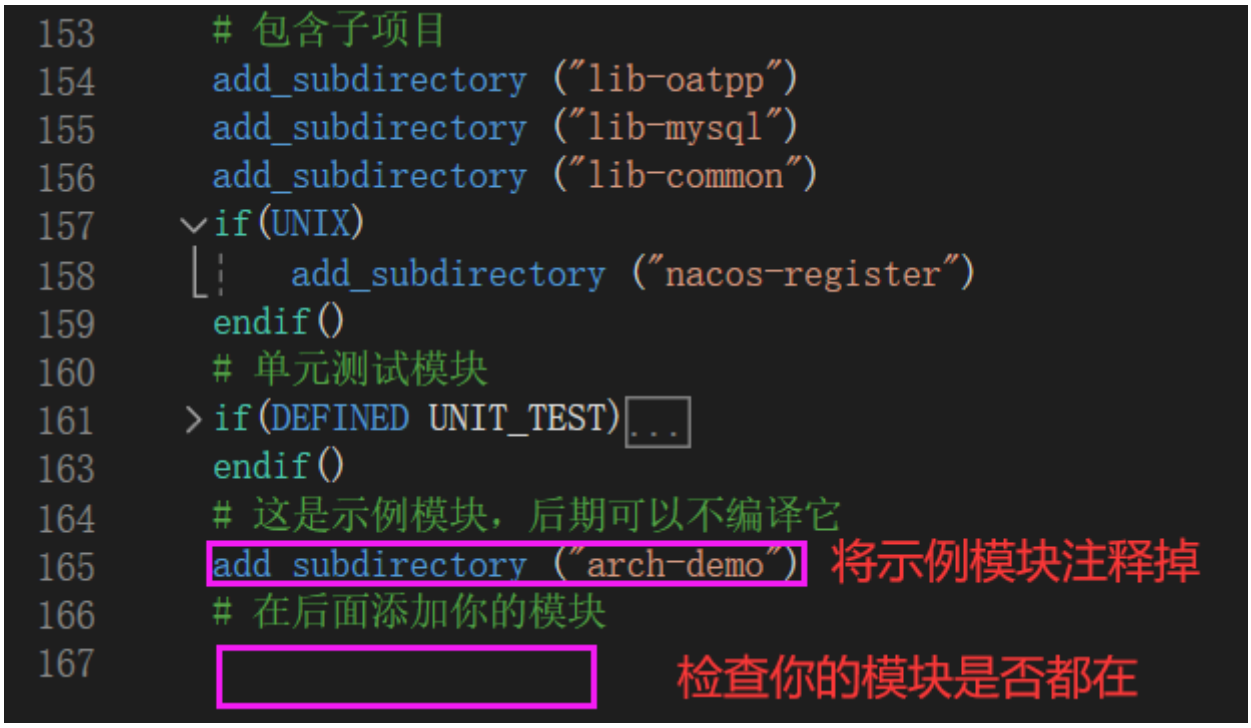
项目打包部署

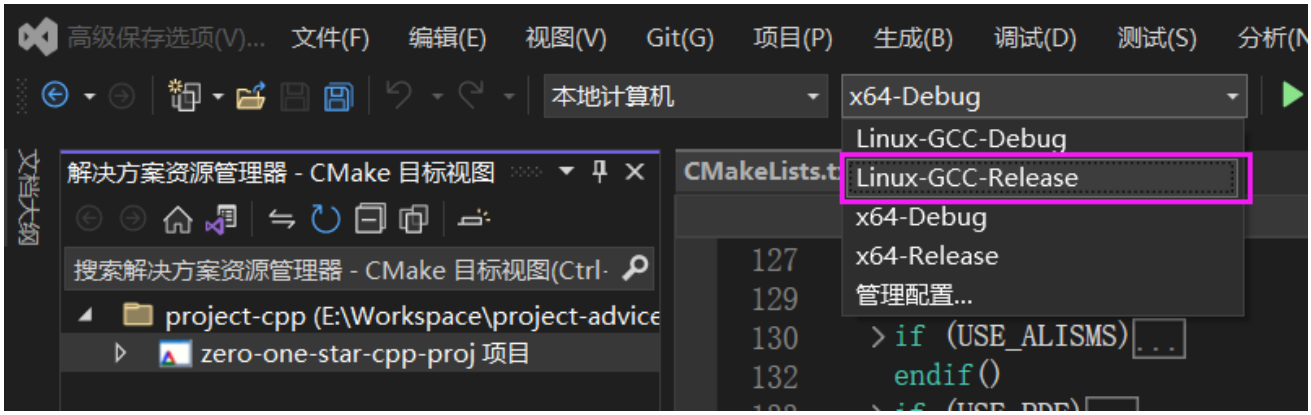
1 检查配置

打包之前需要检查解决方案目录中`CMakeLists.txt`配置中是否配置了所有需要打包的模块

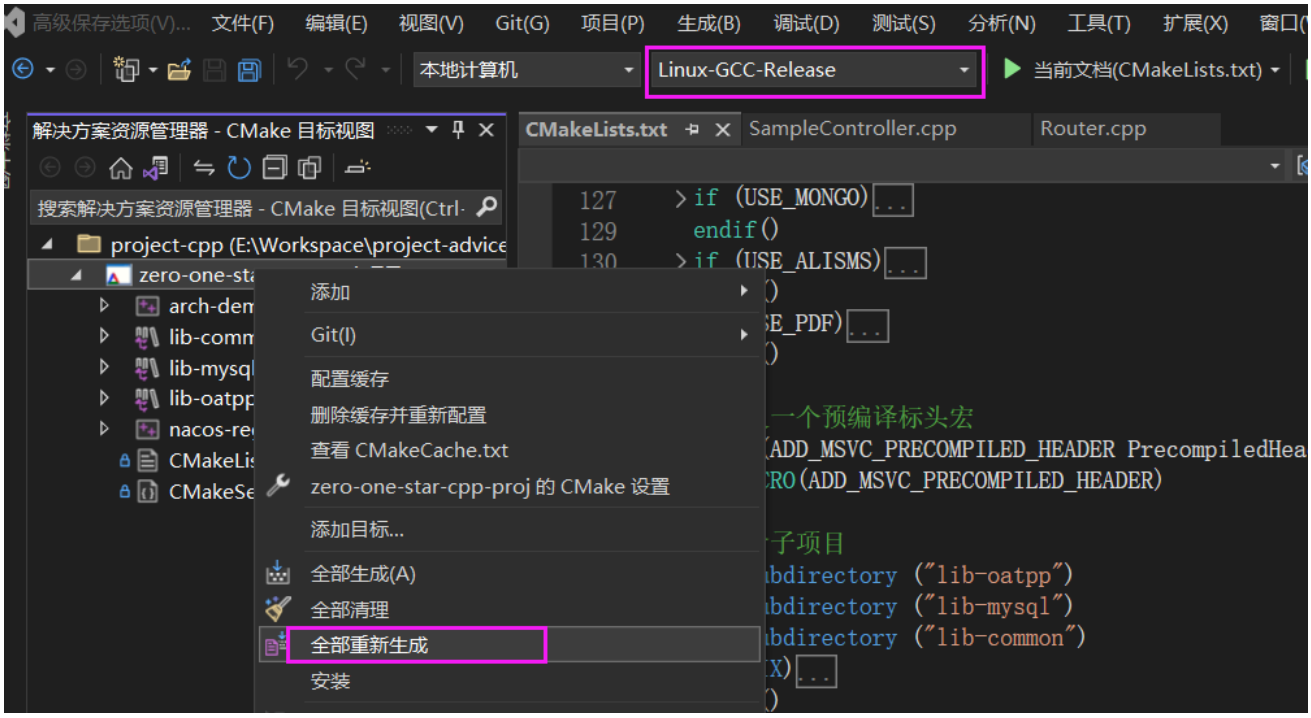


2 测试编译

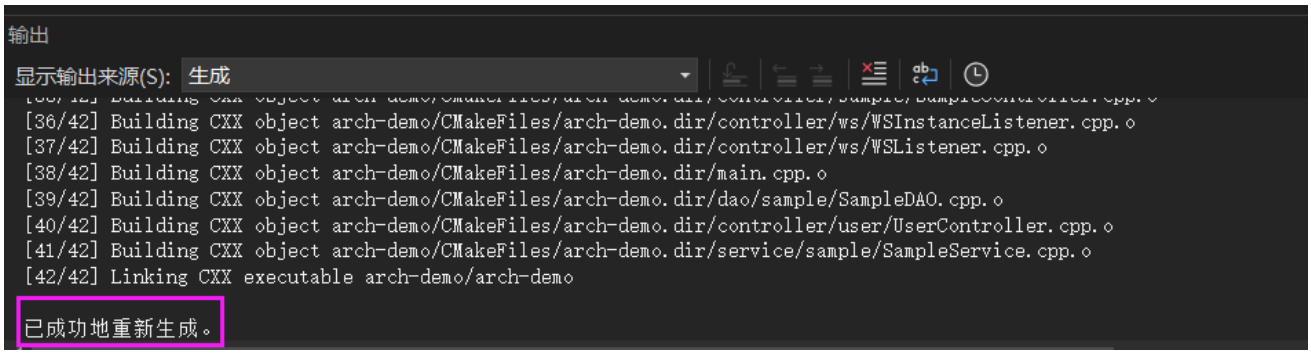
首先选择编译平台为Linux下面的Release平台



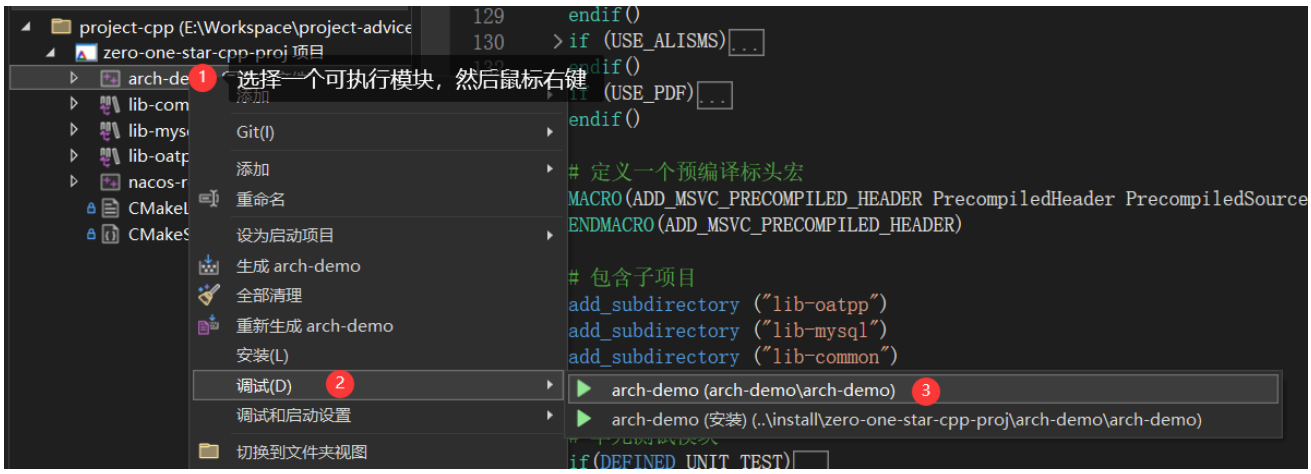
选中项目然后鼠标右键，执行**全部重新生成**指令对项目进行整体编译



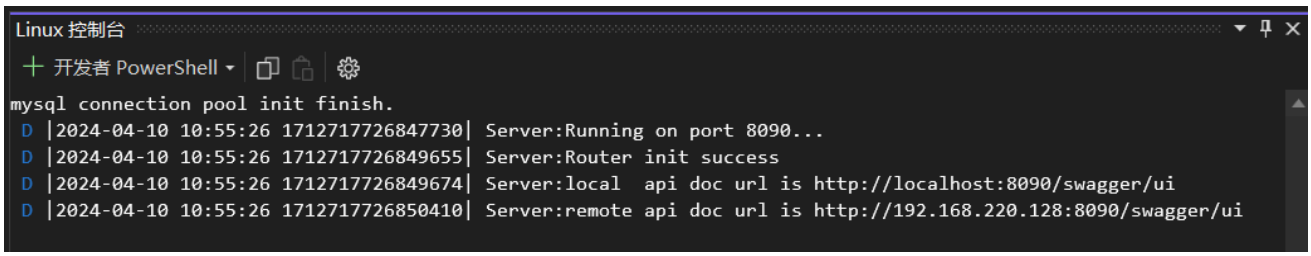
观察控制台查看编译结果



没有编译问题，测试启动一下，下面以arch-demo为例



查看启动结果



建议每个可执行模块都启动调试一下，保证没有问题后才开始打包

3 集成到Jenkins中

注意：在操作Jenkins的时候需要保证以下几点环境要求

Jenkins所在服务器宿主机需要安装以下软件依赖，通常你可以使用你的虚拟机来安装Jenkins，因为里面已经包括完整的编译执行环境。

```
# CentOS、龙蜥
yum -y install gcc-c++ cmake make
yum -y install openssl-devel libcurl-devel libuuid-devel
# Ubuntu
sudo apt-get -y install g++ cmake make
sudo apt-get -y install openssl libssl-dev libcurl4-openssl-dev uuid-dev
```

远程部署目标服务器需要安装以下软件依赖

```
# CentOS、龙蜥
yum -y install openssl-devel libcurl-devel libuuid-devel
# Ubuntu
sudo apt-get -y install openssl libssl-dev libcurl4-openssl-dev uuid-dev
```

