

Jenkins自动部署前后端

参考: <https://learnku.com/articles/44764>

1. 安装JDK

2. 安装nginx

3. 安装Jenkins

1. 在 Ubuntu 上安装 Jenkins 相对比较直接。我们将会启用 Jenkins APT 软件源, 导入源 GPGkey, 并且安装 Jenkins 软件包。使用下面的wget命令, 导入 Jenkins 软件源的 GPG keys: `wget -q -O - https://pkg.jenkins.io/debian/jenkins.io.key | sudo apt-key add -`
2. 添加软件源到系统中: `sudo sh -c 'echo deb http://pkg.jenkins.io/debian-stable binary/ > /etc/apt/sources.list.d/jenkins.list'`
3. 一旦 Jenkins 软件源被启用, 升级apt软件包列表, 并且安装最新版本的 Jenkins:
 - a. `sudo apt update`
 - b. `sudo apt install jenkins`
 - i. 注意Jenkins 2.357以上的版本会和jdk1.8产生冲突。参考: [Jenkins -- 踩坑记录之JDK8不兼容 - 学习的伊甸园](#)
 - ii. 查询Jenkins可安装版本: `apt-cache madison jenkins`
 - iii. 安装指定版本Jenkins: `apt-get install -y jenkins=x.xxx.x`
5. 在安装完成后, Jenkins 服务将会被自动启动。你可以通过打印服务状态来验证它: `systemctl status jenkins`。你应该看到类似下面的信息:

```
jenkins.service – Jenkins Continuous Integration Server
Loaded: loaded (/lib/systemd/system/jenkins.service; enabled; vendor prese>
Active: activating (start) since Mon 2023-01-02 15:00:19 CST; 1min 2s ago
.....
```

6. 输入服务器ip地址加上端口 8080, `http://your_ip_or_domain:8080` 。等待Jenkins开启后会展示该页面。



7. 使用 `sudo cat /var/lib/jenkins/secrets/initialAdminPassword` 命令获得本地的密码，粘贴到页面中。

8. 点击安装推荐的插件进行安装。



9. 按照页面提示创建一个管理员账户。



The screenshot shows the Jenkins '新手入门' (Getting Started) page. The main heading is '创建第一个管理员用户' (Create first administrator user). Below the heading are four input fields: '用户名' (Username), '密码' (Password), '确认密码' (Confirm Password), and '全名' (Full Name). At the bottom left, it says 'Jenkins 2.375.1'. At the bottom right, there are two buttons: '使用admin账户继续' (Continue with admin account) and '保存并完成' (Save and finish).

10. 设置 Jenkins 实例的 URL 地址。这个文本域将会被自动填写生成的 URL。



The screenshot shows the Jenkins '新手入门' (Getting Started) page. The main heading is '实例配置' (Instance Configuration). Below the heading is a text field labeled 'Jenkins URL:' with the value 'http://124.222.139.8:8080/'. Below the text field is a paragraph of text: '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。' At the bottom left, it says 'Jenkins 2.375.1'. At the bottom right, there are two buttons: '现在不要' (Don't now) and '保存并完成' (Save and finish).

11. 点击开始使用jenkins，重定向到jenkins工作台。安装jenkins完成

4. github 生成 Personal Access Token

1. github -> 头像 -> Settings -> Developer settings -> Personal access tokens -> Generate new token

- 勾选如下图所示，最后点击 generate token 生成令牌即可。

Select scopes

Scopes define the access for personal tokens. [Read more about OAuth scopes.](#)

<input checked="" type="checkbox"/> repo	Full control of private repositories
<input checked="" type="checkbox"/> repo:status	Access commit status
<input checked="" type="checkbox"/> repo_deployment	Access deployment status
<input checked="" type="checkbox"/> public_repo	Access public repositories
<input checked="" type="checkbox"/> repo:invite	Access repository invitations
<input checked="" type="checkbox"/> security_events	Read and write security events
<input type="checkbox"/> workflow	Update GitHub Action workflows
<input type="checkbox"/> write:packages	Upload packages to GitHub Package Registry
<input type="checkbox"/> read:packages	Download packages from GitHub Package Registry
<input type="checkbox"/> delete:packages	Delete packages from GitHub Package Registry
<input type="checkbox"/> admin:org	Full control of orgs and teams, read and write org projects
<input type="checkbox"/> write:org	Read and write org and team membership, read and write org projects
<input type="checkbox"/> read:org	Read org and team membership, read org projects
<input type="checkbox"/> manage_runners:org	Manage org runners and runner groups
<input type="checkbox"/> admin:public_key	Full control of user public keys
<input type="checkbox"/> write:public_key	Write user public keys
<input type="checkbox"/> read:public_key	Read user public keys
<input checked="" type="checkbox"/> admin:repo_hook	Full control of repository hooks
<input checked="" type="checkbox"/> write:repo_hook	Write repository hooks
<input checked="" type="checkbox"/> read:repo_hook	Read repository hooks

