

**软件测试课程设计报告**

学 号： 20191001171

班级序号： 111192-12

学生姓名： 滕德淋

指导教师： 张剑波

**中国地质大学地理与信息工程学院软件工程系**

**2021年 10 月**

自动化测试平台

# **系统概述**

## 1.1 系统目标

|  |  |
| --- | --- |
| 序号 | 目标内容 |
| 1 | Gitlab代码托管库搭建 |
| 2 | Jenkins持续集成环境搭建 |
| 3 | Tomcat测试服务器搭建 |
| 4 | Sonarqube代码审查服务器搭建 |

## 1.2 任务分配

|  |  |  |
| --- | --- | --- |
| **序号** | **姓名** | **承担任务和职责** |
| 1 | 滕德淋 | 负责整体框架搭建 |

## 1.3 开发环境与工具

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 类别 | 具体需求 | 备注 |
| 1 | 硬件 | 笔记本电脑 | Win10 |
| 2 | 软件 | IDEA编辑器+Git | 专业版 |
| 3 | 硬件 | 服务器 | 阿里云服务器+腾讯云服务器 |

# 2. 需求分析

## 2.1 需求概述

我们知道在软件工程中有**软件开发生命周期**这个概念，软件开发生命周期又叫做SDLC（Software Development Life Cycle），它是集合了计划、开发、测试和部署过程的集合。

**瀑布模型**是最著名和最常使用的软件开发模型。瀑布模型就是一系列的软件开发过程。它是由制造业繁衍出来的。一个高度化的结构流程在一个方向上流动，有点像生产线一样。在瀑布模型创建之初，没有其它开发的模型，有很多东西全靠开发人员去猜测，去开发。这样的模型仅适用于那些简单的软件开发， 但是已经不适合现在的开发了。

**敏捷开发**（Agile Development） 的核心是迭代开发（Iterative Development） 与 增量开发（Incremental Development） 。**迭代：**对于大型软件项目，传统的开发方式是采用一个大周期（比如一年）进行开发，整个过程就是一次"大开发"；迭代开发的方式则不一样，它将开发过程拆分成多个小周期，即一次"大开发"变成多次"小开发"，每次小开发都是同样的流程，所以看上去就好像重复在做同样的步骤。**增量：**软件的每个版本，都会新增一个用户可以感知的完整功能。也就是说，按照新增功能来划分迭代。

**持续集成**（ Continuous integration ， 简称 CI ）指的是，频繁地（一天多次）将代码集成到主干。通过持续集成， 团队可以快速的从一个功能到另一个功能，**简而言之，敏捷软件开发很大一部分都要归功于持续集成。**

所以，搭建一个持续集成测试平台是实现敏捷开发流程的重要一步。

* 持续集成可以降低风险，由于持续集成不断去构建，编译和测试，可以很早期发现问题，所以修复的代价就少；
* 对系统健康持续检查，减少发布风险带来的问题；
* 减少重复性工作；
* 持续部署，提供可部署单元包；
* 持续交付可供使用的版本；
* 增强团队信心；

## 2.2 功能性需求

|  |  |
| --- | --- |
| 需求 | 需求描述 |
| Gitlab用户权限管理 | 不同的用户登录可以看见不同的组、人员、项目、上传等信息 |
| Gitlab代码上传管理 | 开发人员在本地开发环境中更新软件版本或者修复bug后可上传到Gitlab代码仓库中。 |
| Git hook自动触发构建 | 就是利用Gitlab的webhook实现代码push到仓库，立即触发项目自动构建。 |
| Gitlab秘钥配置 | 可配置ssh秘钥，进行git的使用。 |
| Jenkins用户权限管理 | 用户可以使用自己的帐号进行登陆访问，提交请求。 |
| Jenkins凭证管理 | 凭据可以用来存储需要密文保护的数据库密码、Gitlab密码信息、Docker私有仓库密码等，以便Jenkins可以和这些第三方的应用进行交互。 |
| Jenkins插件管理 | Jenkins本身不提供很多功能，我们可以通过使用插件来满足我们的使用。例如从Gitlab拉取代码，使用Maven构建项目等功能需要依靠插件完成。 |
| Jenkins拉取代码 | 持续集成平台需要从代码仓库中拉取代码到当前项目下。 |
| Jenkins编译打包项目 | 在Jenkins集成服务器上，我们需要安装Maven来编译和打包项目。 |
| Jenkins发布项目 | 把项目部署到远程的Tomcat里面。 |
| Tomcat用户权限管理 | 进入Tomcat的管理人员后台，查看当前服务下所拥有的项目。 |
| Sonarqube代码审查 | SonarQube是一个用于管理代码质量的开放平台，可以快速的定位代码中潜在的或者明显的错误。 |
| 用户权限管理 | 用户可以使用自己的帐号进行登陆访问，提交请求。 |
| 测试包管理 | 用户根据自身的具体需求，上传、下载、更新、选择已经部署在平台上的测试包（Suite、Case等不同测试单位），并自动执行。 |
| 并发访问 | 多用户并发请求的执行，彼此之间相互独立，互不干扰。 |
| 状态查询 | 执行状态和历史纪录的查询。 |
| 状态反馈 | 执行完毕后的 Email 通知，含测试结果报告。 |
| 用户体验 | 良好的 Web 页面访问模式。 |

## 2.3 非功能性需求

|  |  |
| --- | --- |
| 需求 | 描述 |
| 用户界面友好 | Web界面应尽量友好 |
| 性能处理 | 程序应能处理100用户并发请求，多用户并发操作 |
| 安全性 | 各个用户之间的操作应当不可见 |

## 2.4 用例模型

### 2.4.1 参与者

|  |  |
| --- | --- |
| 参与者 | 参与者说明 |
| 开发人员 | 开发人员可将每次集成的代码上传至仓库，并查看用例通过测试的情况，下载测试包等 |
| 测试人员 | 测试人员可以通过上传测试代码，在服务器端执行测试用例，利用邮件得到测试结果 |

### 2.4.2 顶层用例图

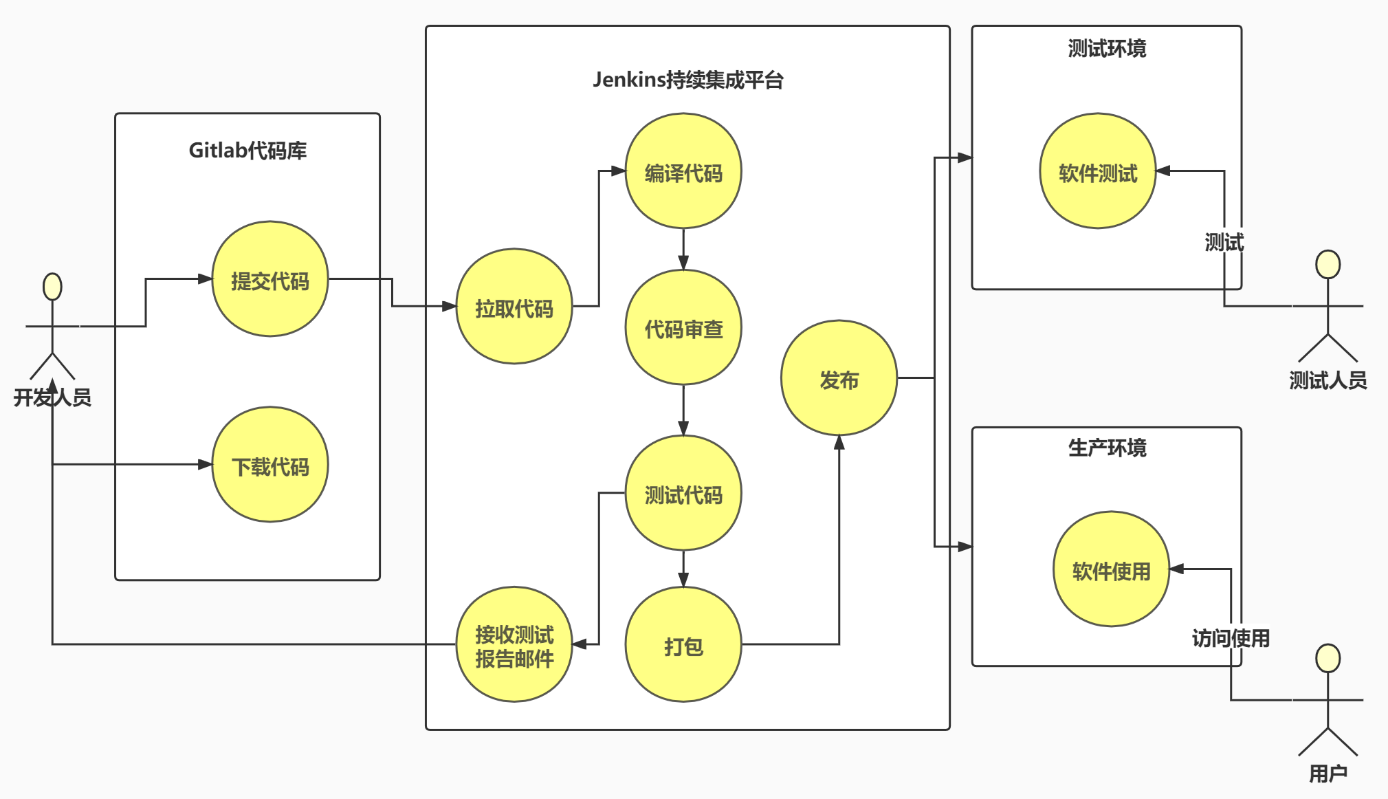


图1-平台用例图

# 3. 概要设计

## 3.1 系统逻辑结构设计

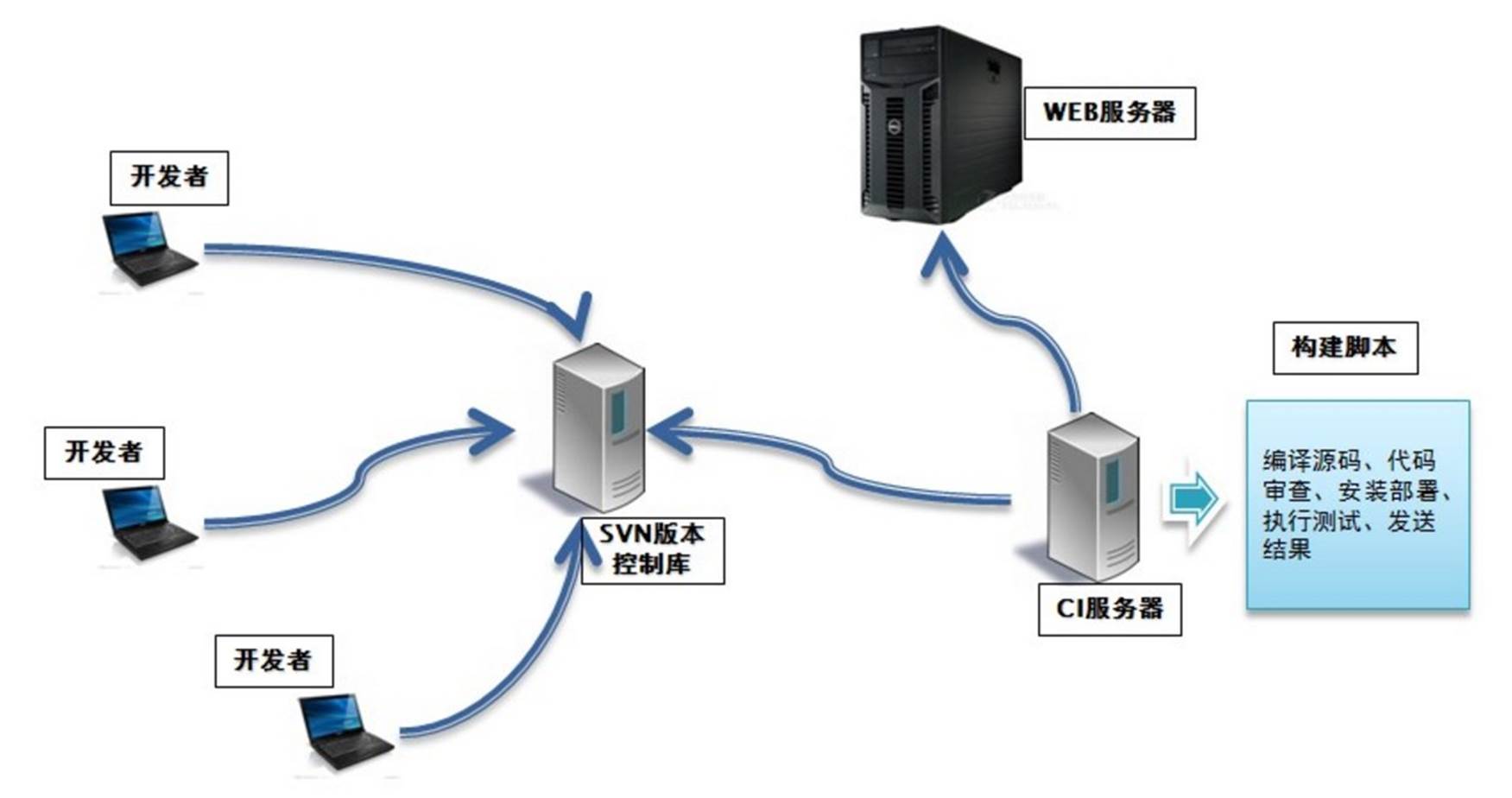


图2-系统架构设计图

用户通过浏览器访问JenkinsWeb服务器，服务器通过扩展插件支持不同的功能，同时能够根据用户自定义的内容自行添加插件。用户通过关联外部仓库进行代码存储，通过Maven项目进行远程测试构建。最后支持邮件扩展发送测试报告。

