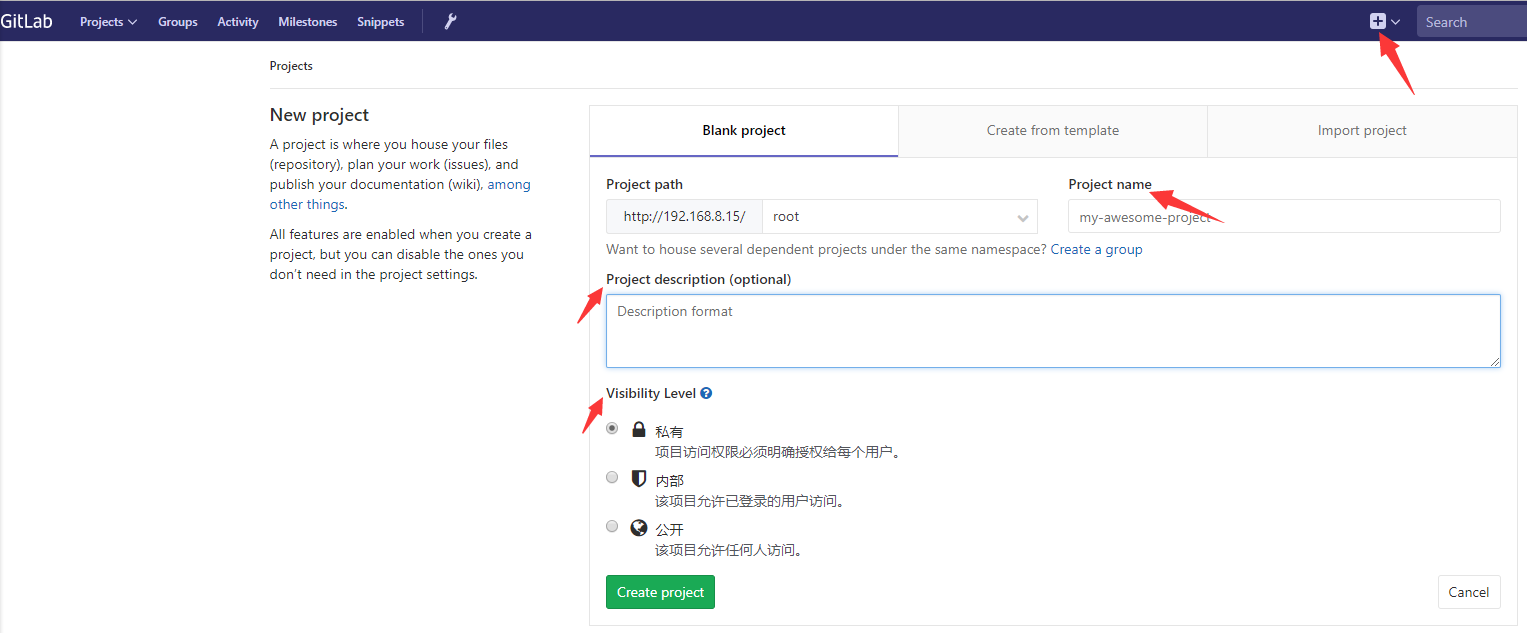
GitLab+Jenkins使用说明

1. GitLab使用（GitLab 是一个用于仓库管理系统的开源项目）
2. 新建项目
3. 网页新建

登录<http://192.168.8.15/users/sign_in> root wdzj@2017，如图：



1. 命令行本地新建并关联远端

**环境配置**：

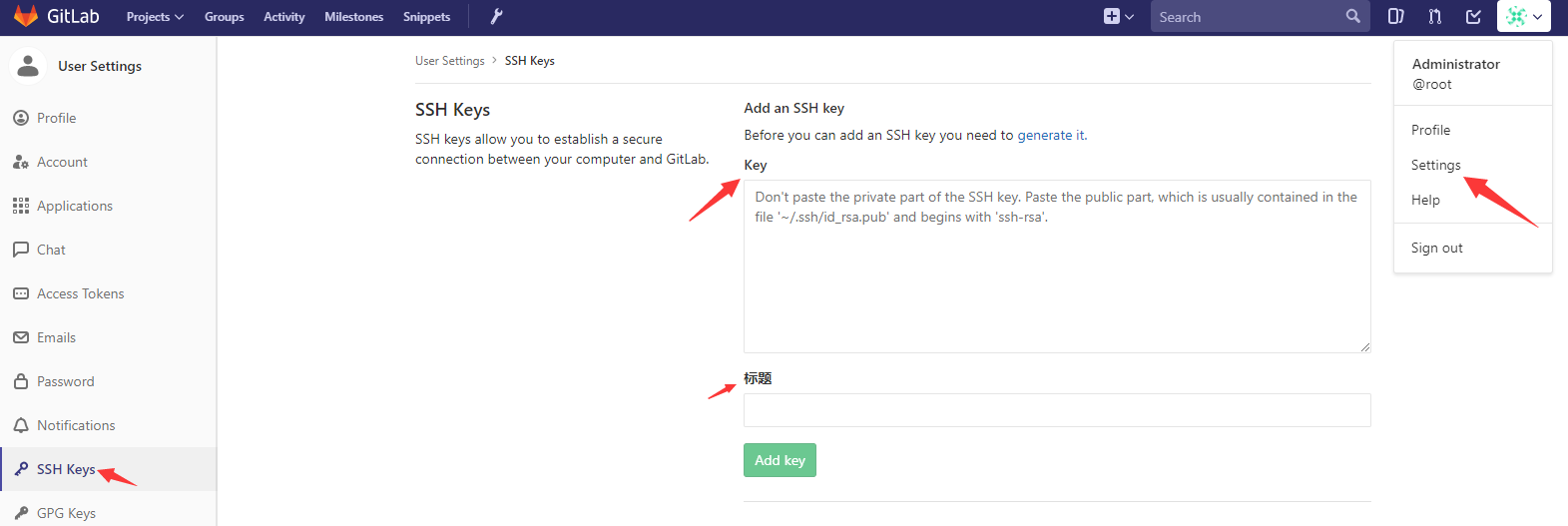
安装git: <https://git-scm.com/>

git config --global user.name "name"

git config --global user.email " xx@xx.com"

配置ssh密钥,Windows git-bash中: ssh-keygen -t rsa -C "xx@xx.com"

复制默认用户目录.ssh文件夹下id\_rsa\_pub内容粘贴至网站ssh key中，如图：



**本地新建：**

git init(以当前文件夹名新建项目)

git remote add origin <http://.....git>(关联，默认远端项目名origin，可自己改)

git pull(如果远端项目非空，需先拉取下来)