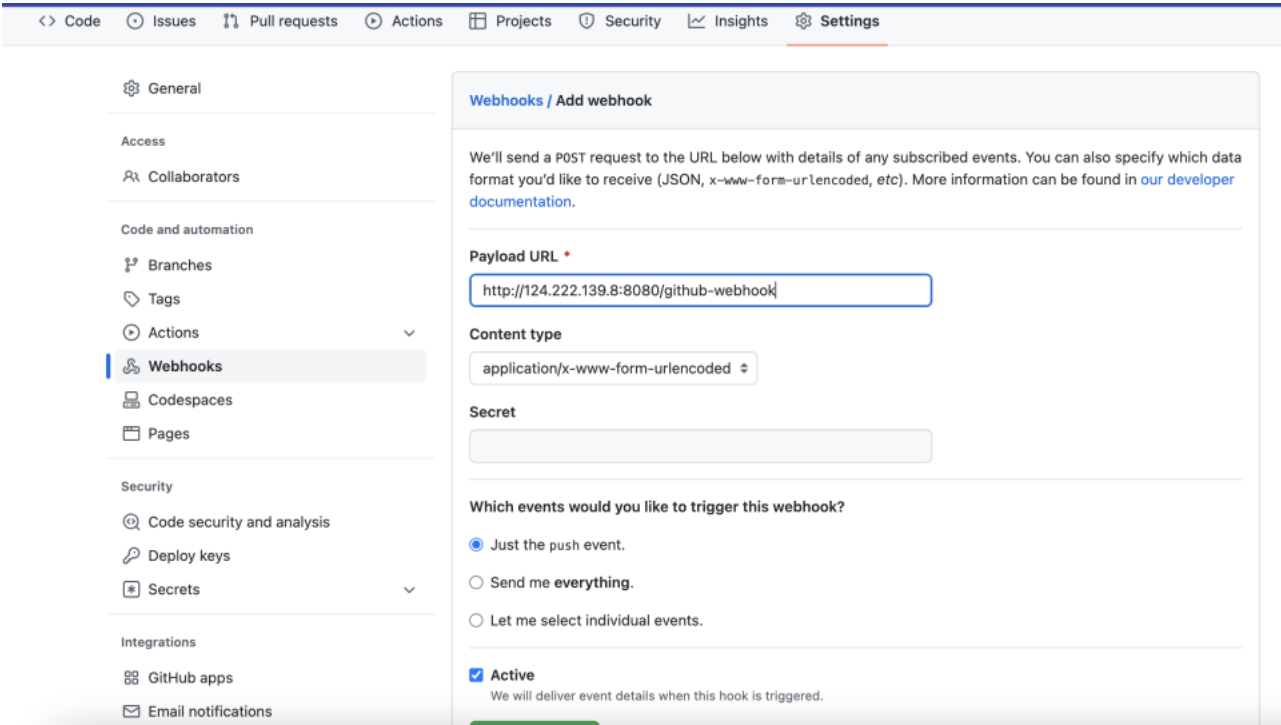
- 生成令牌之后要记录下来，因为只显示一次。

5. github 设置 GitHub webhooks

在具体需要持续集成的项目下进行操作

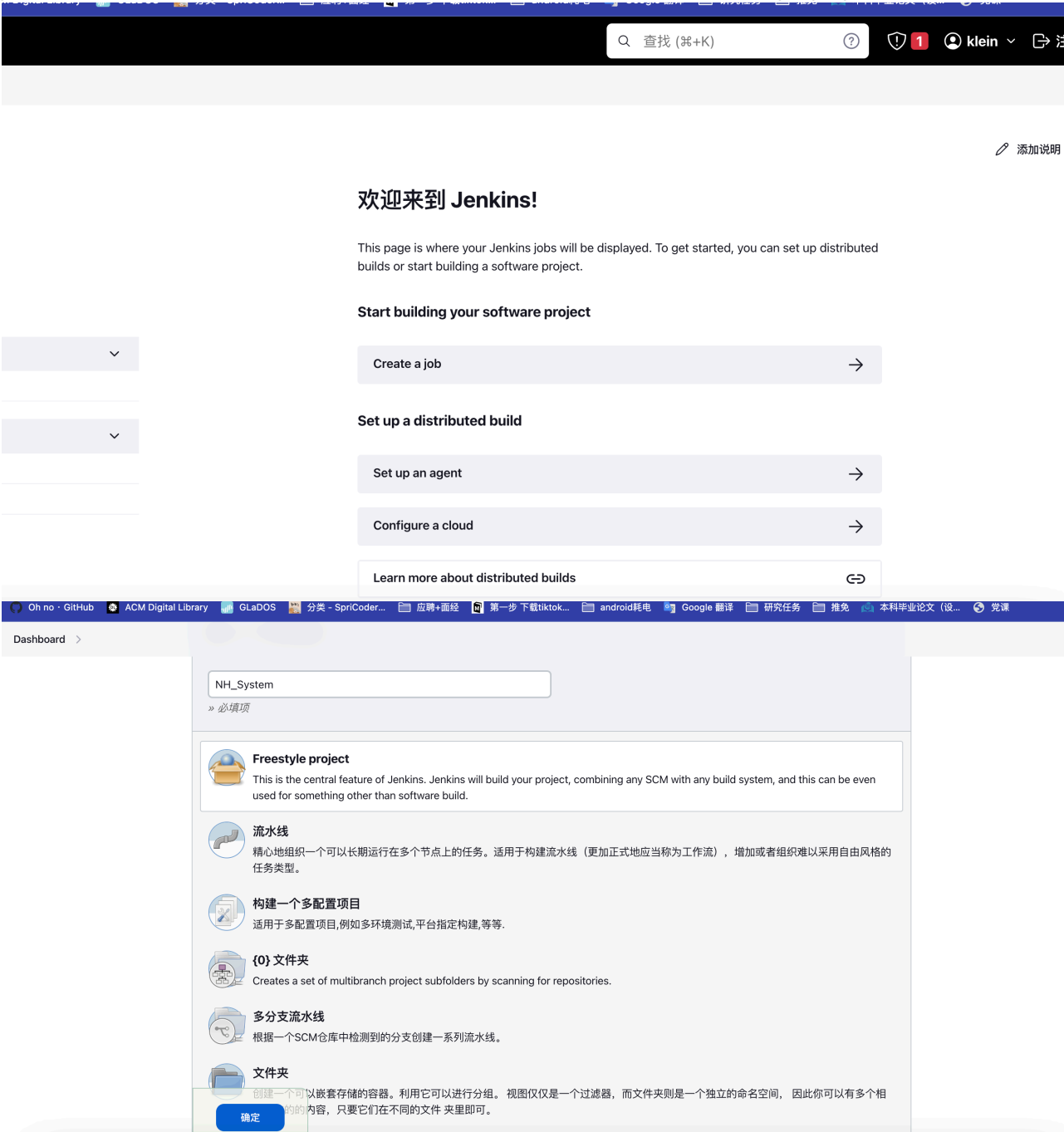
- 现有项目->settings->Webhooks
- 选项url: 部署的服务器的 IP + 端口 + github-webhook