## 3.2 系统功能设计

### 3.2.1 总体结构

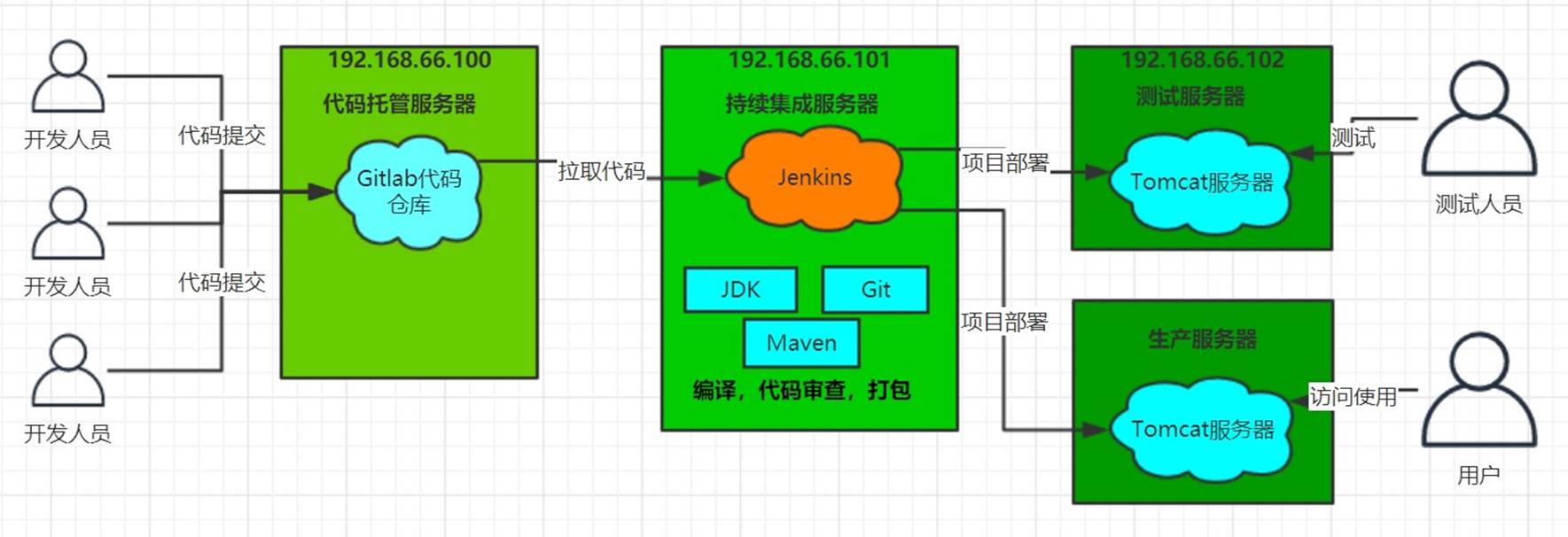


图3-系统模块设计图

Maven环境内部包括内部代码 与 测试用例。测试用例通过服务器调用mvn命令进行测试构建，开发者通过git插件进行代码上传与更新。安全组插件负责用户的登陆与访问。邮件提醒插件关联测试插件，负责发送测试记录。