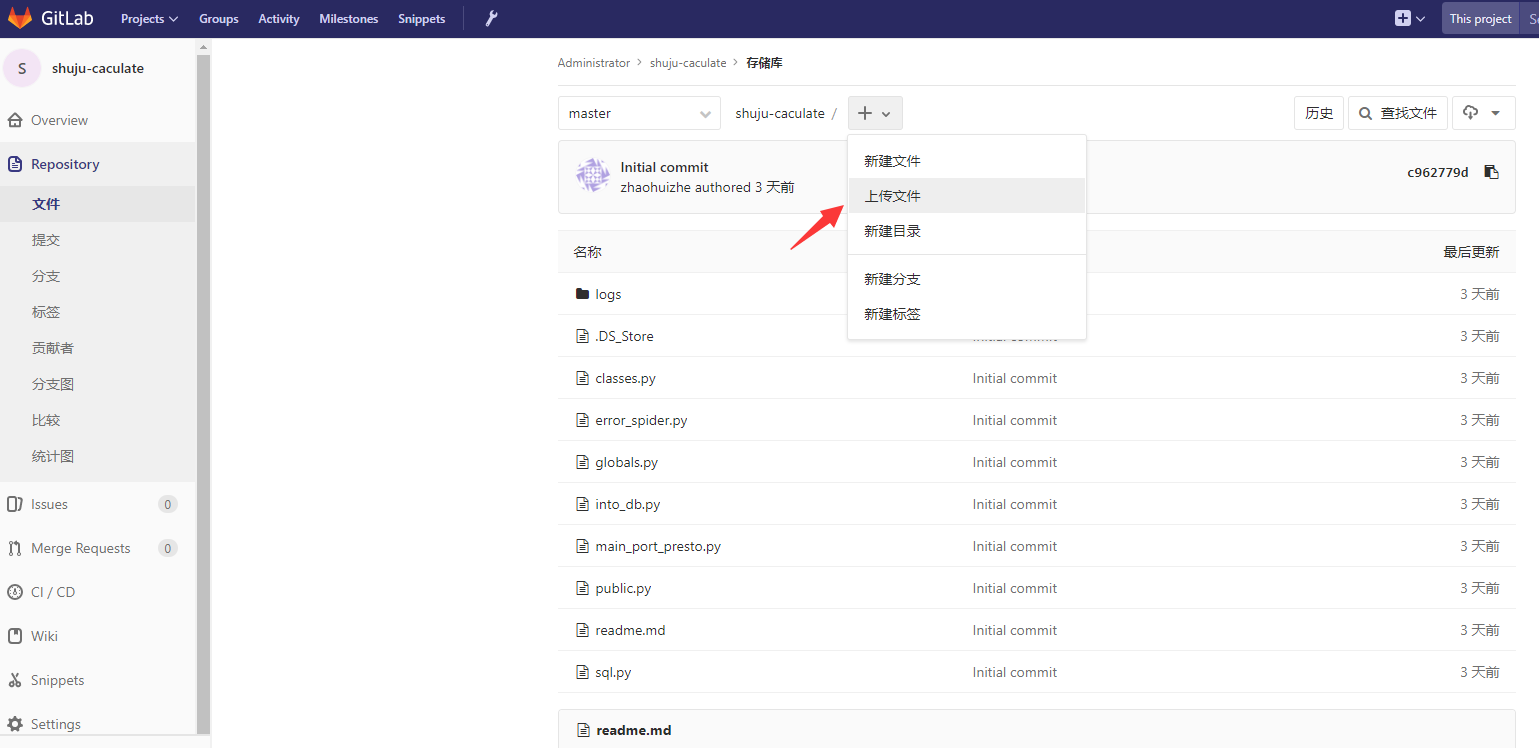
git push –u origin master(推送至远端origin项目中master分支，首次加-u指定默认主机)

**本地克隆远端：**

git clone http://...[[.git] myrepository]

2、代码上传

1. 网页上传



1. 命令行上传

git add –A（添加所有变化文件，或者添加指定文件）

git commit –m “”（提交至本地仓库，-m指定说明信息必须）

git push origin master（推送至远程仓库）

1. 分支管理

本地

git branch(查看本地分支)

git branch dev(创建dev分支，删除-d)

git checkout dev(切换至dev分支)

git checkout –b dev(创建并切换至dev分支)

远端

git branch -r (查看远端分支)

git push origin dev:dev(本地dev分支推送至远端[无则新建]dev分支，或网页新建)

git push origin :dev(删除远端dev分支,本地依就保存或者git push origin --delete dev)

推送

git push origin dev:master(推送本地dev分支至远端master分支)

git branch --set-upstream dev origin/dev (建立本地当前分支与远端dev分支链接)

git branch --unset-upstream master(取消对本地master分支的跟踪)

二、Jenkins使用（Jenkins是基于Java开发的一种持续集成工具，用于监控持续重复的工作，旨在提供一个开放易用的软件平台，使软件的持续集成变成可能，做到提交、编译、测试、发布）可以理解成可以自动执行任务（类似cron）并收集报告的基础工具。