3. 点击最下方的Add Webhook按钮



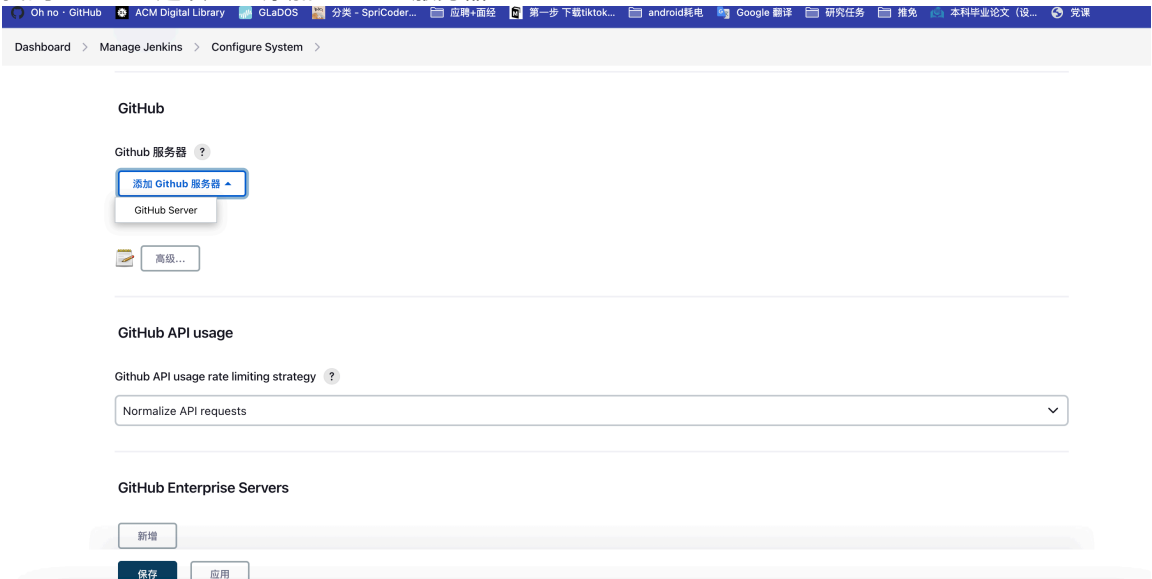
6. 设置 jenkins 的 github 配置

1. jenkins 点击create a job创建一个新任务，填写你的任务名称，并选择构建freestyle project

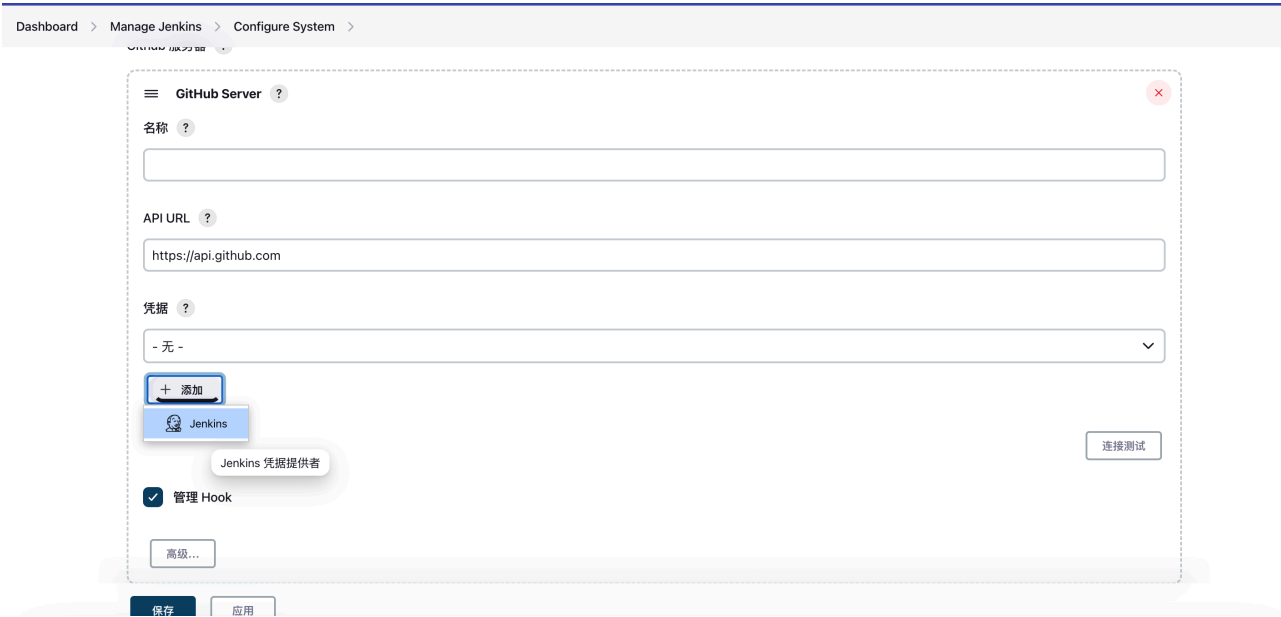


2. Manage Jenkins -> Configure System

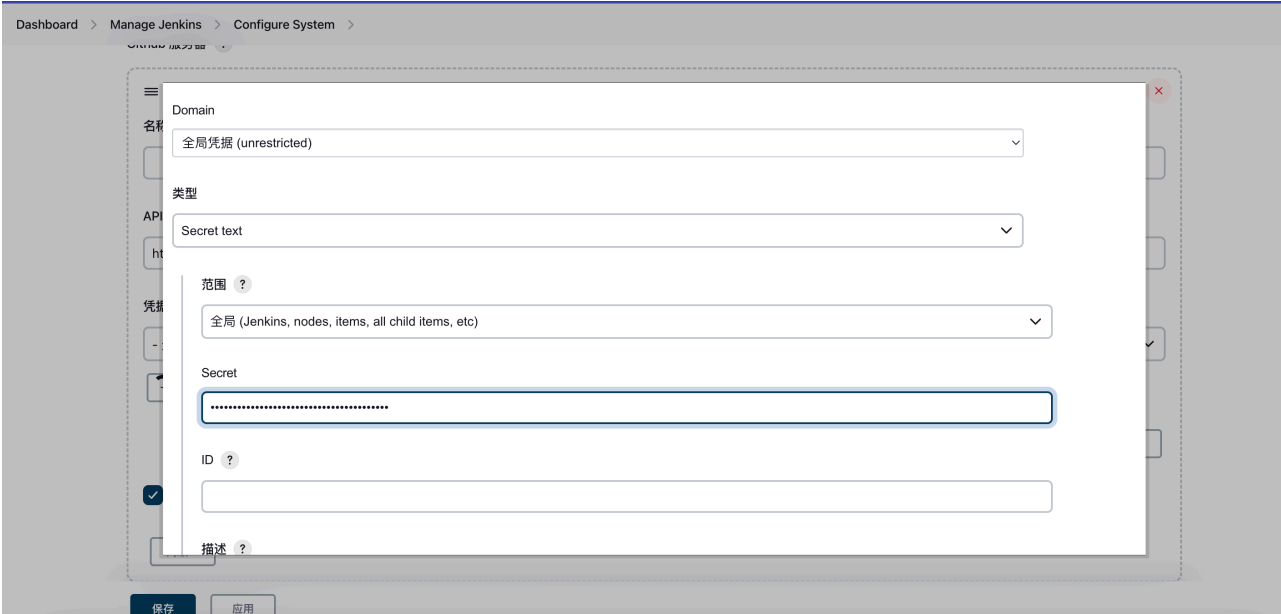
3. 找到 GitHub 选项 -> 添加 Github 服务器 -> GitHub Server