### 3.2.2 功能描述

#### Gitlab代码库

由于配置教程和功能介绍在文档中不太好展示，所以将配置教程和功能介绍写在了CSDN中：

<https://blog.csdn.net/xiten/article/details/121086411>

本地查看：<Gitlab代码托管服务器.html>

#### Jenkins持续集成平台

由于配置教程和功能介绍在文档中不太好展示，所以将配置教程和功能介绍写在了CSDN中：

<https://blog.csdn.net/xiten/article/details/121088093>

本地查看：<Jenkins持续集成服务器.html>

#### Tomcat测试环境

由于配置教程和功能介绍在文档中不太好展示，所以将配置教程和功能介绍写在了CSDN中：

<https://blog.csdn.net/xiten/article/details/121091462>

本地查看：<Tomcat应用测试服务器.html>

#### Sonarqube代码审查平台

由于配置教程和功能介绍在文档中不太好展示，所以将配置教程和功能介绍写在了CSDN中：

<https://blog.csdn.net/xiten/article/details/121303861>

本地查看：<Sonarqube代码审查平台.html>

# 系统展示

## Gitlab代码仓库

### 用户权限管理

管理员用户在创建用户的时候可以赋予权限：

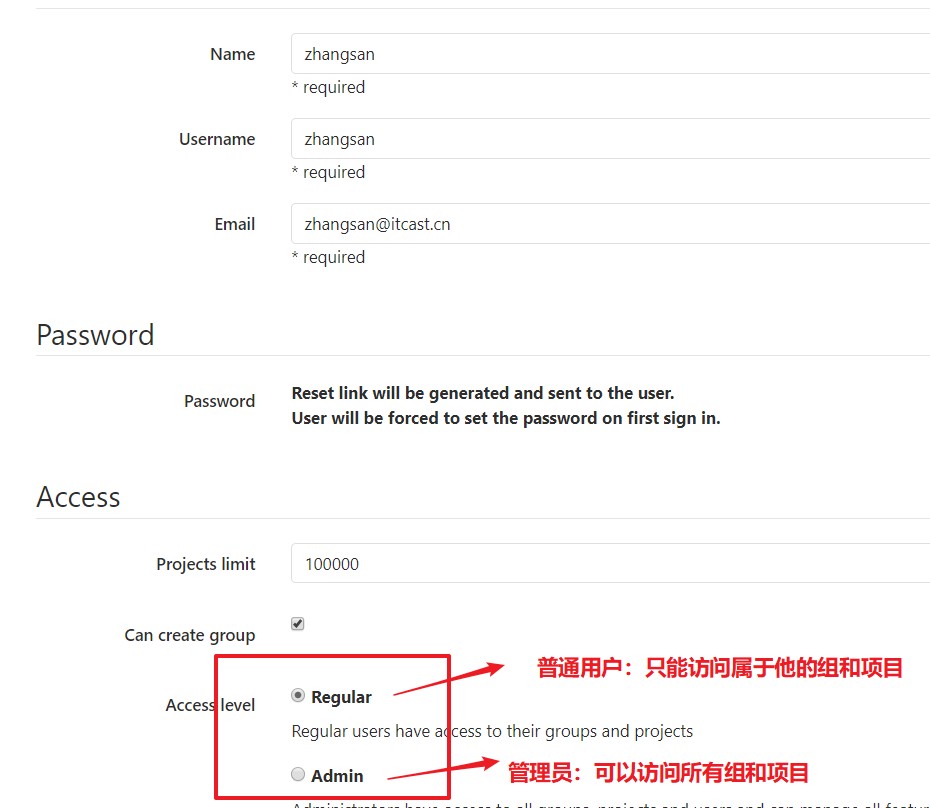


图4-用户权限管理

选择某个用户组，进行Members管理组的成员

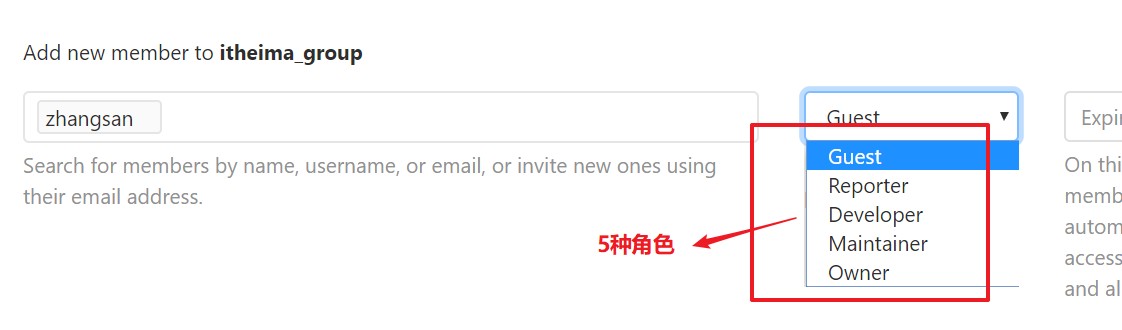


图5-五种用户角色

**Gitlab用户在组里面有5种不同权限：**

Guest：可以创建issue、发表评论，不能读写版本库

Reporter：可以克隆代码，不能提交，QA、PM可以赋予这个权限

Developer：可以克隆代码、开发、提交、push，普通开发可以赋予这个权限

Maintainer：可以创建项目、添加tag、保护分支、添加项目成员、编辑项目，核心开发可以赋予这个权限

Owner：可以设置项目访问权限 - Visibility Level、删除项目、迁移项目、管理组成员，开发组组长可以赋予这个权限

用户登录之后只能看到属于自己的用户组的项目，看不到其他用户组里面的项目。

当用管理员账号登录时：

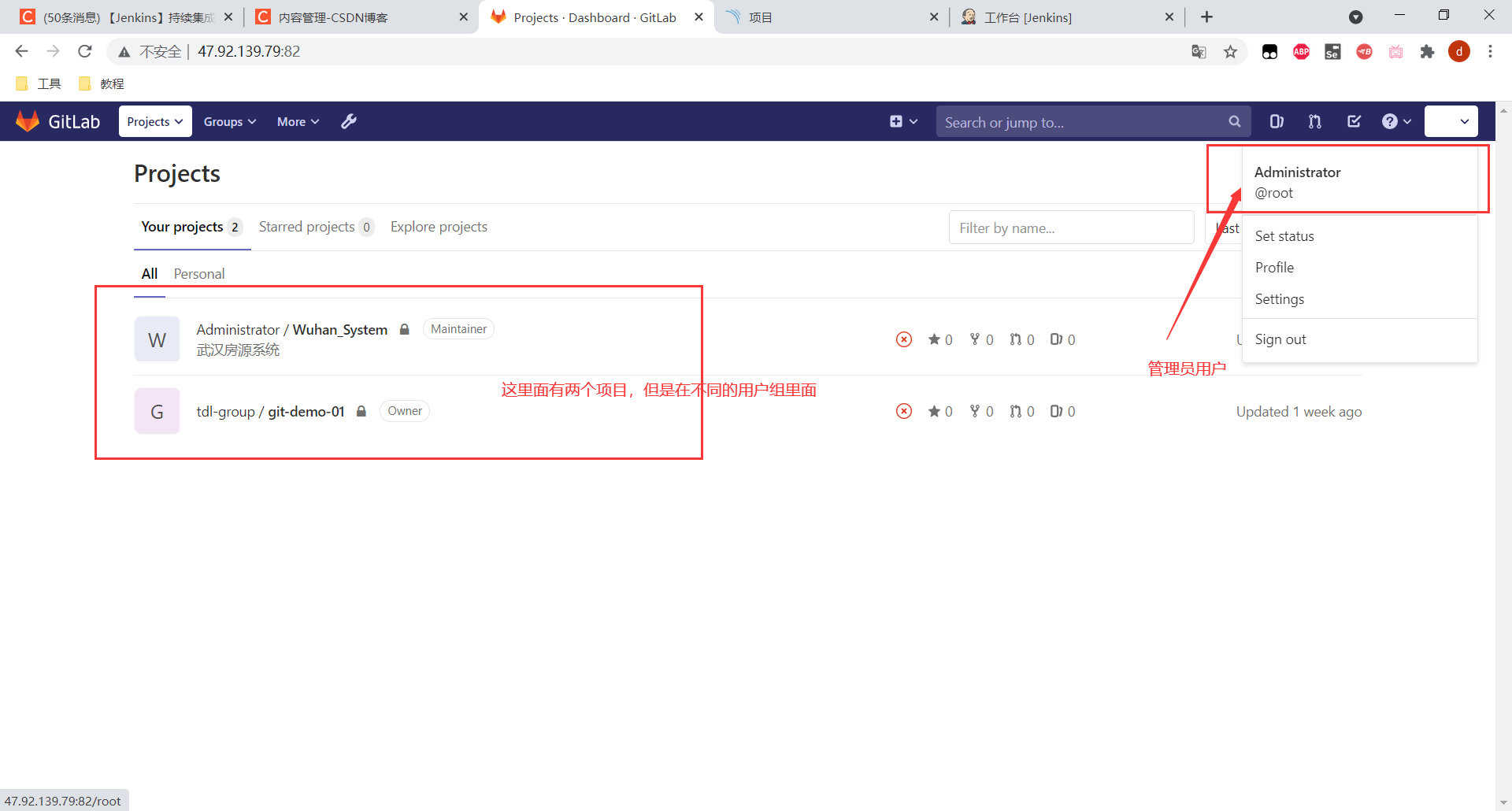


图6-管理员权限

当用zhangsan普通用户账号登录时：

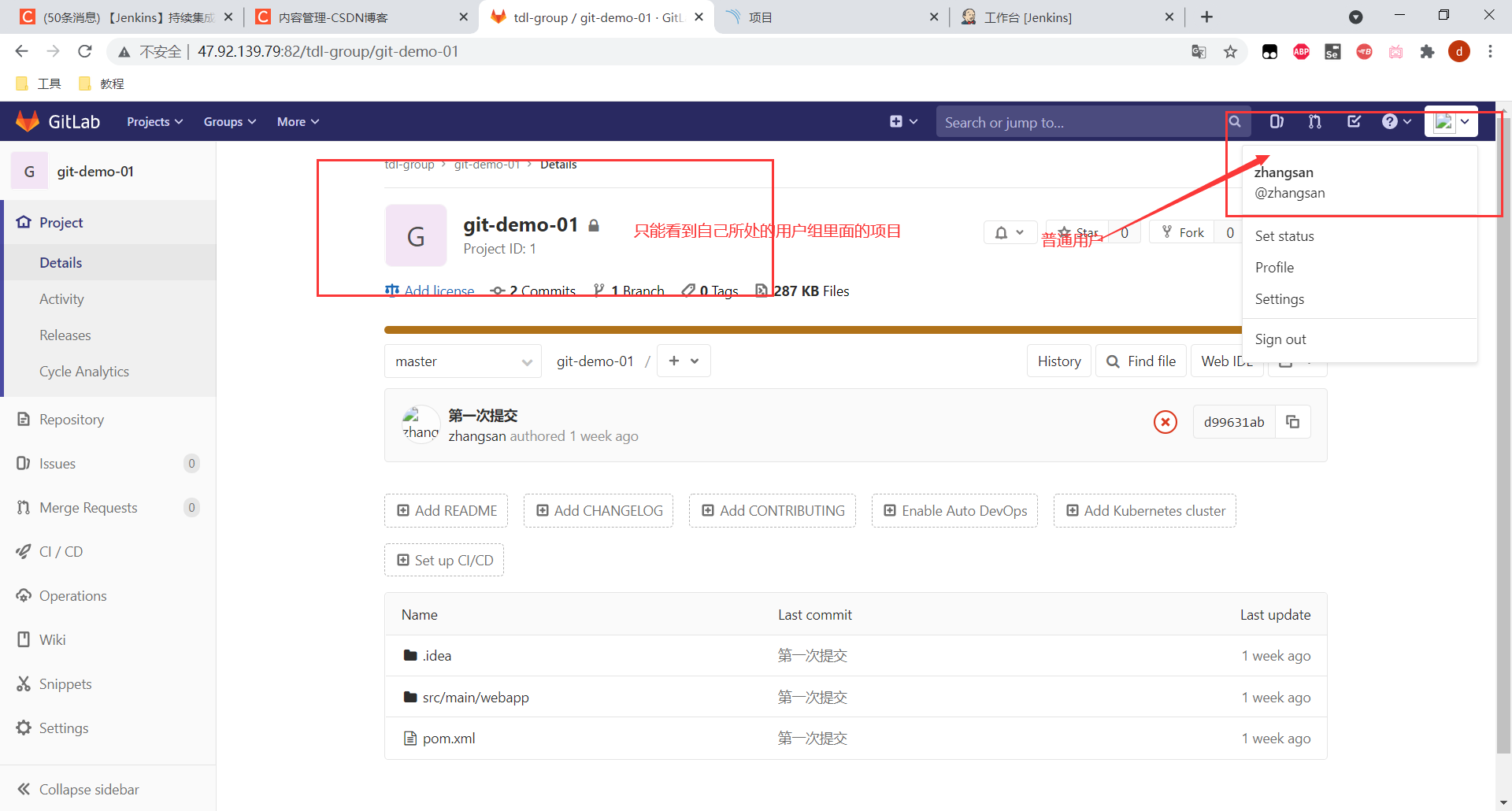
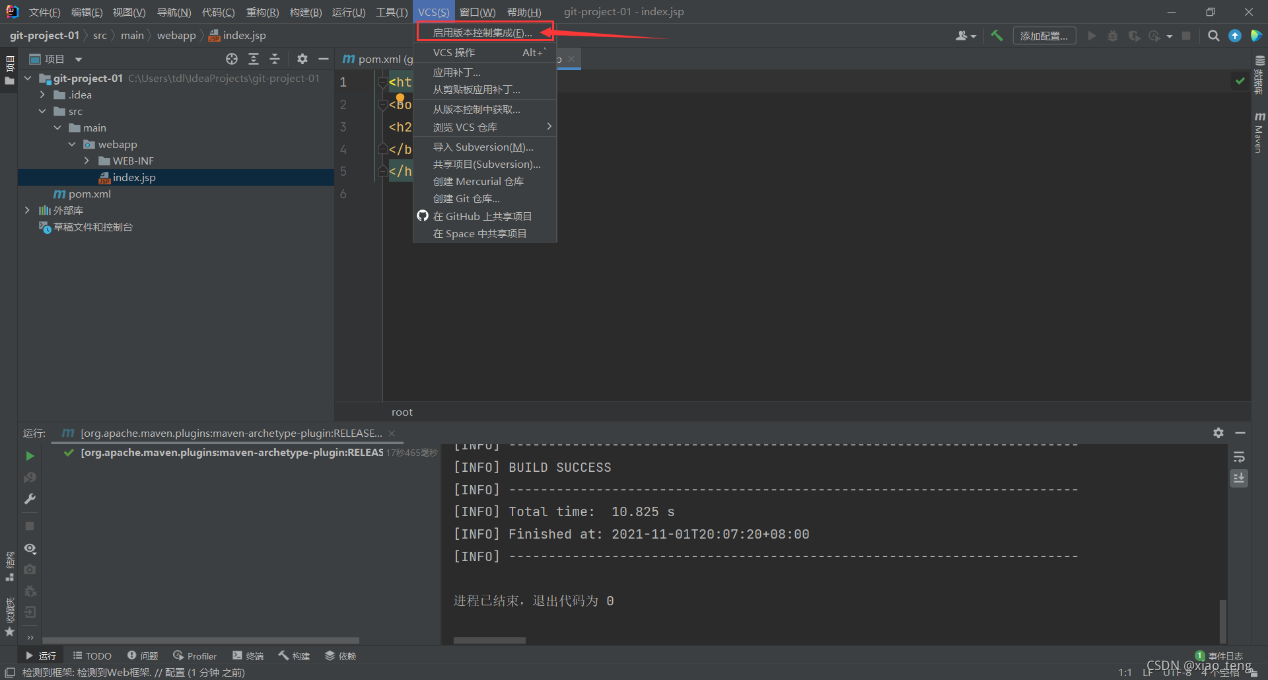
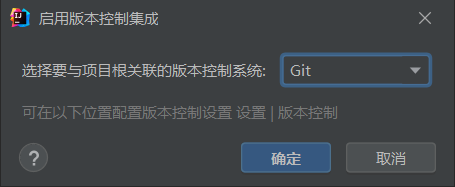