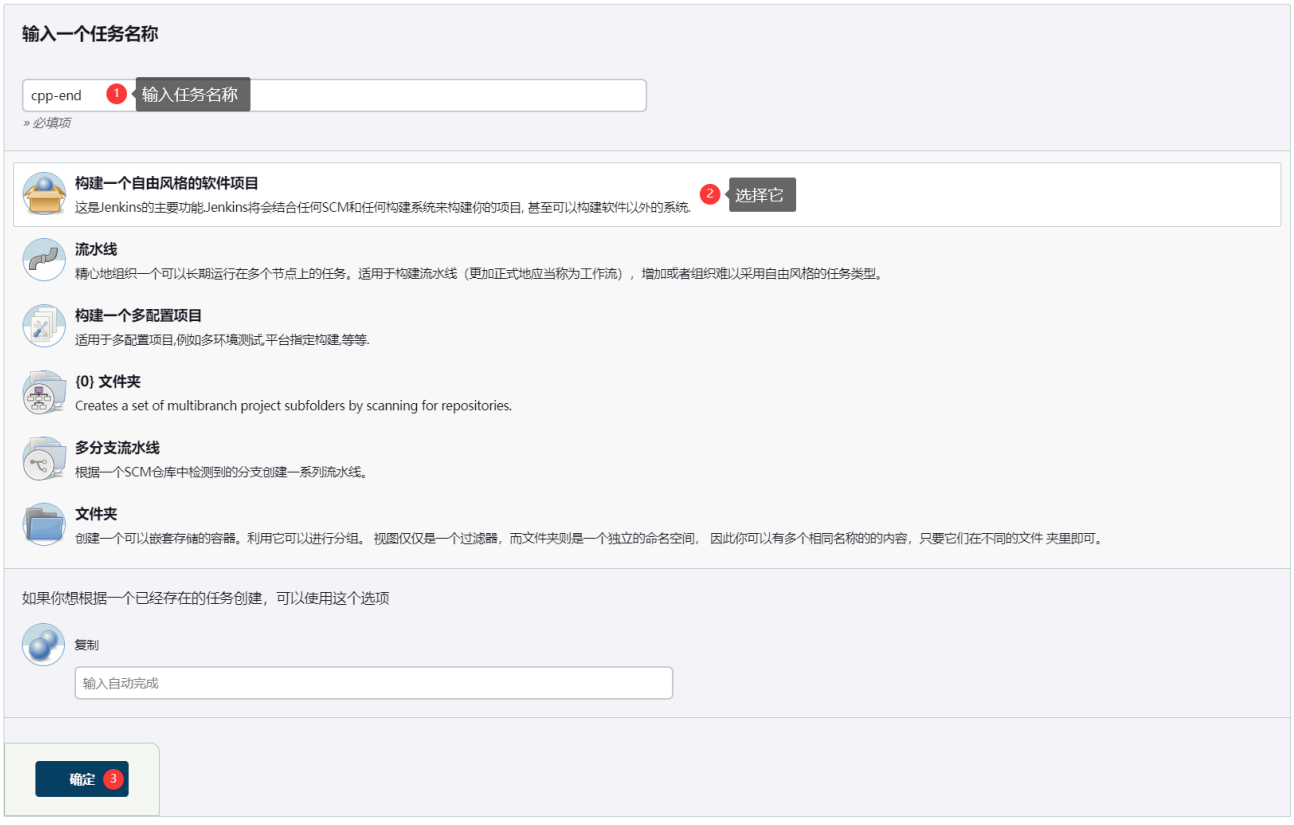
3.1 集成操作流程

3.1.1 创建任务

在主面板，选择创建任务


Jenkins
Dashboard
新建任务
1
用户列表
构建历史
项目关系
检查文件指纹

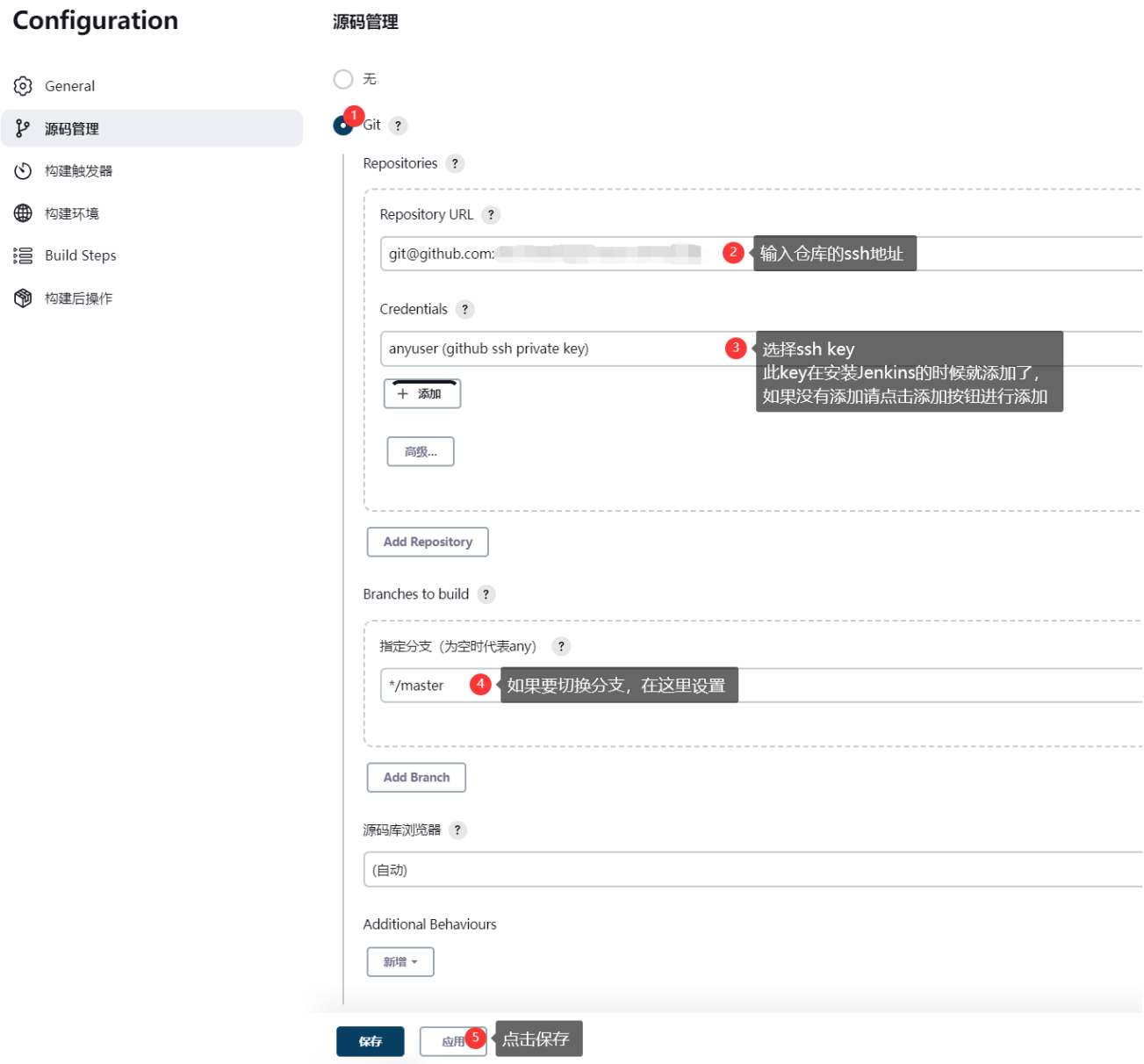
输入任务信息


输入一个任务名称
cpp-end
1
输入任务名称
» 必填项
构建一个自由风格的软件项目
这是Jenkins的主要功能.Jenkins将会结合任何SCM和任何构建系统来构建你的项目, 甚至可以构建软件以外的系统.
2
选择它
流水线
精心地组织一个可以长期运行在多个节点上的任务。适用于构建流水线（更加正式地应当称为工作流），增加或者组织难以采用自由风格的任务类型。
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.
(0) 文件夹
Creates a set of multibranch project subfolders by scanning for repositories.
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。
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， 因此你可以有多个相同名称的的内容，只要它们在不同的文件 夹里即可。
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
输入自动完成
确定
3