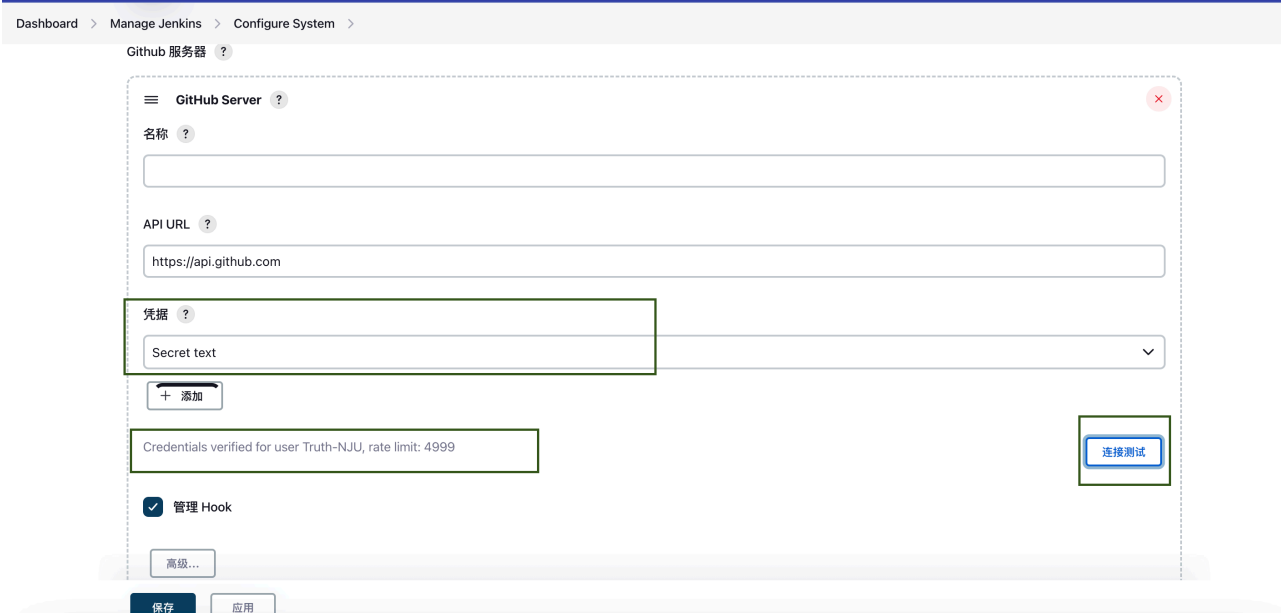
4. 勾选“管理 Hook”，添加 -> Jenkins



5. 在弹出的窗口中，如下图配置，这里需要用到之前生成的令牌

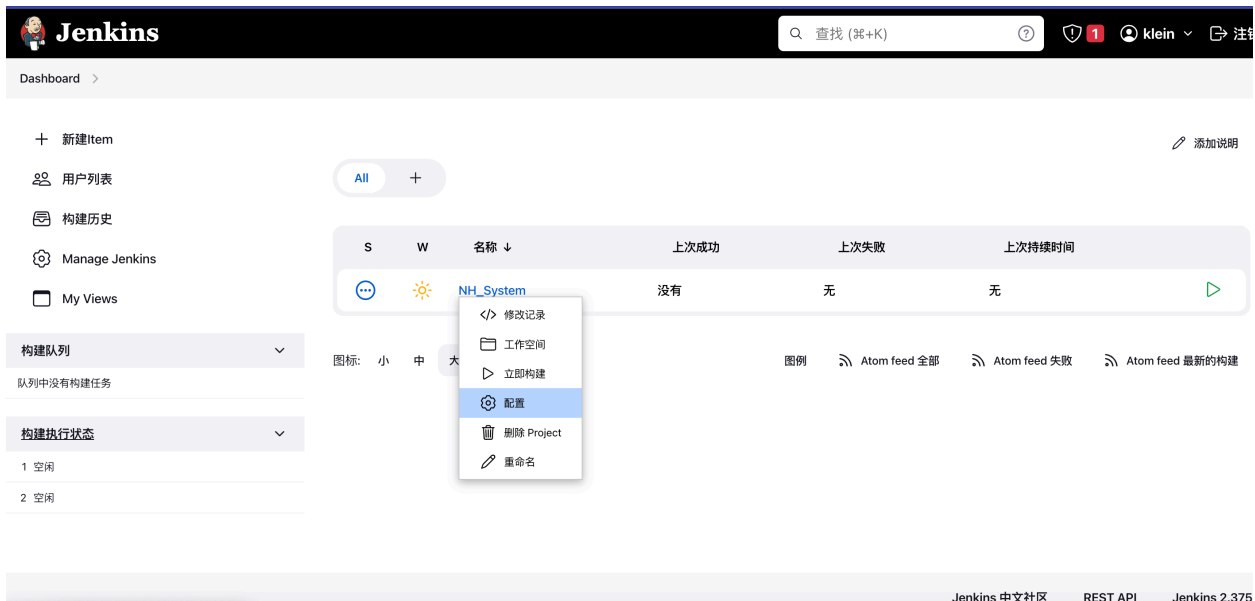


6. 选择生成的凭证，测试 jenkins 连接 github 服务器，如下图所示则配置成功，记得在页面底部保存配置。

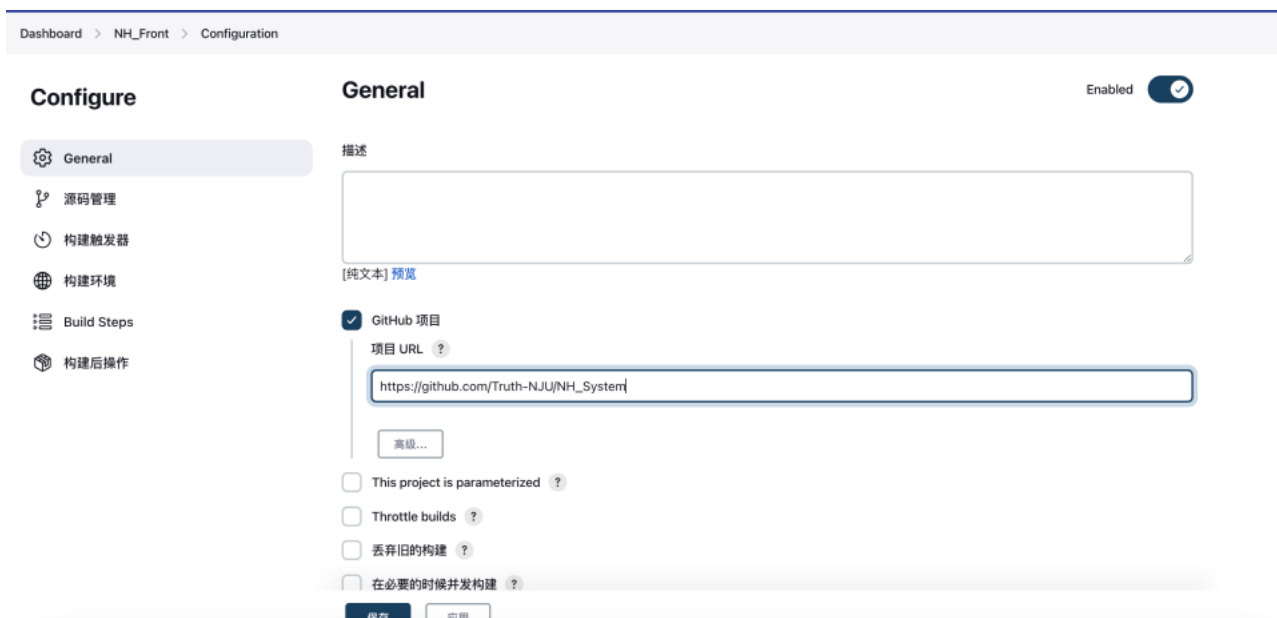


7. 设置 jenkins 的项目配置

7.1 进入该项目的配置项



7.2 选择 github 项目，并填入项目 URL（复制浏览器上的地址即可）



7.3 在源码管理下选择 git，并输入 Repository URL（克隆下载你项目的地址）

Configure

General

源码管理

构建触发器

构建环境

Build Steps

构建后操作

☐ 无☒ Git ?

Repositories ?

Repository URL ?

git@github.com:Truth-NJU/NH_System.git

❗ 无法连接仓库: Command "git ls-remote -h -- git@github.com:Truth-NJU/NH_System.git HEAD" returned status code 128:
stdout:
stderr: Host key verification failed.
fatal: Could not read from remote repository.

Please make sure you have the correct access rights
and the repository exists.

Credentials ?

- 无 -

+ 添加

保存

应用

1. 若没有安装git

- a. `sudo apt update`
- b. `sudo apt install git`
- c. `git --version`

2. 安装完git后, 仍然会报上面的错误, 在服务器上切换到jenkins用户 `su jenkins`, 执行报错中类似于 `git ls-remote -h git@bitbucket.org:person/projectmarket.git HEAD` 形式的命令。首次通过 SSH 连接到新主机时, 您将收到标准 SSH 警告: `The authenticity of host 'bitbucket.org (207.223.240.181)' can't be established. RSA key fingerprint is 97:8c:1b:f2:6f:14:6b:5c:3b:ec:aa:46:46:74:7c:40. Are you sure you want to continue connecting (yes/no)?` 键入yes并按 Enter。主机密钥github.com将会添加到`~/.ssh/known_hosts`文件中。