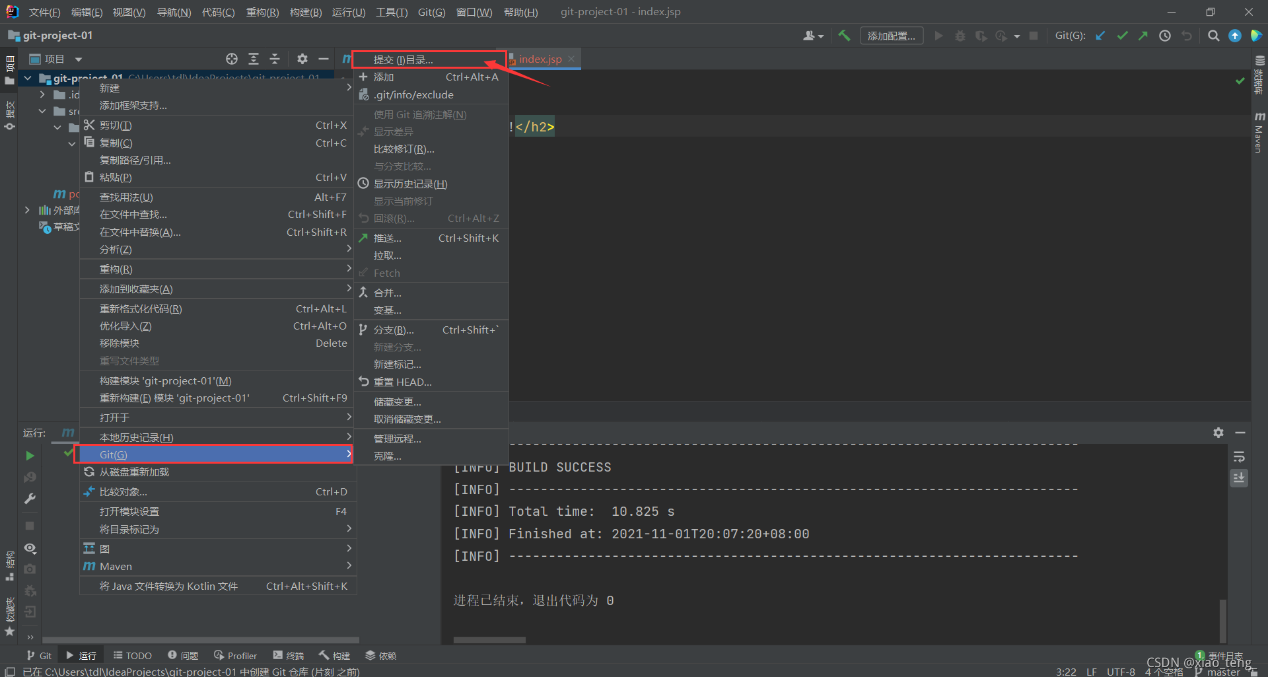
图7-普通用户权限

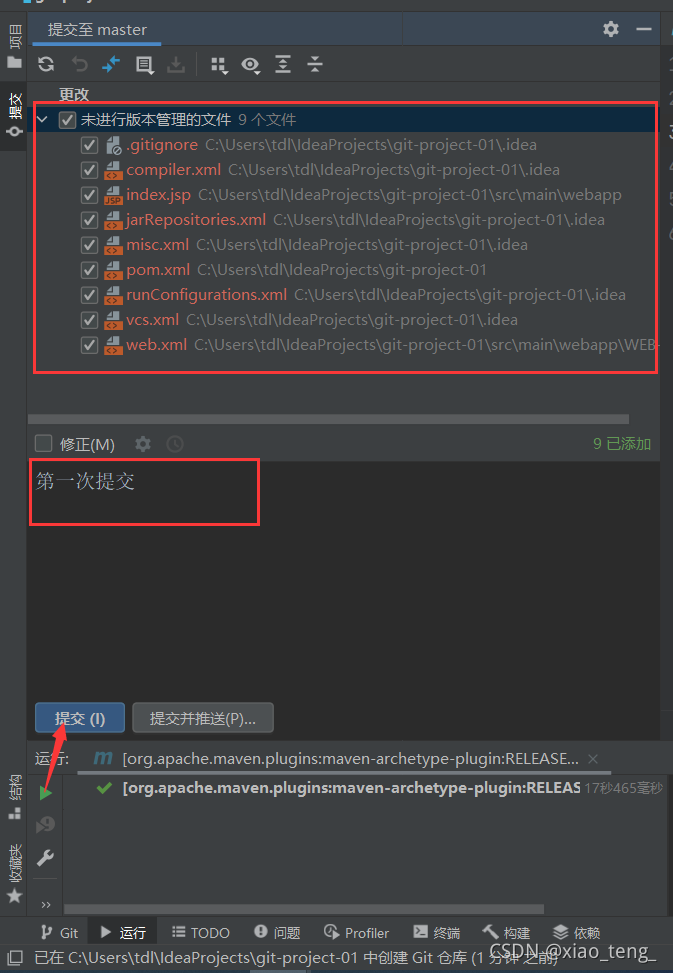
### 代码上传功能

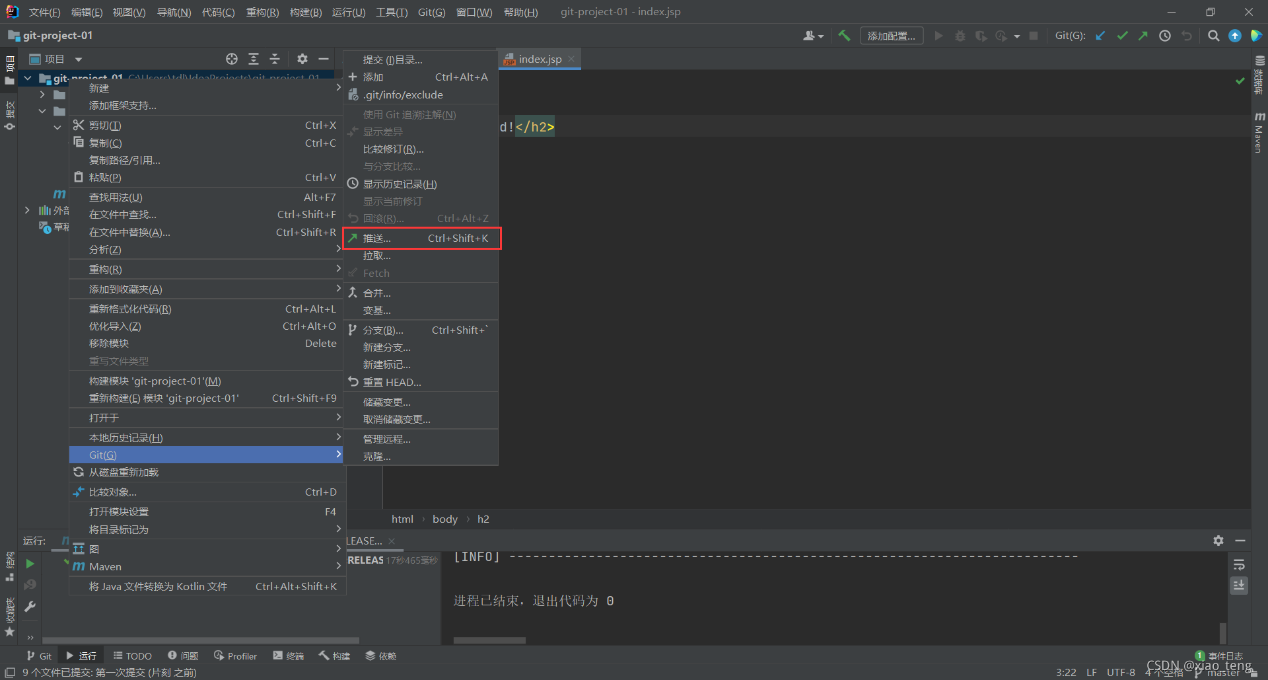
在IDEA开发工具，准备好一个简单的Web应用准备到集成部署。



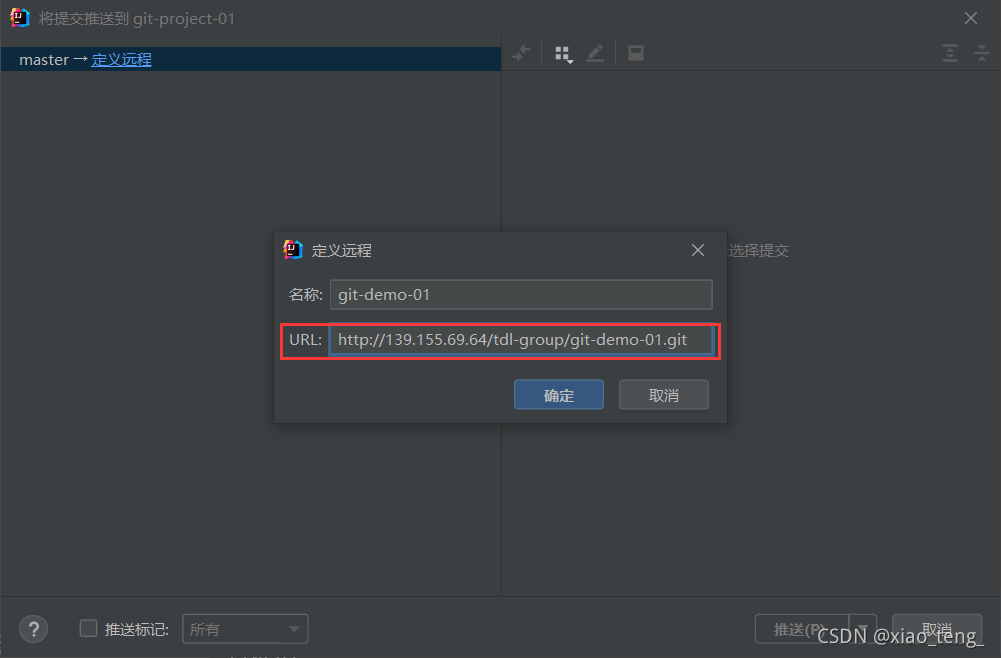












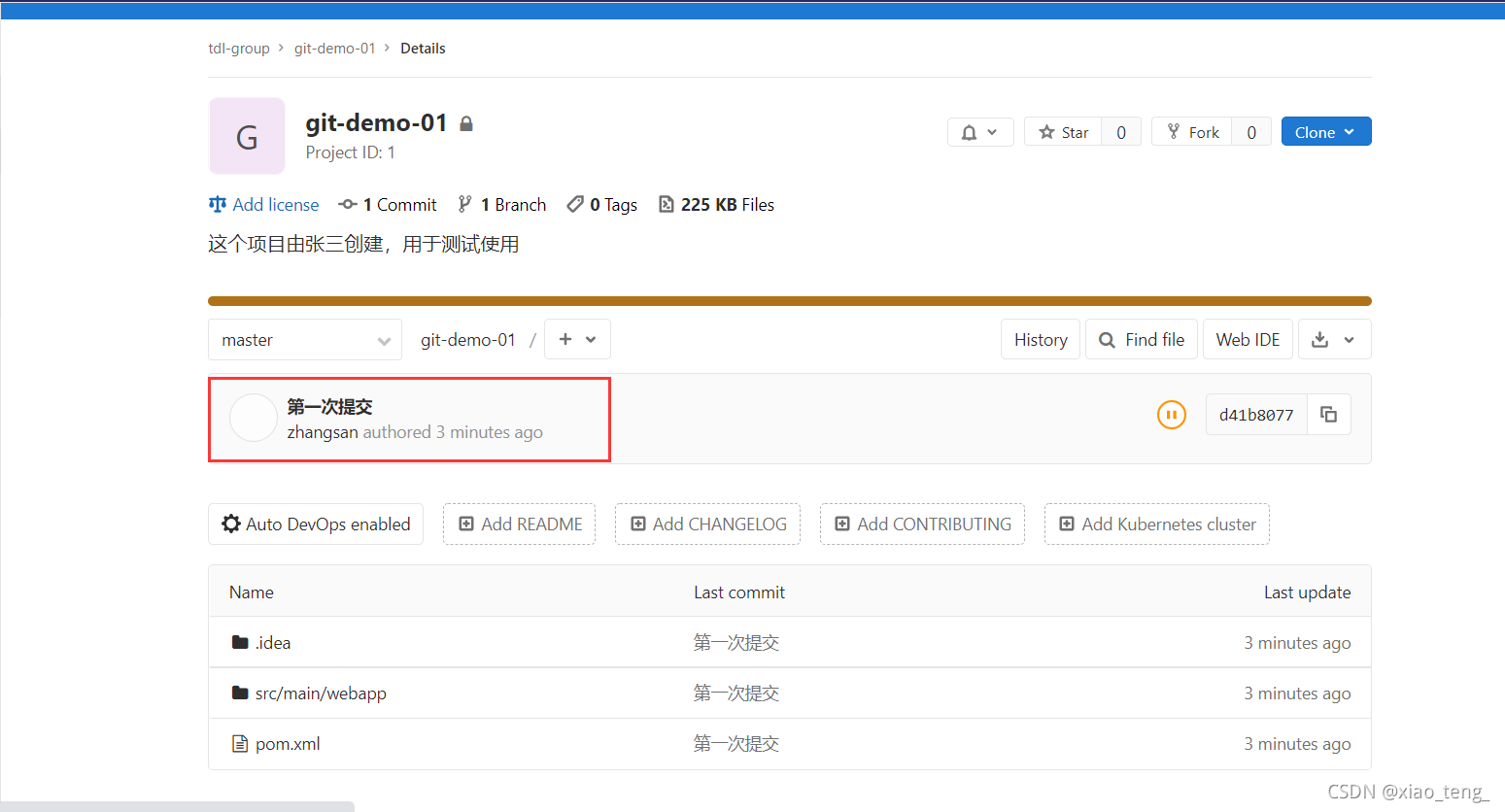


图8-15 本地idea上传代码到gitlab

至此，就可以将本地的代码提交到gitlab仓库里面进行托管了。

### ssh秘钥配置

id\_rsa：私钥文件

id\_rsa.pub：公钥文件

把生成的公钥放在Gitlab中：