3.1.2 配置源码管理

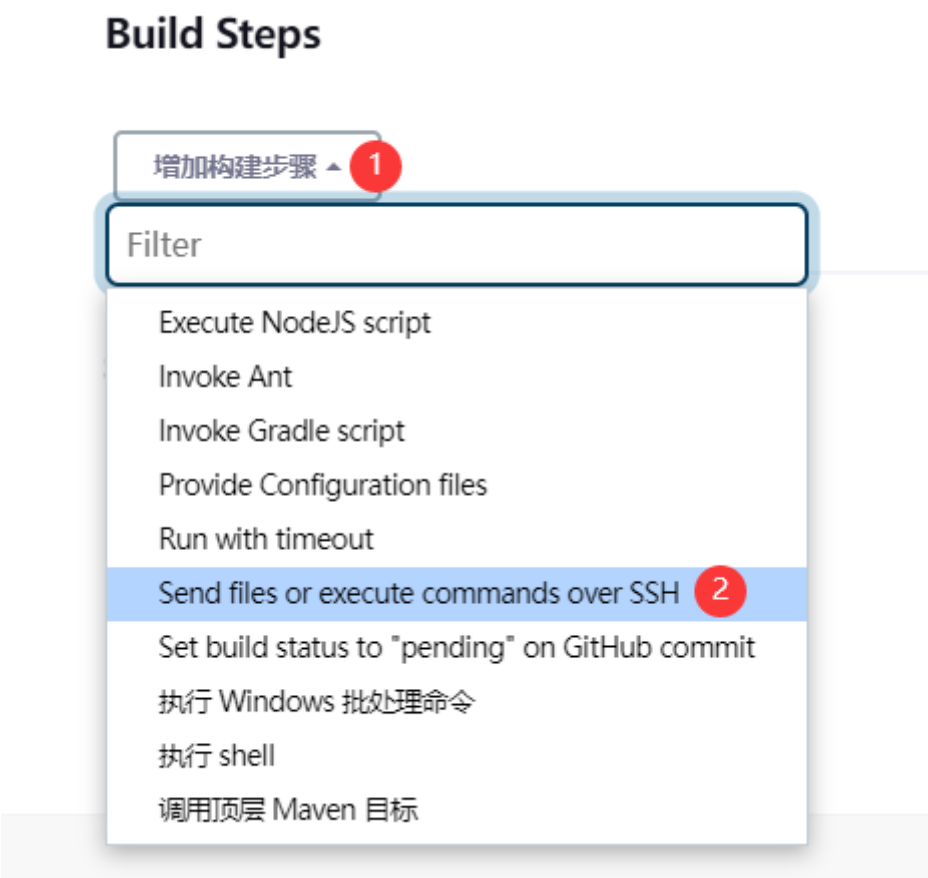
任务描述自己按照情况书写即可，在源码管理中添加你的仓库地址，在这里注意选择自己的分支。

提示：如果`GitHub`下载太缓慢，可以考虑将`GitHub`仓库导入到`Gitee`上面使用`Gitee`仓库来部署



3.1.3 编译源代码

由于是Docker安装的Jenkins，无法直接调用操作系统的相关指令，此时使用SSH方式连接到Jenkins所在服务器，调shell脚本进行用构建。


Build Steps
增加构建步骤
1
Filter
Execute NodeJS script
Invoke Ant
Invoke Gradle script
Provide Configuration files
Run with timeout
Send files or execute commands over SSH
2
Set build status to "pending" on GitHub commit
执行 Windows 批处理命令
执行 shell
调用顶层 Maven 目标

配置相关参数

Build Steps

Send files or execute commands over SSH ?

SSH Publishers

SSH Server
Name ?

192.168.220.128 1 选择Jenkins所在服务器

高级...

Transfers

Transfer Set
Source files ?

Remove prefix ?

Remote directory ?

Exec command ?

```
cd /home/jenkins/jenkins_home/workspace/cpp-end/
cd project-cpp
sh build.sh
```

2 输入执行命令

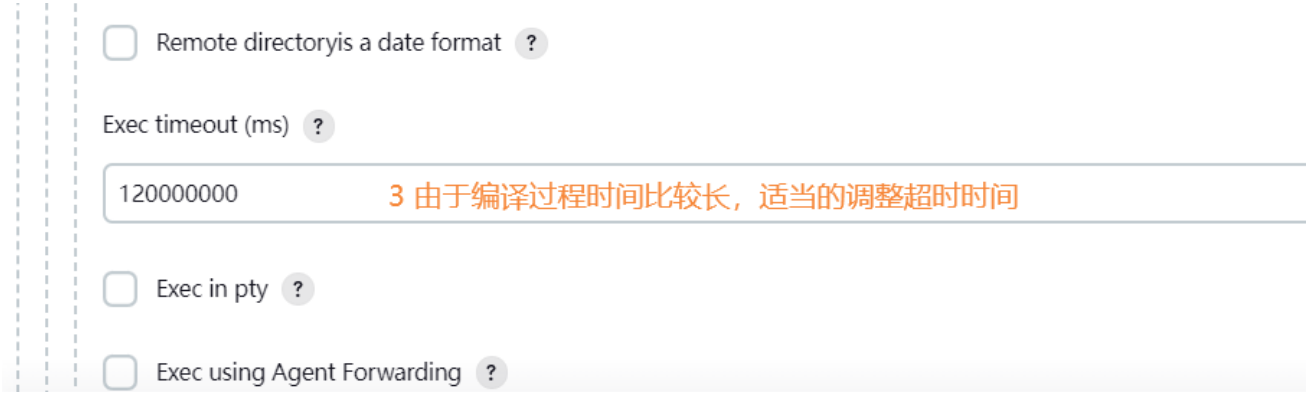
All of the transfer fields (except for Exec timeout) support substitution of **Jenkins environment variables**

Exclude files ?

Pattern separator ?