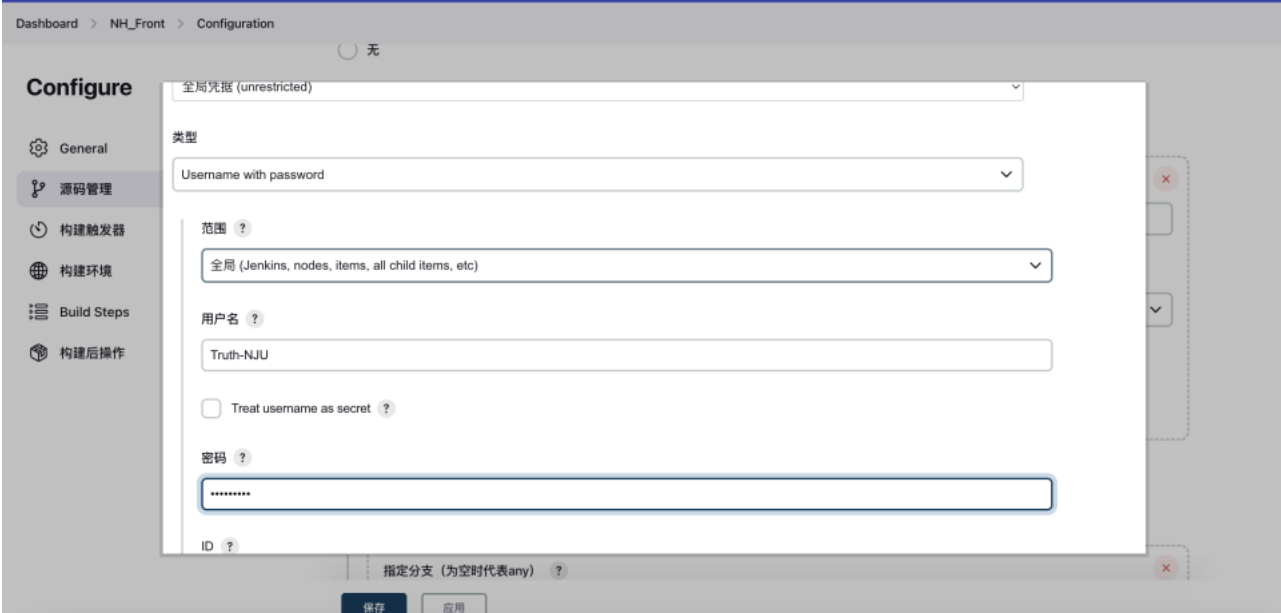
3. 这时手动在服务器的命令行执行 `git ls-remote -h git@bitbucket.org:person/projectmarket.git HEAD`, 还会出现如下报错:

```
git@github.com: Permission denied (publickey).  
fatal: Could not read from remote repository.
```

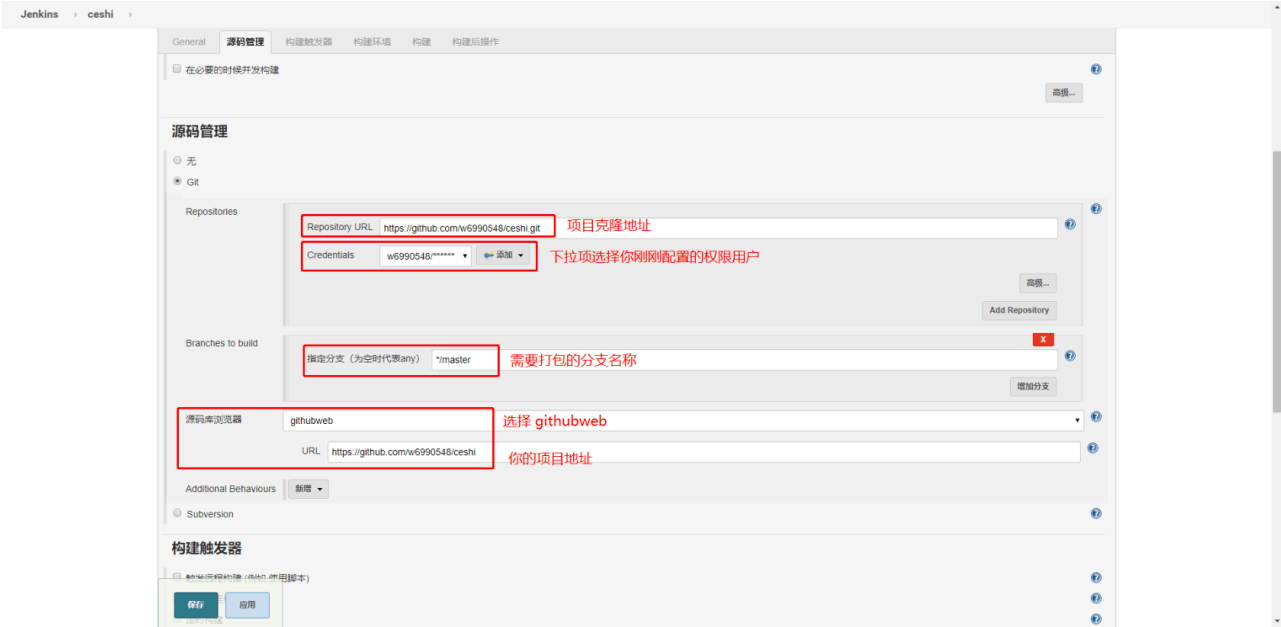
Please make sure you have the correct access rights
and the repository exists.