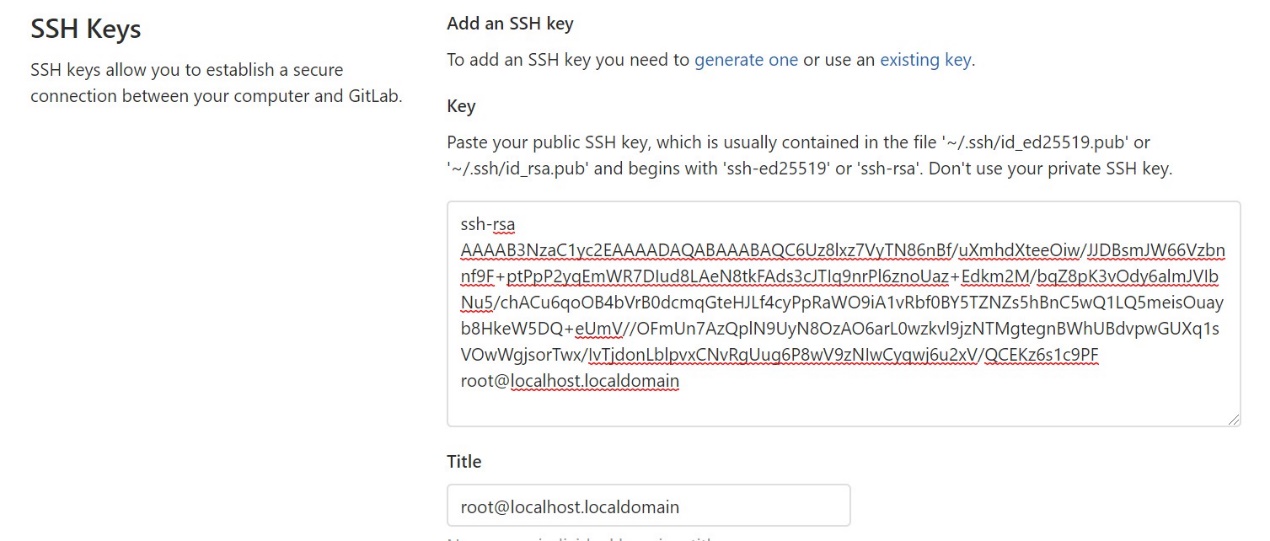


图16-gitlab配置ssh公钥

在Jenkins中添加凭证，配置私钥



图17-Jenkins添加ssh私钥凭证

使用git进行代码拉取：

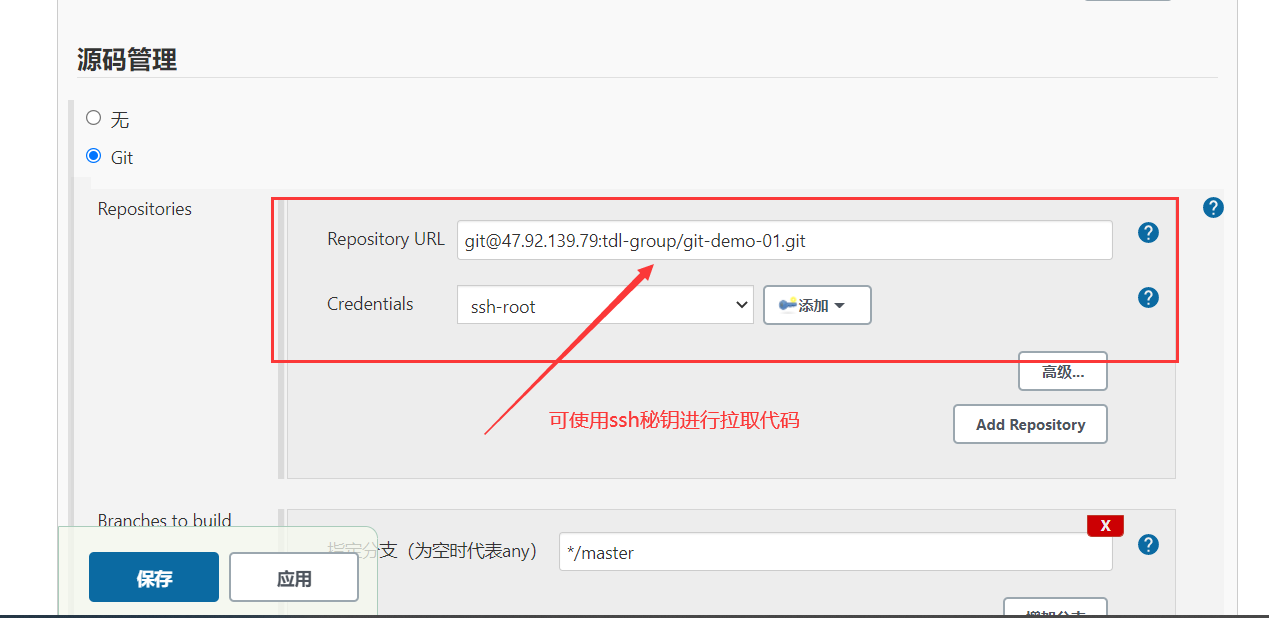


图18-测试ssh秘钥拉取代码

### Web hook功能

这个功能就是利用Gitlab的webhook实现代码push到仓库，立即触发项目自动构建。

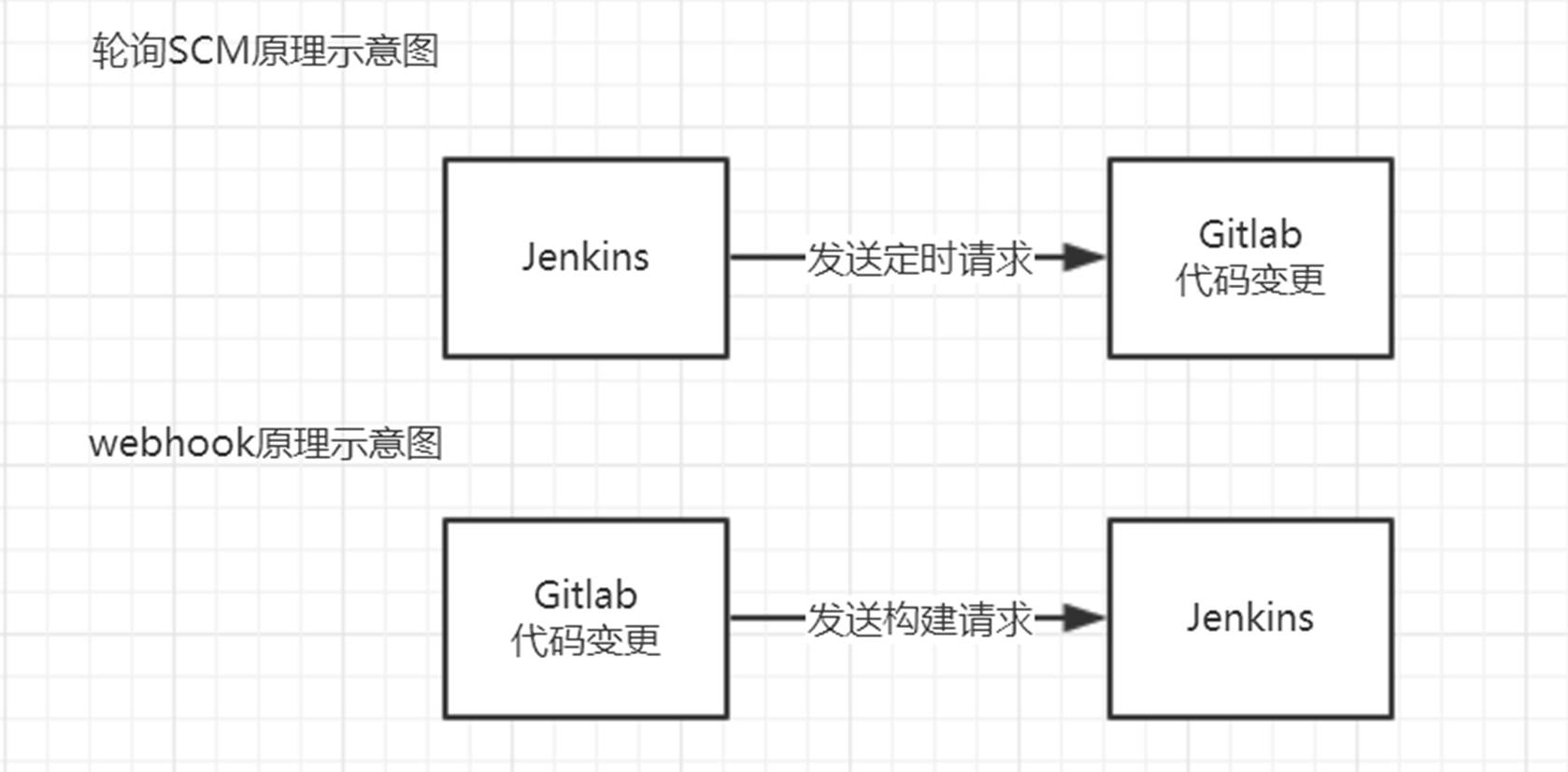


图19-webhook与scm区别解释

Jenkins设置自动构建

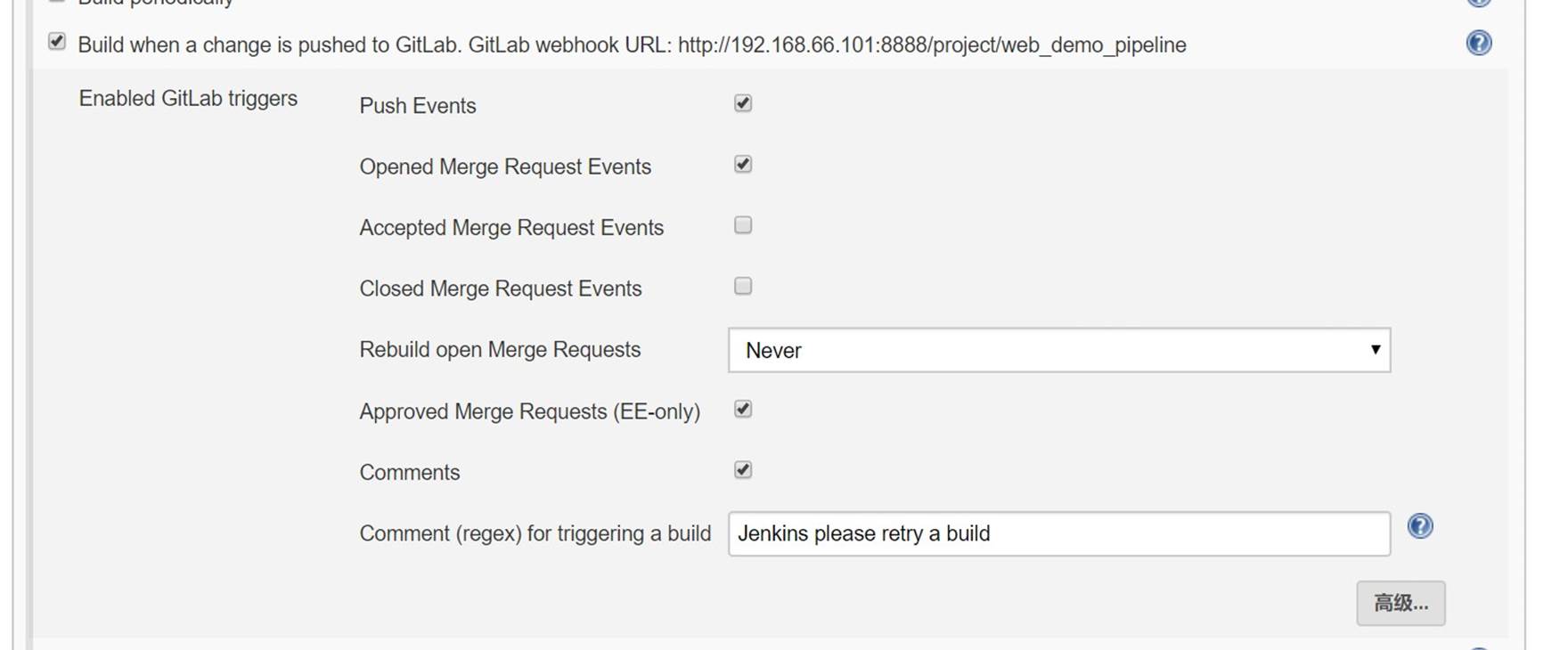
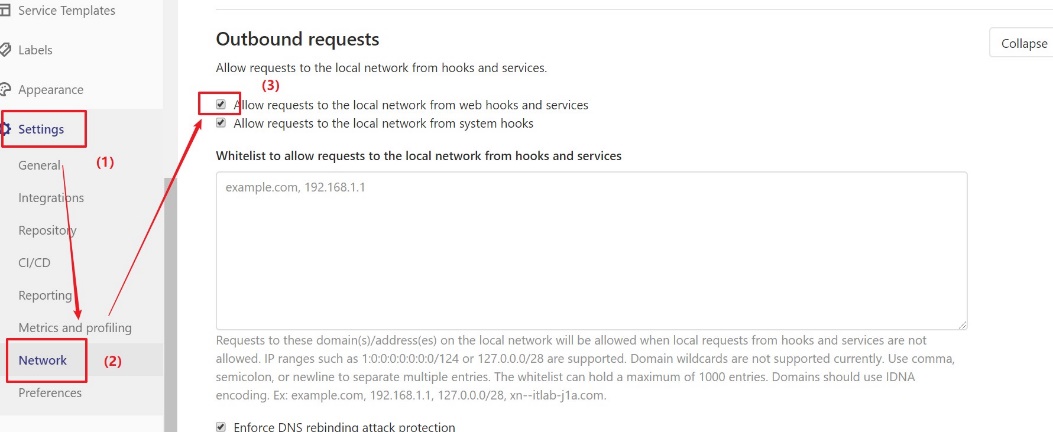


