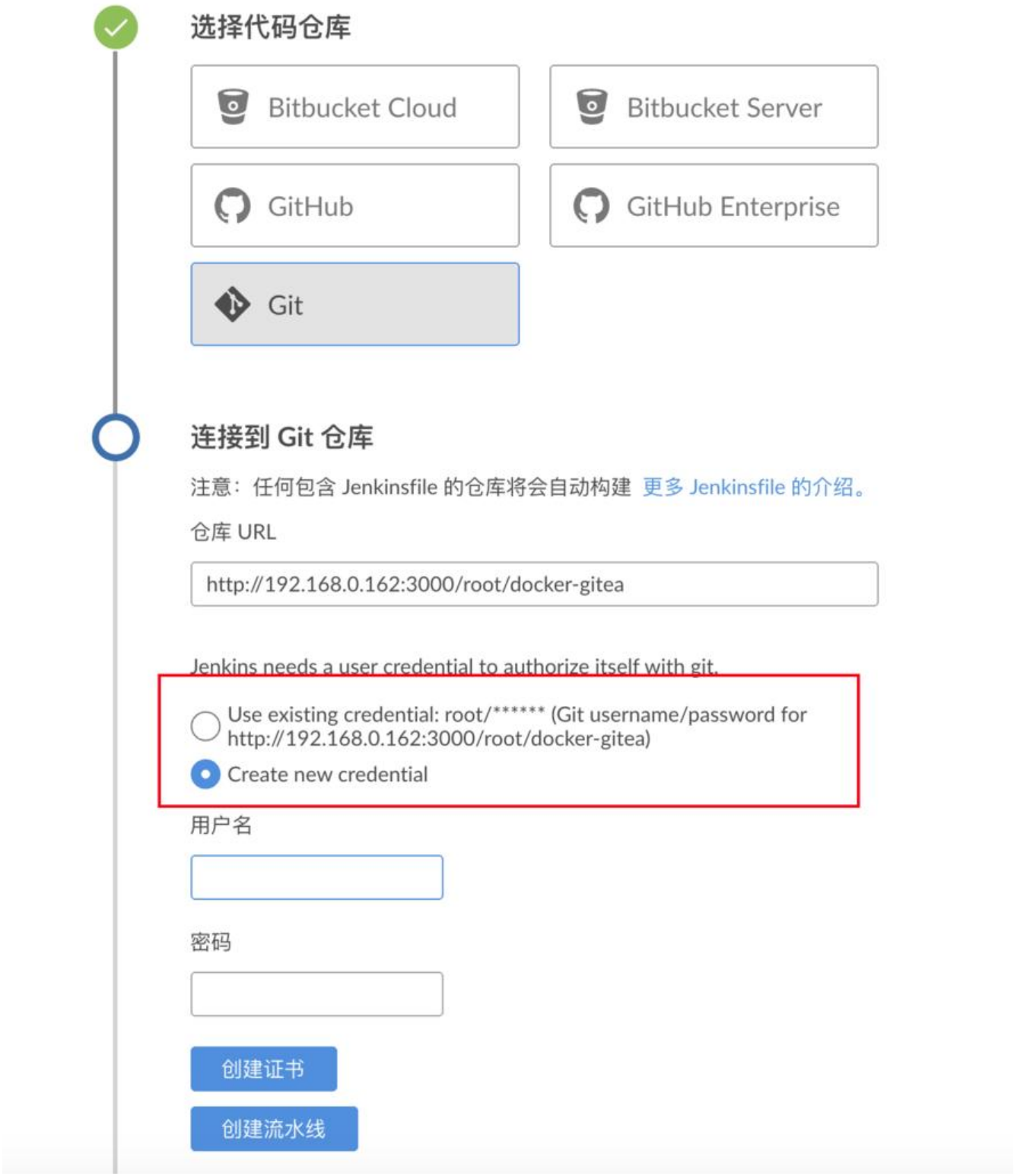
安装了插件Blue Ocean 和 Blue Ocean Pipeline Editor在项目面板的左边，会出现选项：打开 Blue cean