[,]+

- [ ] No default excludes ?
- [ ] Make empty dirs ?
- [ ] Flatten files ?
- [ ] Clean remote

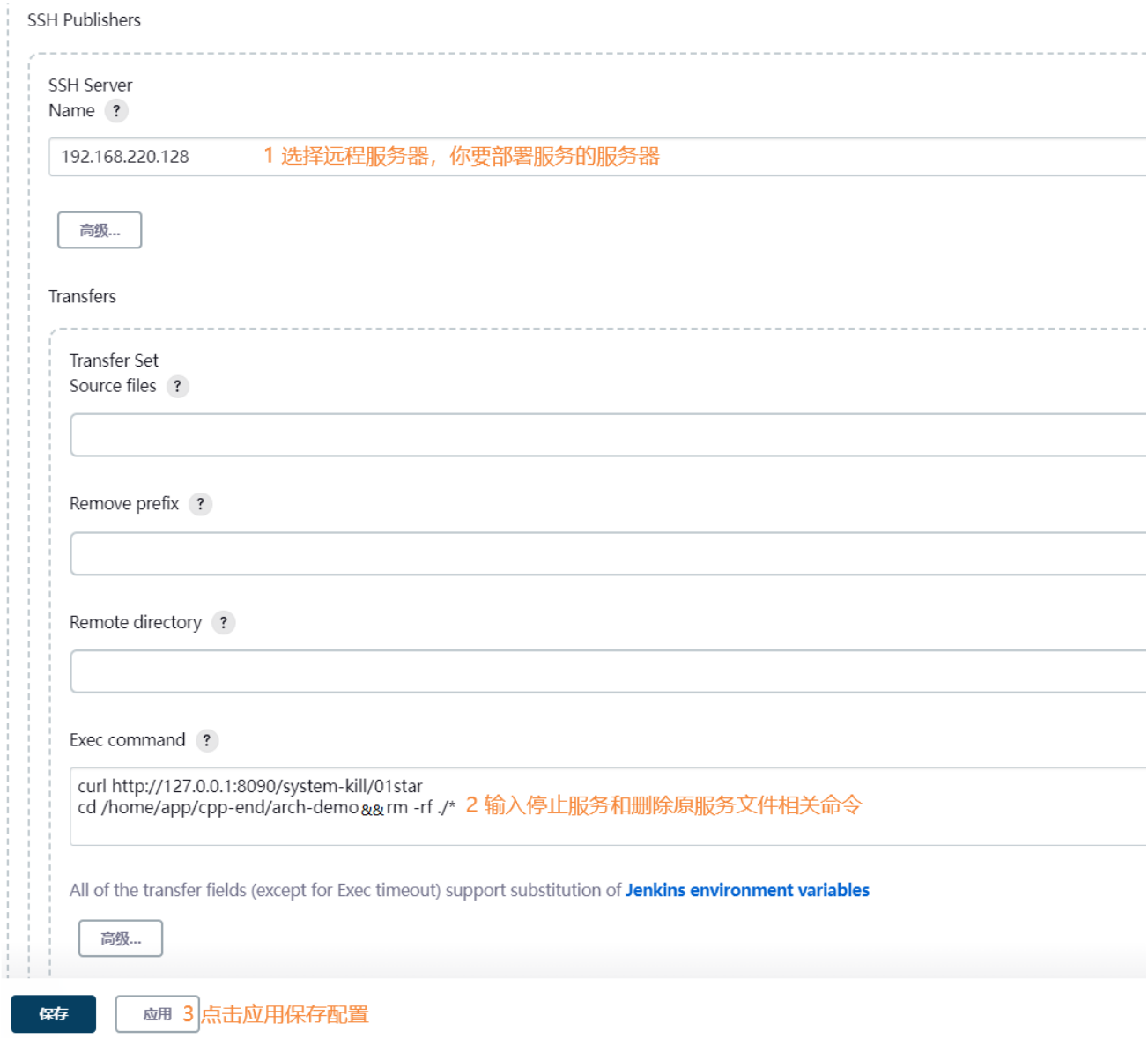


命令行解释

```
# 进入到clone资源目录路径格式： JenkinsHome + workspace + 任务名
cd /home/jenkins/jenkins_home/workspace/cpp-end/
# 进入项目源码目录
cd project-cpp
# 删除原来的编译文件
rm -rf out
# 执行编译
sh build.sh
```