* 1. 登录<http://192.168.8.15:8070> root wdzj@2017
  2. 创建任务

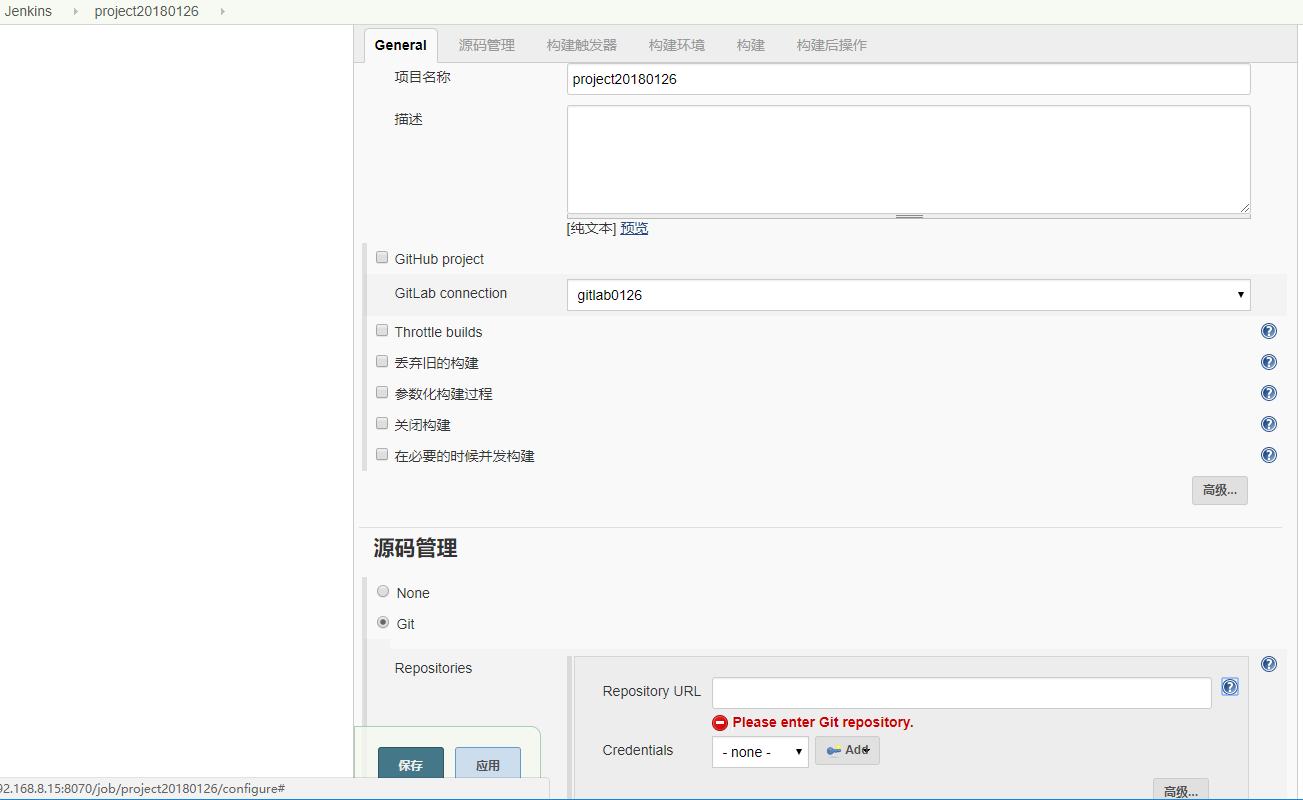
github project :配置响应的url和需要显示的名称

throttle builds：节流构建，通过设置时间段内允许并发的次数来实现构建的控制

参数化构建过程：配置不同的参数供构建时引用

安静期：设置每次构建的时间间隔

重试次数：这拉取源码重试的次数



**源码管理：**

Repository URL:（填写仓库的地址）

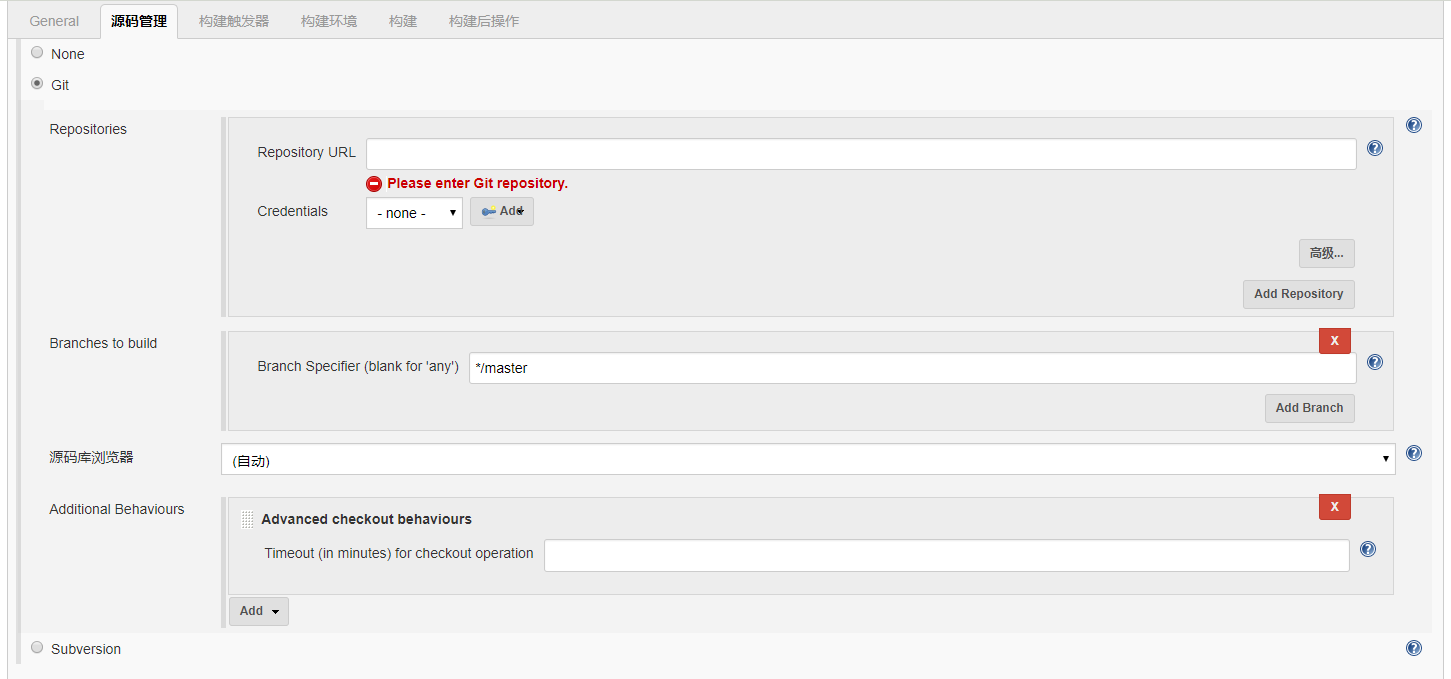