4. 输入 `ssh-keygen -t rsa -C "email of github"` 命令生成id_rsa和id_rsa.pub文件, 将id_rsa.pub中的内容复制, 进入github账号, 在settings下, SSH and GPG keys下new SSH key, title随便取一个名字, 然后将id_rsa.pub里的内容复制到Key中, 完成后Add SSH Key。(也就是配置一个免密clone)
5. 刷新页面重新填入url, 报错消失。(注意2、3、4三步都是在jenkins用户下进行的)

6. 选择 Credentials，若下拉选项中有，则直接选择即可。若没有，点击添加 -> jenkins，添加一个 Username with password 类型的权限用户，直接用 github 的登陆名称和密码创建。

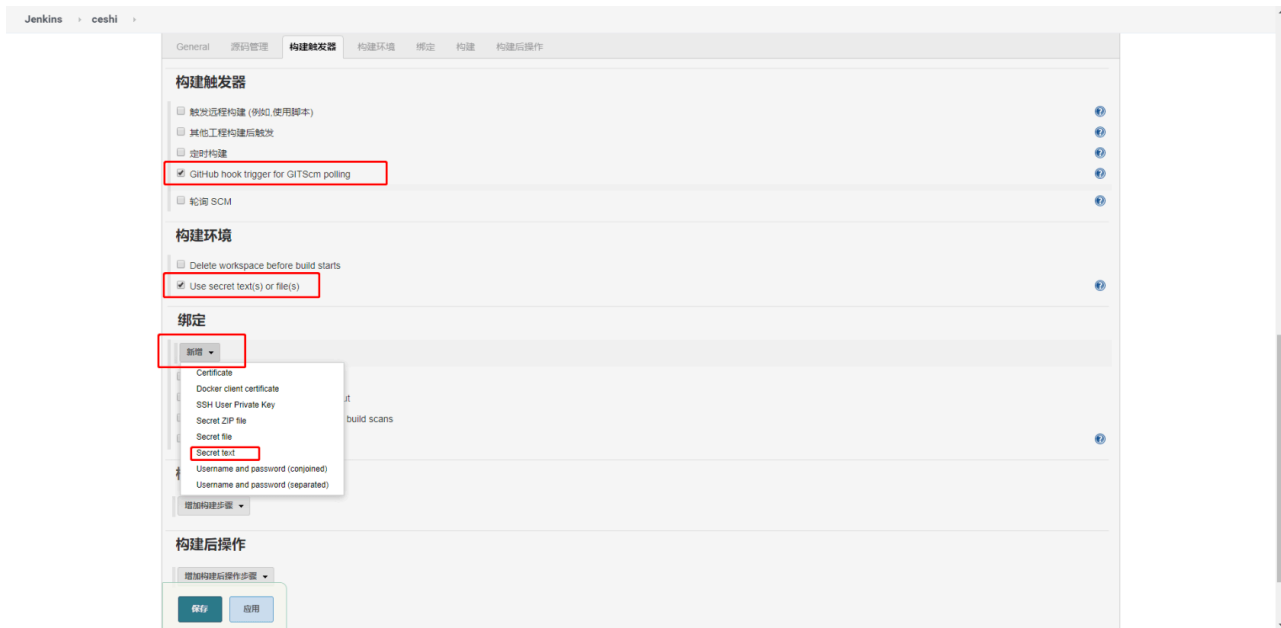


7. 最终源码管理的配置类似于下图

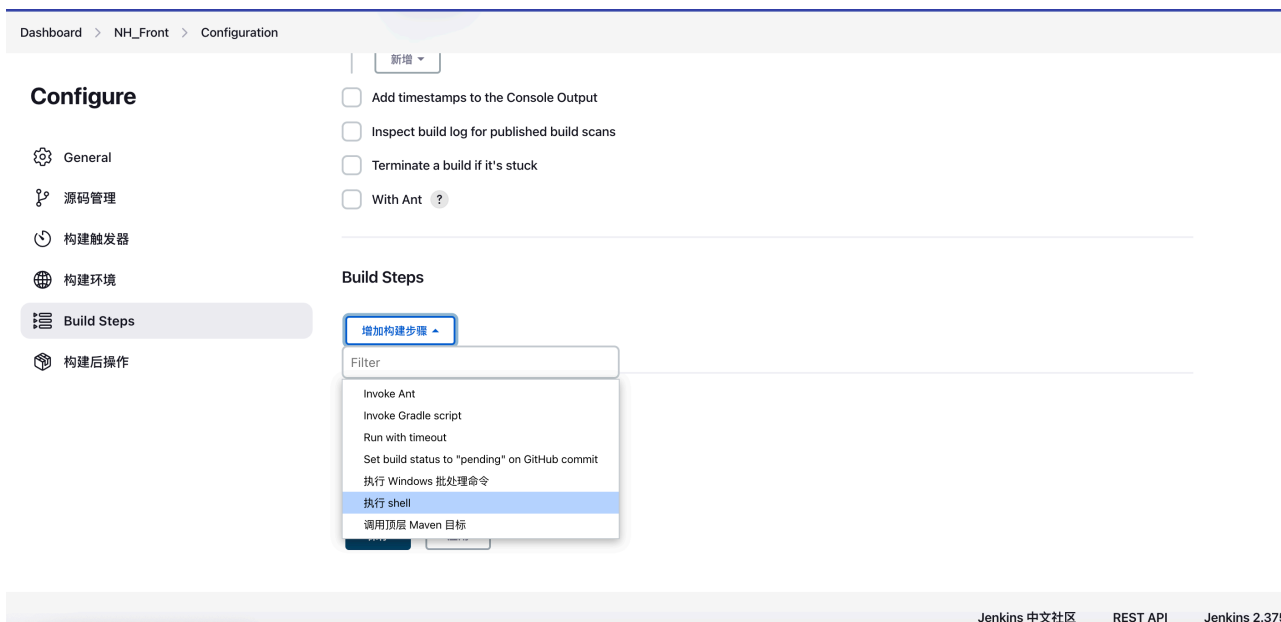


7.4 构建触发器 + 构建环境 + 绑定配置

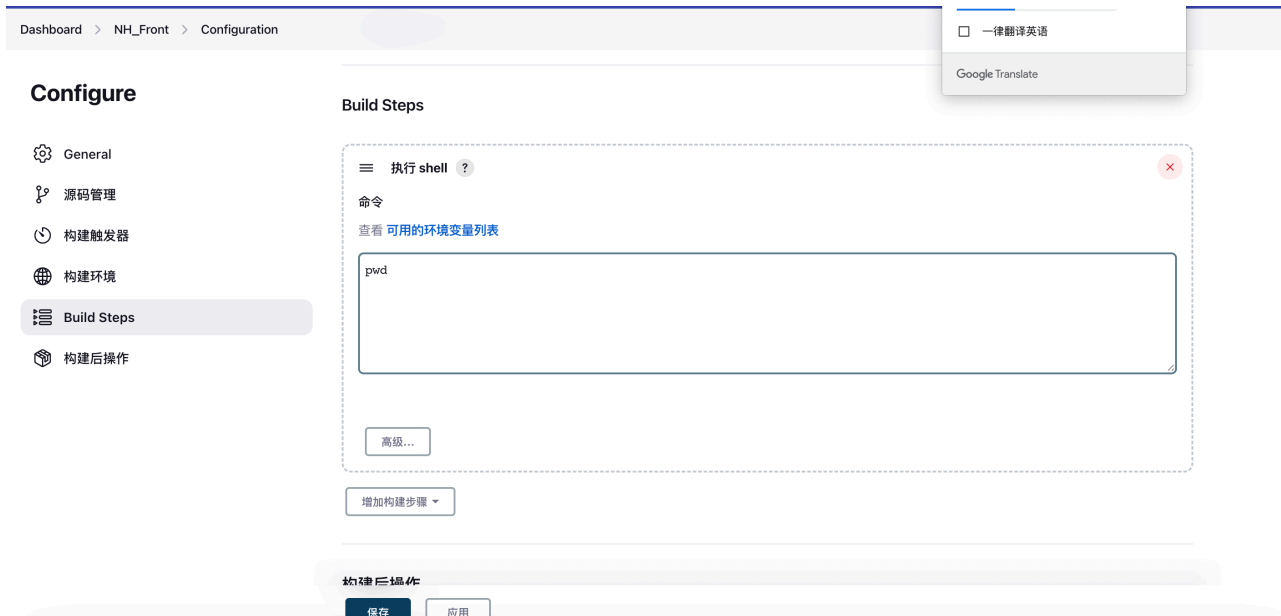
点击新增并选择 secret text 选项，在新出现的选项中选择添加的权限用户



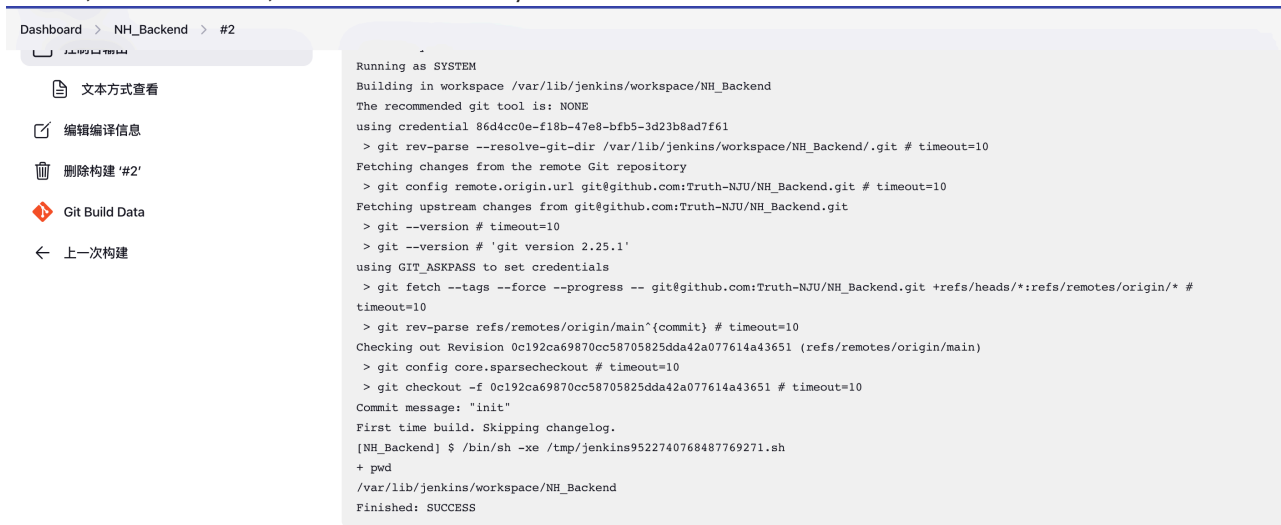
7.5 构建配置，添加执行 shell



既然可以执行 shell 命令，可以先来执行 pwd，看下默认的工作目录是在哪里。



保存后，点击立即构建，就会在 build history 下面看到本次构建的 ID。可以点击查看控制台输出。



8. 编写shell命令

8.1 前端

```
npm install
# 将前端打包成dist
npm run build
```

```
# 下面命令需要在服务器上手动执行
docker ps
docker stop container_id
docker container rm container_id
# dist文件夹必须是绝对路径
```

```
# 这一步之前需要确保jenkins用户可以执行sudo命令，不可以执行的话可以在root用户下使用echo 'jenkins ALL=(ALL) ALL'
```

```
>> /etc/sudoers命令进行添加权限
echo "Smtc123456" | sudo -S docker run -p 8081:80 -d -v /var/lib/jenkins/workspace/NH_Frontend/dist:/usr/share/nginx/html nginx
```

8.2 后端

```
mvn clean package
mvn package -B -Dmaven.test.skip=true
netstat -tunlp|grep 8082|awk '{print $7}'|cut -d '/' -f 1|xargs test -z || netstat -tunlp|grep 8082|awk '{print $7}'|cut -d '/' -f 1|xargs kill
nohup java -jar /var/lib/jenkins/workspace/NH_Backend/target/backend-NH-0.0.1-SNAPSHOT.jar --server.port=8082 >/dev/null 2>&1 &
```