Dashboard
新建任务
用户列表
构建历史
项目关系
检查文件指纹
系统管理
我的视图
打开 Blue Ocean
Lockable Resources
新建视图

可以选择新建流水线



其中第一步就是选择 git 仓库：



输入地址后，会要求是使用已经创建好的凭证还是创建一个新的凭证，这个随你。

然后会要求你输入这个流水线的名称，比如blue，创建完成后，跳转到该界面

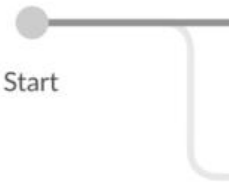
原文链接：Jenkins 部署和踩坑总结

简单说下上述面板的几个意思：

- 活动：表示的每一次构建任务的情况。
- 分支：这个概念其实就是 git的分支概念，本质上你可视化流程的操作，最终会在项目根路径下形一个文档jenkinsfile，并 通过git进行管理。

此时选择分支，然后将鼠标移动到 master分支，选择后面的笔表示。

接下来的流程就是创建，点击+进行流程创建，基本都是shell脚本，创建好后如下所示：



点击确定后提交到 git仓库。

最终生成的文件如下所示：

```
pipeline {
  agent any
  stages {
    stage('git') {
      steps {
        git(url: 'http://192.168.0.162:3000/root/docker-gitea.git', branch: 'master', credentialsId: '
itea')
      }
    }

    stage('maven') {
      steps {
        sh 'mvn -B clean package -Dmaven.test.skip=true -Dautoconfig.skip'
      }
    }
```

```
    stage('deploy') {
      parallel {
        stage('remote deploy-1') {
          steps {
            sh '''scp target/*.jar root@192.168.0.162:/home/admin/docker-hi/docker-hi-0.0.2.jar
ssh root@192.168.0.162  \'bash -s\' < /home/admin/docker-hi/deploy-temp.sh  docker-hi-0.0
2 restart'''
          }
        }

        stage('local deploy') {
          steps {
            sh '''\\cp target/docker-hi-0.0.2.jar /home/admin/docker-hi/docker-hi-0.0.2.jar
\\cp deploy-temp.sh /home/admin/docker-hi/deploy-temp.sh
sh /home/admin/docker-hi/deploy-temp.sh docker-hi-0.0.2 restart'''
          }
        }

      }
    }

  }
}
```

Linux互信

严格来讲这篇文章不算是 Jenkins相关内容，但是我认为这个蛮关键的，所以这里记录下。

为了远程部署方便，除了通过Over SSH插件，还有一个办法就是通过将Jenkins所在主机的公钥写入目标机器的authorized_keys

比如 Jenkins所在的机器是192.168.0.161，目标机器是192.168.0.162。

- 在 Jenkins所在的机器执行：ssh-keygen -t rsa，默认情况下该指令会在~/.ssh下生成私钥公钥id_rsa.pub
- 在 Jenkins所在的机器执行：cat ~/.ssh/id_rsa.pub 然后复制内容``
- 在目标机器 192.168.0.162查看下~/.ssh是否有文件authorized_keys
- 如果没有那么创建一个，如果有的话直接将第二步复制的内容追加其中，记得要在新的一行粘贴容。

然后你可以尝试在161的主机尝试登陆162

```
# 此时用户是默认的本机 hostname
ssh 192.168.0.162
```

上述指令其实是最简答的写法,严格的写法是加上对方主机的 hostname。

```
ssh hostname@192.168.0.162
```

上述流程配置好后，就可以通过scp进行文件传输，比如这样

```
scp target/*.jar root@192.168.0.162:/home/admin/docker-hi/docker-hi-0.0.2.jar
```

也可以在远程主机执行本地的脚本

```
ssh root@192.168.0.162  'bash -s' < /home/admin/docker-hi/deploy-temp.sh  docker-hi-0.0.2 restart
```

当然也可以在远程主机执行远程的脚本

```
ssh root@192.168.0.162 /home/admin/docker-hi/deploy-temp.sh  docker-hi-0.0.2 restart
```

区别就是不带'bash -s'