Credentials：（配置拉取源码证书，分SSH,用户名与密码认证等）

Local module directory：(具体的项目的路径，默认从根目录拉取）

Branches to build：（设置分支)

源码库浏览器：（添加跳转至版本控制工具连接）

Additional Behaviours：(其它操作，包括代码检查策略例如拉取超时、检查前后操作等）



**构建触发器：**

触发远程构建（使用脚、外部通过url命令触发）

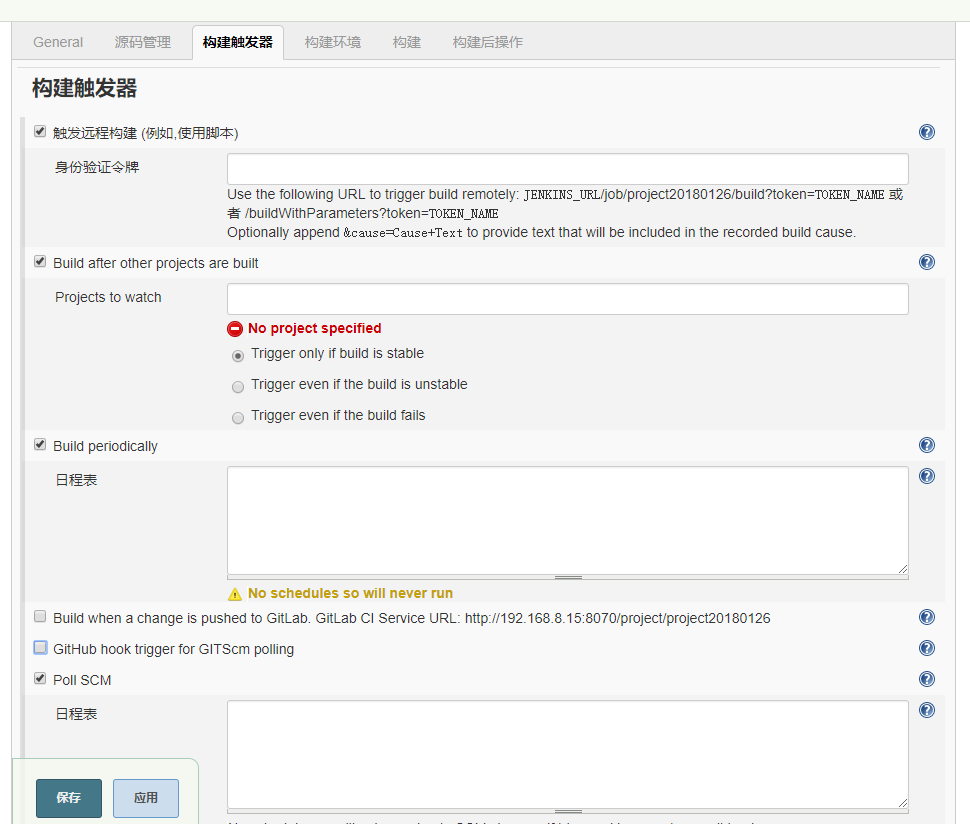
Build after other projects are built（在其他项目被构建后构建本项目）