3.3.4 停止远程服务

继续添加一个构建步骤，用于停止和清理远程服务器上已启动的服务，输入参数值如下

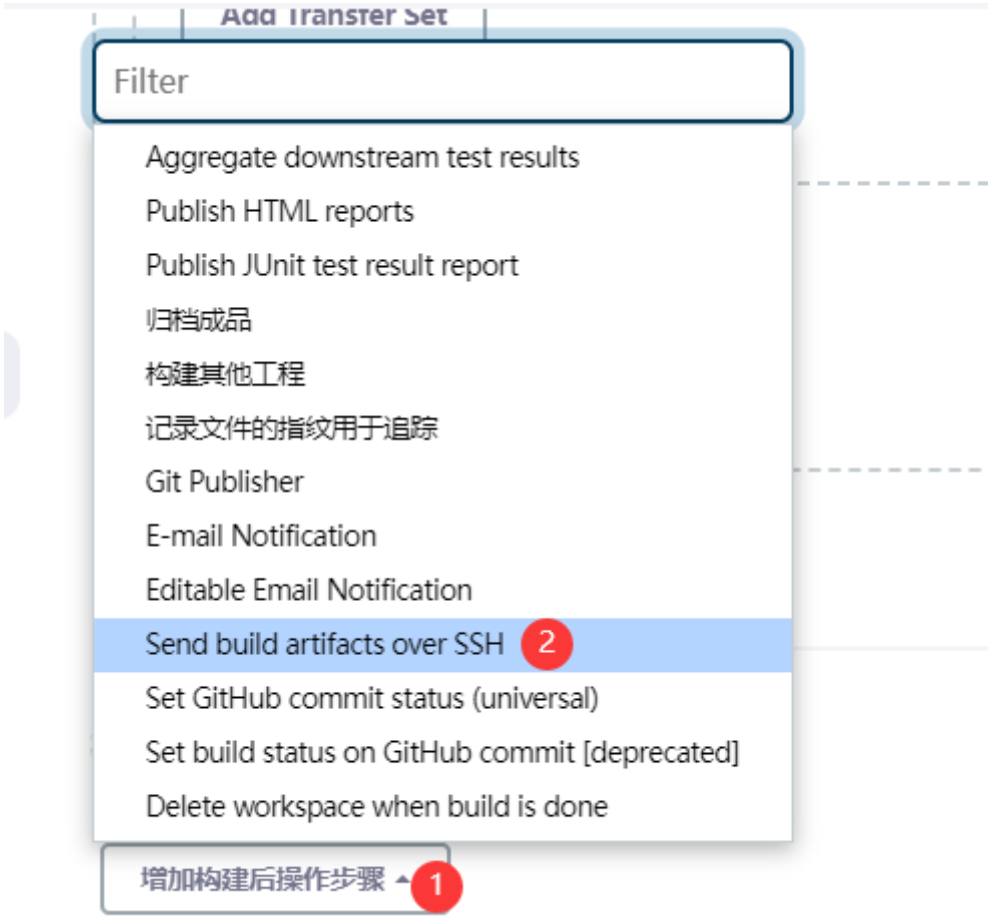


命令行解释

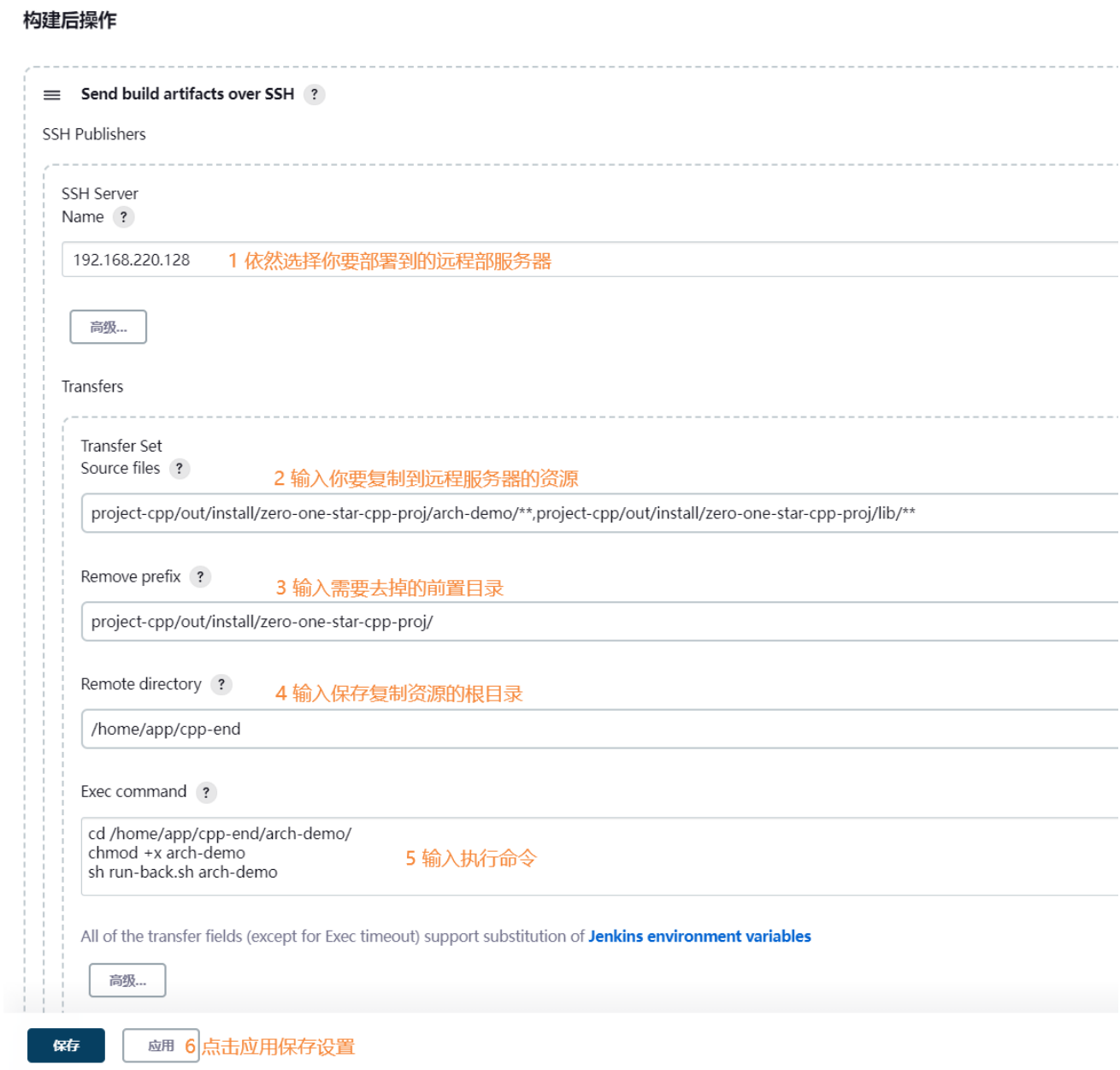
```
# 停止原来的服务
curl http://127.0.0.1:8090/system-kill/01star
# 进入到执行文件目录并删除目录下面的内容
cd /home/app/cpp-end/arch-demo && rm -rf ./*
```