图20-Jenkins配置webhook

Gitlab配置webhook

勾选"Allow requests to the local network from web hooks and services" 让网络钩子允许请求本地网络



在项目添加webhook

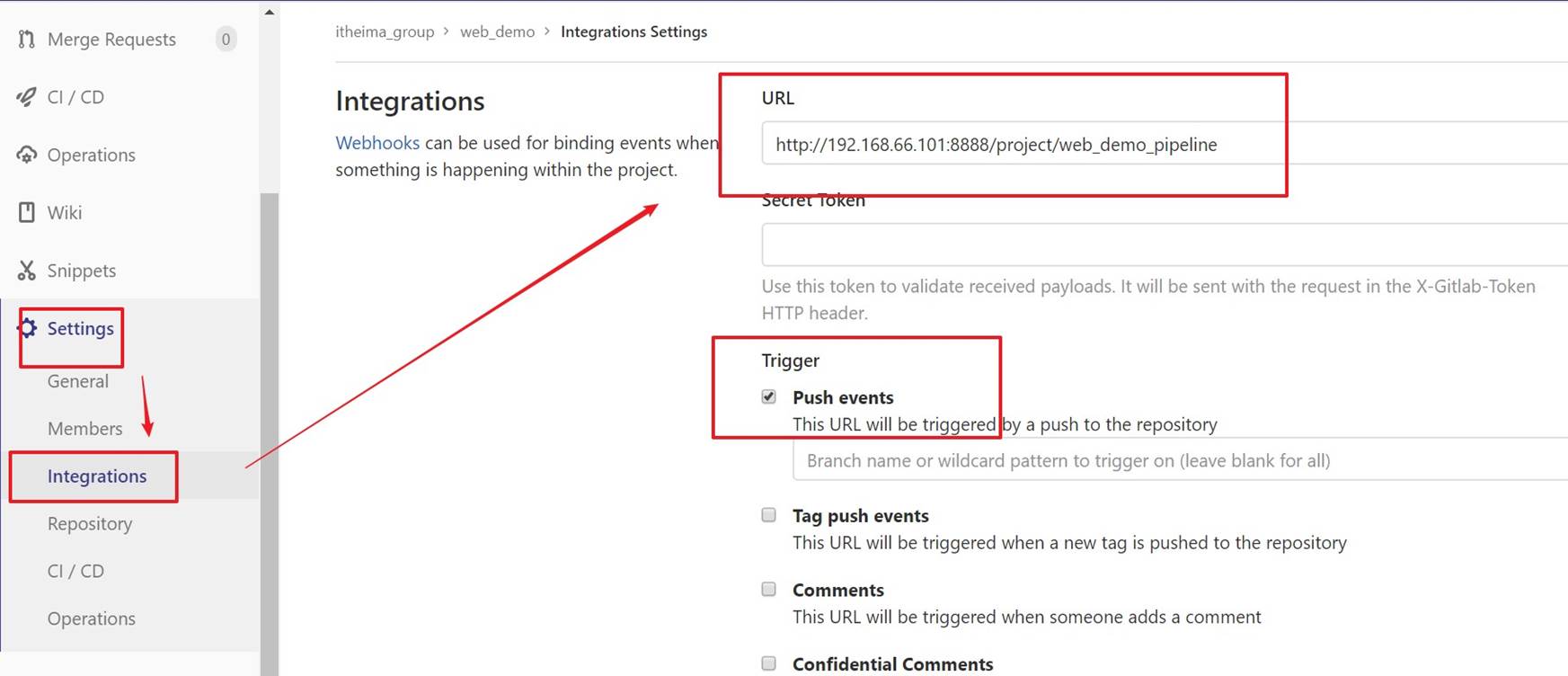




图21-23 gitlab配置webhook

然后在本地修改代码，然后提交到gitlab平台上，然后看是否会进行自动拉取代码进行持续集成。

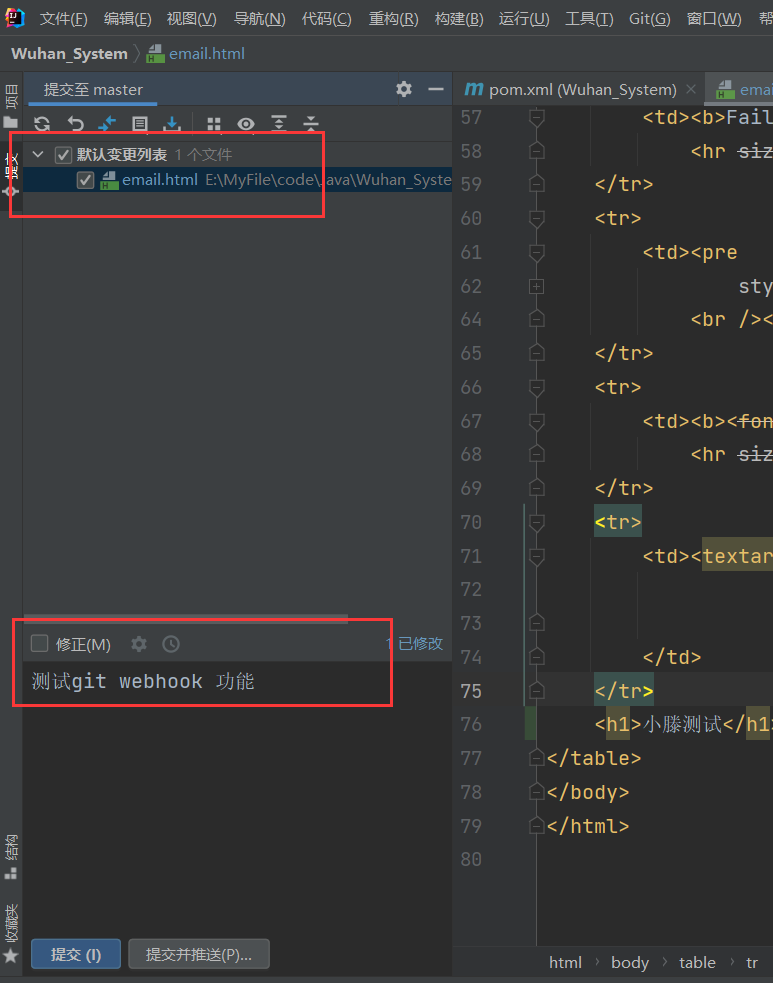


图24-本地idea中修改代码并推送

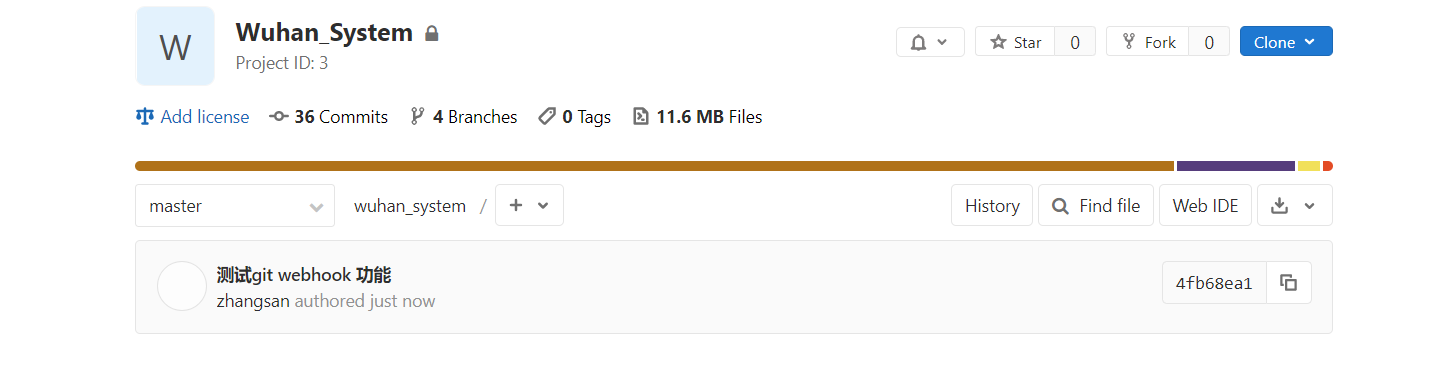


图25-gitlab中代码显示已经修改

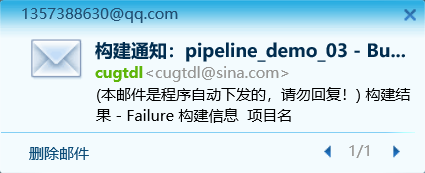


图26-收到编译邮件



图27-Tomcat测试环境中出现部署项目

可以看到并没有登录Jenkins持续集成平台，但是邮箱收到了一个编译消息，并且Tomcat管理后台也多了一个项目，所以webhook功能实现。

## Jenkins持续集成平台

### 用户权限管理

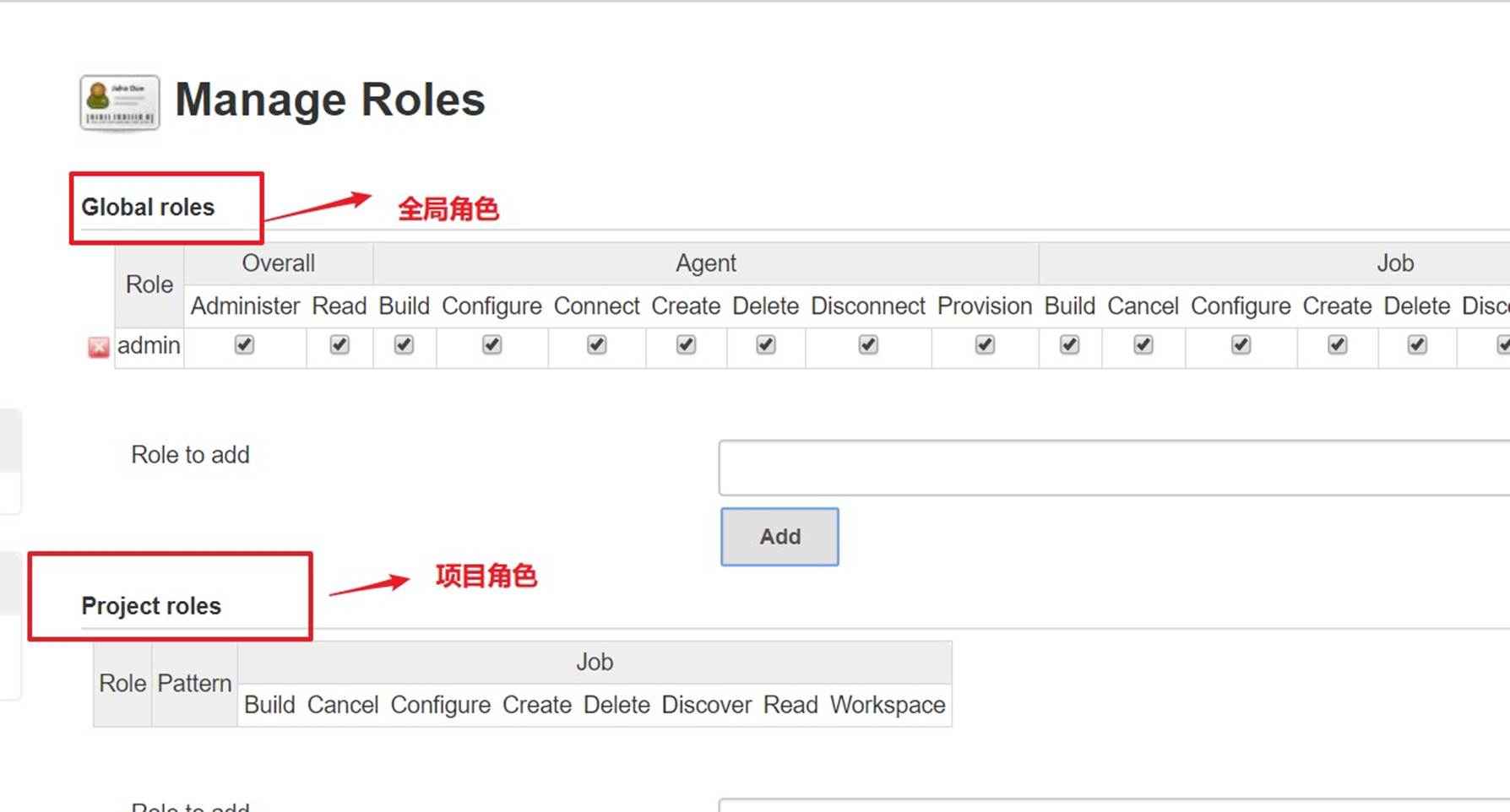


图28-用户权限管理

Global roles（全局角色）：管理员等高级用户可以创建基于全局的角色

Project roles（项目角色）：针对某个或者某些项目的角色

Slave roles（奴隶角色）：节点相关的权限

我们添加以下三个角色：

baseRole：该角色为全局角色。这个角色需要绑定Overall下面的Read权限，是为了给所有用户绑定最基本的Jenkins访问权限。注意：如果不给后续用户绑定这个角色，会报错误：用户名 is missing the Overall/Read permission