Build periodically（定期构建）

Build when a change is pushed to GitLab（有推送时构建，Github回调Jenkins的接口）

GitHub hook trigger for GITScm polling（上一条改名后）

Poll SCM（检查代码库）



**构建环境：**

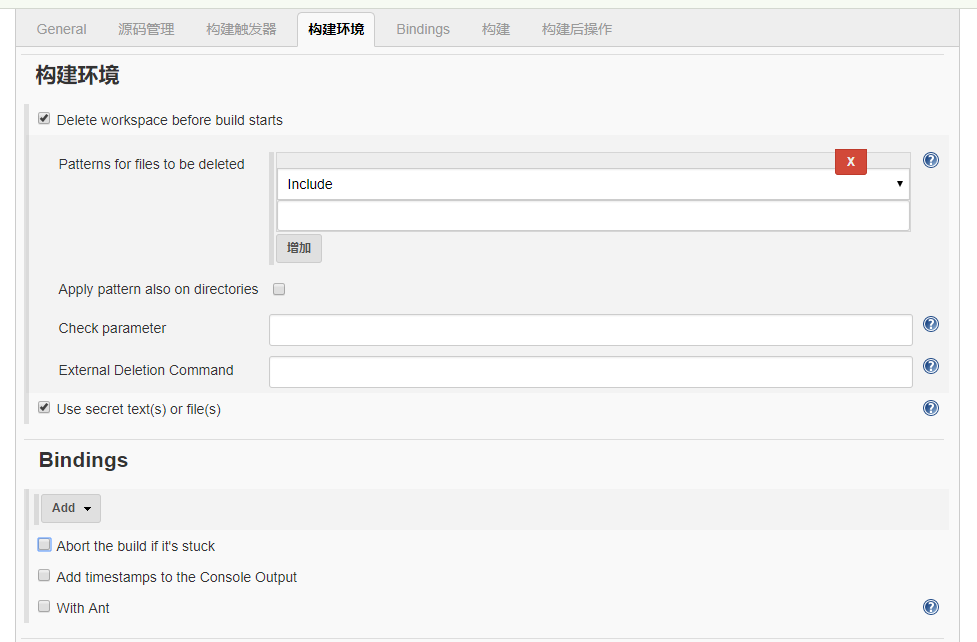
Delete workspace before build starts（构建前删除工作空间，可指定删除某些文件）