3.3.5 配置远程启动

添加构建后操作步骤


Add Transfer Set
Filter
Aggregate downstream test results
Publish HTML reports
Publish JUnit test result report
归档成品
构建其他工程
记录文件的指纹用于追踪
Git Publisher
E-mail Notification
Editable Email Notification
Send build artifacts over SSH
2
Set GitHub commit status (universal)
Set build status on GitHub commit [deprecated]
Delete workspace when build is done
增加构建后操作步骤
1

然后设置参数如下：



复制资源描述

```
# 可执行文件目录
project-cpp/out/install/zero-one-star-cpp-proj/arch-demo/**,
# 依赖动态库目录
project-cpp/out/install/zero-one-star-cpp-proj/lib/**
```

执行命令描述

```
# 进入到执行文件目录
cd /home/app/cpp-end/arch-demo/
# 给可执行文件执行权限
chmod +x arch-demo
# 使用脚本启动服务，参数自己根据情况指定
sh run-back.sh arch-demo na=xxxx ns=xxxx ip=xxxx sn=xxxxx
```

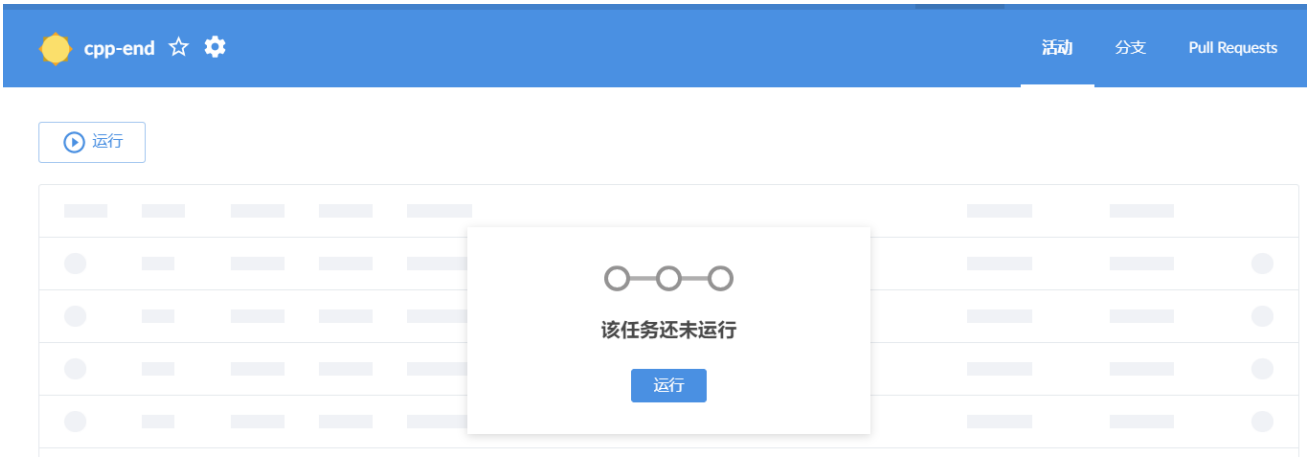
3.3.6 保存配置

所有操作执行完成后保存配置，为了不丢失操作，你可以每一步设定后都点击应用，防止丢失连接。

3.3.7 执行任务

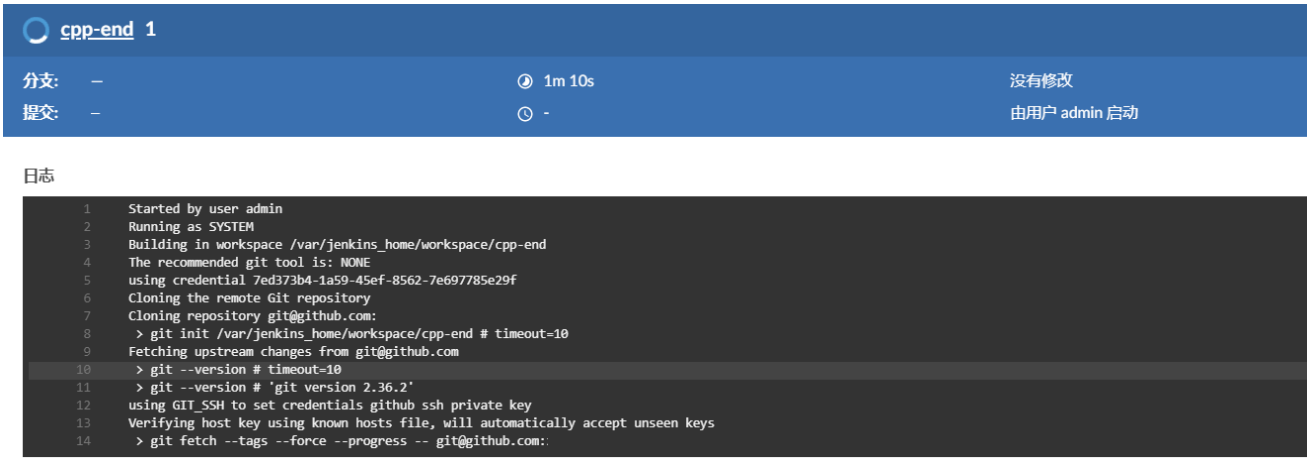
打开Blue Ocean

选择执行任务



3.3.8 查看日志

查看构建日志



任务执行完成