Use secret text(s) or file(s)（使用加密文本或文件）

Abort the build if it's stuck（卡住时终止）

Add timestamps to the Console Output（将时间戳输出至控制台）

With Ant（使用Ant构建）



**构建步骤：**

Execute Python script（添加Python脚本构建步骤）

Execute Windows batch command（添加Dos脚本构建步骤 ）

Execute shell（添加shell构建步骤 ）

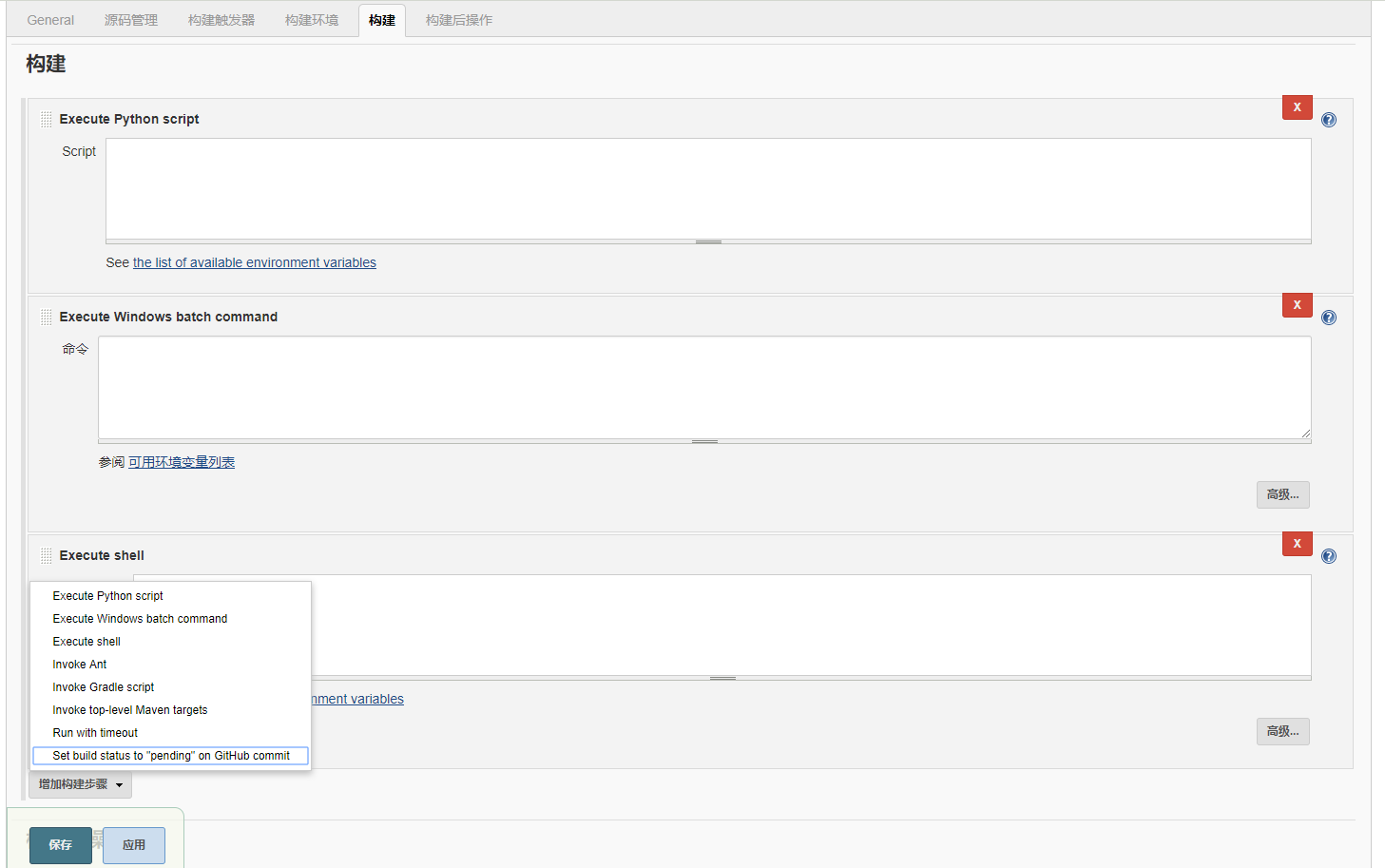
Invoke Ant（添加Ant构建步骤 ）

Inovke Gradle script（添加Ant脚本步骤）

Invoke top—level Maven targets（添加Marven构建步骤, maven命令）

Run with timeout（添加超时时间）

Set build status to “pending” on GitHub commit（设置构建状态提交至GitHub）



**构建后操作：**

Aggregate downstream test results（整合downstream测试结果）

Archive the artifacts（压缩存档）

Build other projects（构建其它项目）

Publish JUnit test result report（发布单元测试结果报告）

Record fingerprints of files to track usage（生成构件的指纹码）

Git Publisher（通过git发布）

Accept GitLab merge request on success（成功后接受GitLab的合并请求）

Add note with build status on GitLab merge requests（添加构建状态说明在GitLab合并请求）

Add vote for build status on GitLab merge request（为构建状态添加决定在GitLab合并请求）

E-mail Notification （增加邮件通知配置）

Editable Email Notification（增加可编辑邮件通知配置，设置主题、内容等等）

Publish build status to Gitlab commit(GitLab8.1+ required) （发布构建状态提交至GitLab）

Set GitHub commit status(universal) （设置GitHub提交状态）

Set build status on GitHub commit[deprecated] （设置构建状态提交至GitHub）

Delete workspace when build is done（构建完成后删除工作空间，可删除部分文件）

Publish Html report (HTML Publisher plugin插件)（增加查看Html结果配置）