role1：该角色为项目角色。使用正则表达式绑定"git.\*"，意思是只能操作git开头的项目。

role2：该角色也为项目角色。绑定"Jenkins.\*"，意思是只能操作Jenkins开头的项目。

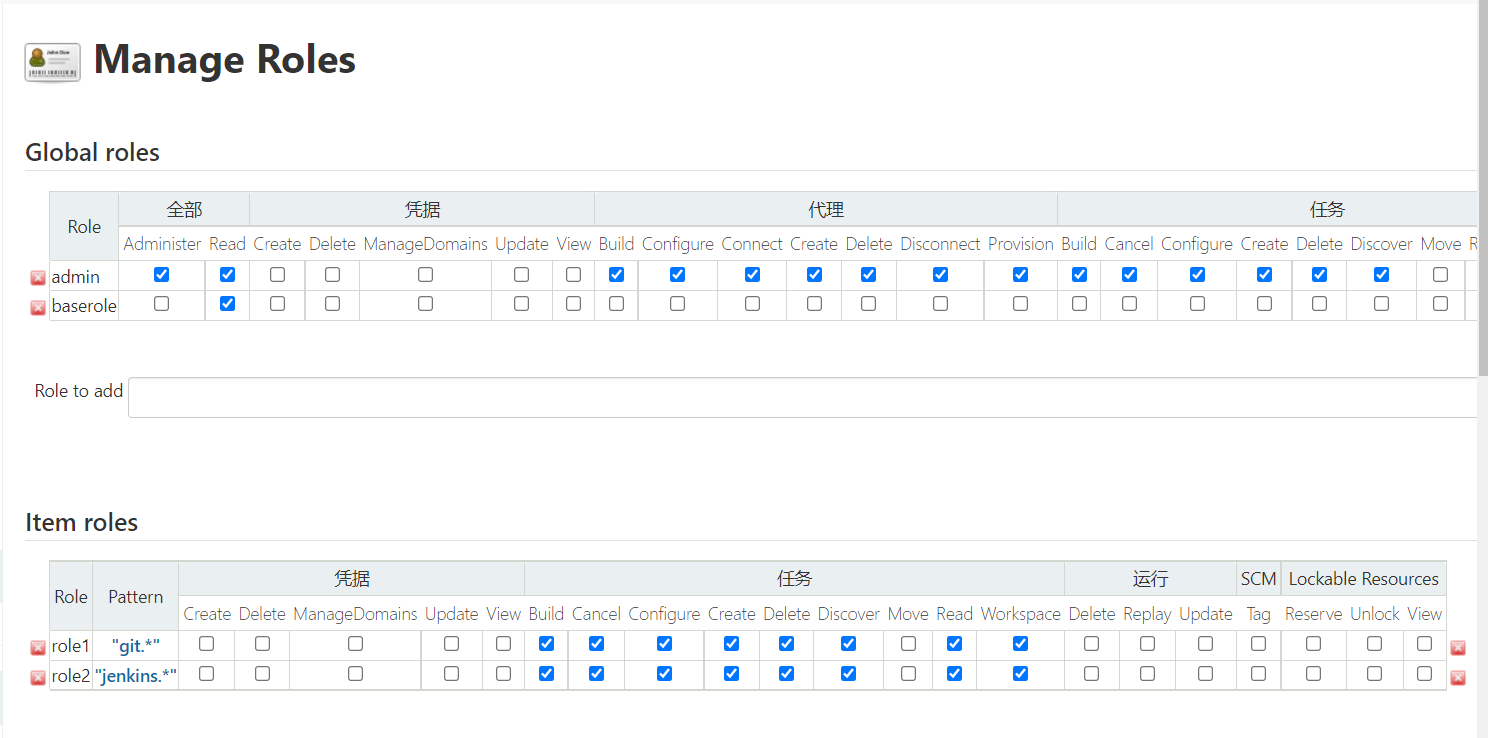


图29-用户角色分配

lsl用户分别绑定baseRole和role1角色

thr用户分别绑定baseRole和role2角色

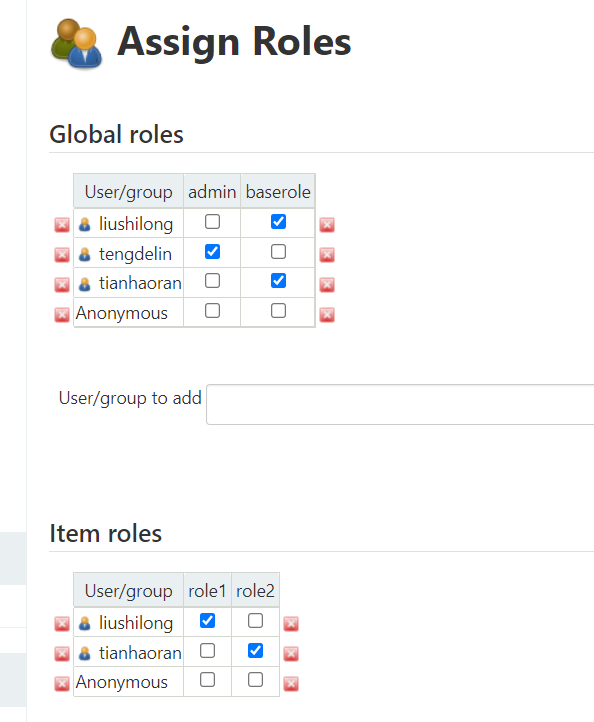


图30-用户分配角色

管理员登录可以看到所以的测试项目:

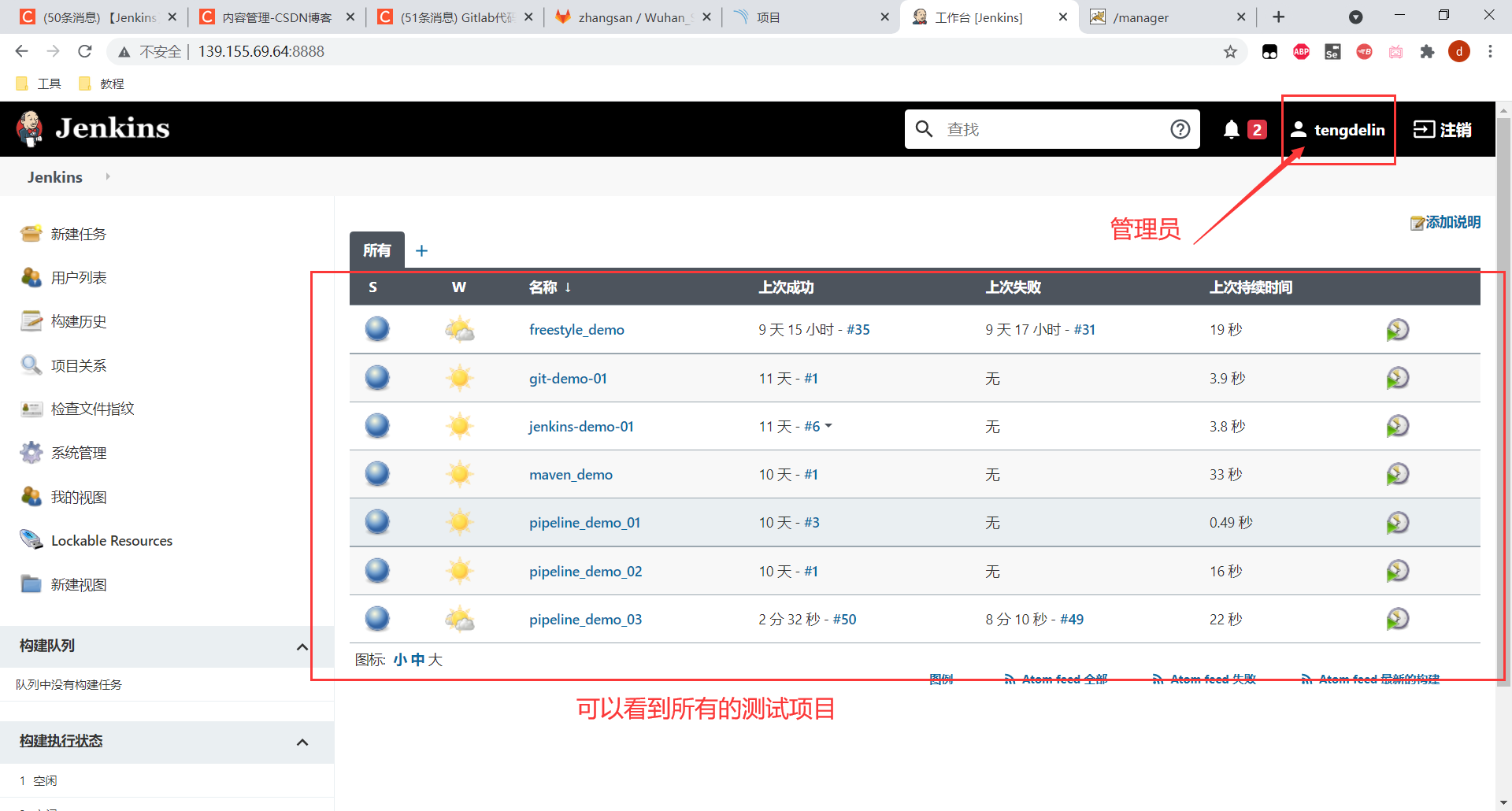


图31-管理员权限

当普通用户登录时，只能看到自己拥有权限的项目：

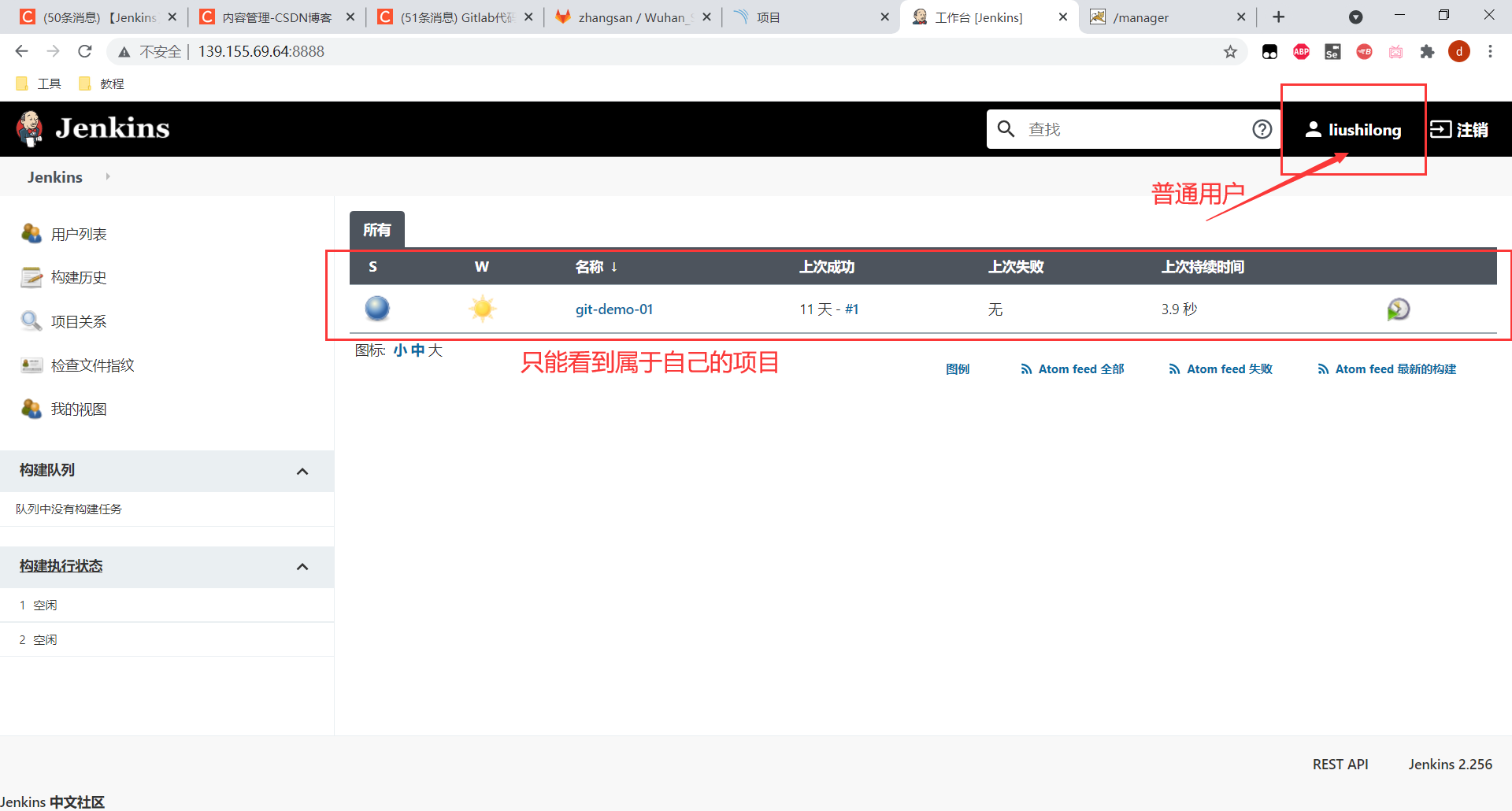


图32-普通用户权限

### 凭证管理

凭据可以用来存储需要密文保护的数据库密码、Gitlab密码信息、Docker私有仓库密码等，以便Jenkins可以和这些第三方的应用进行交互。

要在Jenkins使用凭证管理功能，需要安装Credentials Binding插件

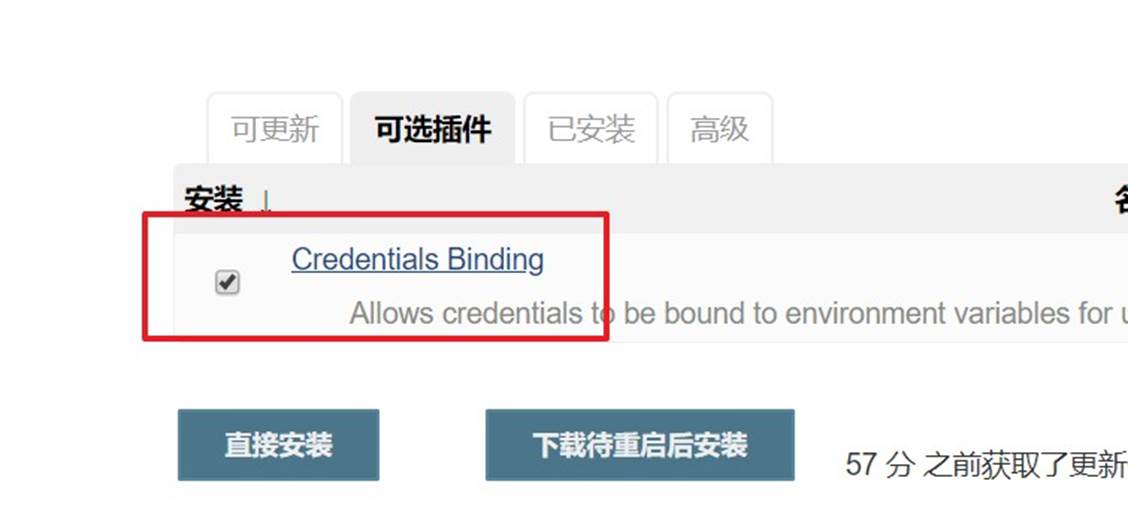


图33-安装凭证管理插件

**可以添加的凭证有5种：**

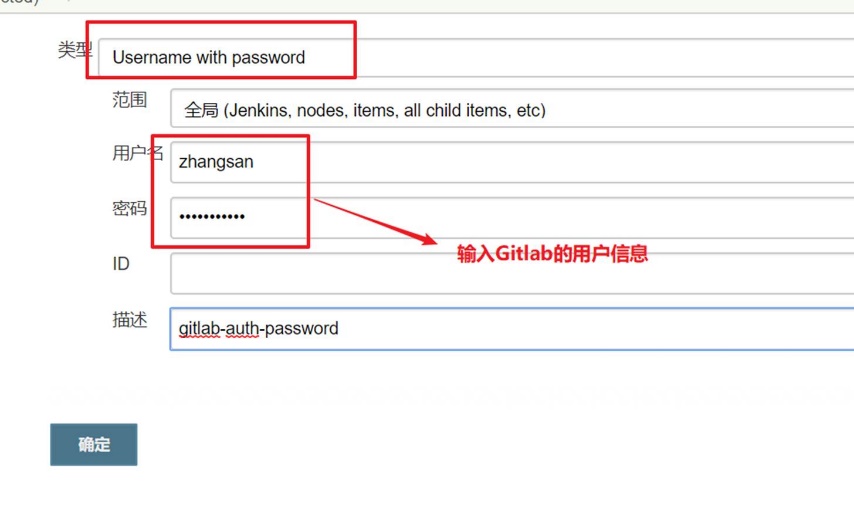
Username with password：用户名和密码

SSH Username with private key： 使用SSH用户和密钥

Secret ﬁle：需要保密的文本文件，使用时Jenkins会将文件复制到一个临时目录中，再将文件路径设置到一个变量中，等构建结束后，所复制的Secret ﬁle就会被删除。

Secret text：需要保存的一个加密的文本串，如钉钉机器人或Github的api token

Certiﬁcate：通过上传证书文件的方式



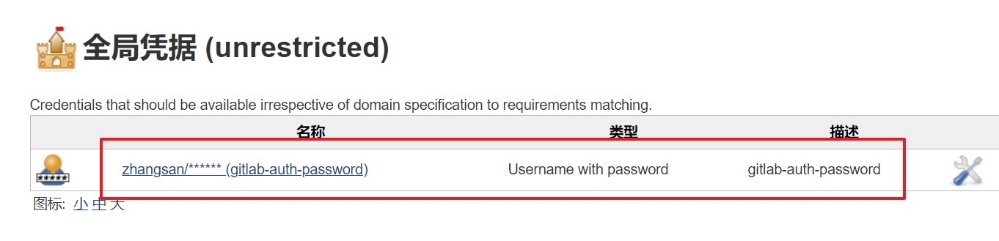


图34-35 添加凭证管理

这样就完成了凭证管理功能。

### 自动化持续集成

Jenkins中自动构建项目的类型有很多，常用的有以下三种：

* 自由风格软件项目（FreeStyle Project）
* Maven项目（Maven Project）
* 流水线项目（Pipeline Project）

每种类型的构建其实都可以完成一样的构建过程与结果，只是在操作方式、灵活度等方面有所区别，在实际开发中可以根据自己的需求和习惯来选择。

下面我就只演示流水线项目的持续化集成：

在这个项目中我配置了自动拉取代码、编译构建、代码测试、代码审查、项目部署、邮件通知流程。

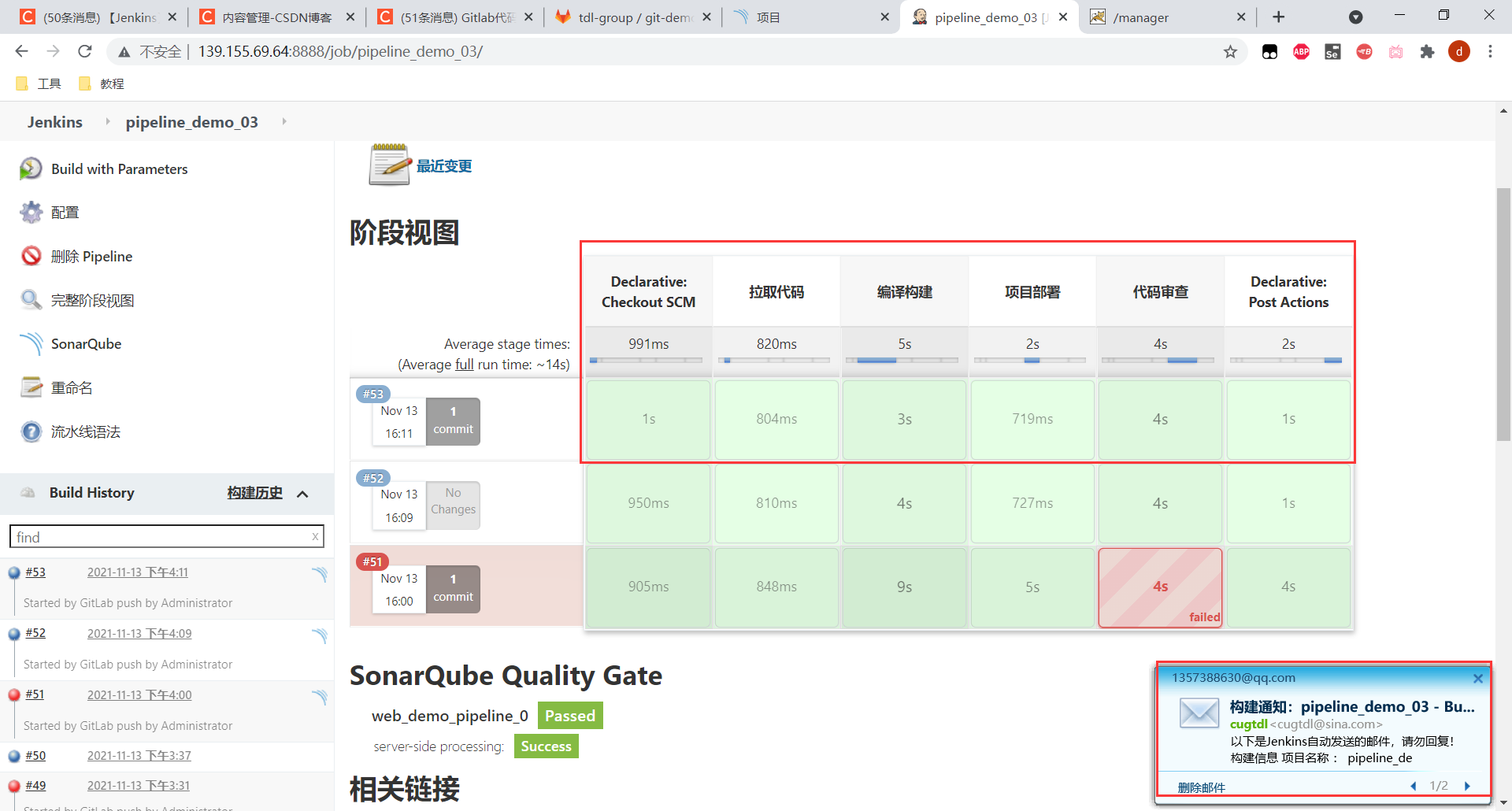


图36-Jenkins中查看流水线项目持续集成流程

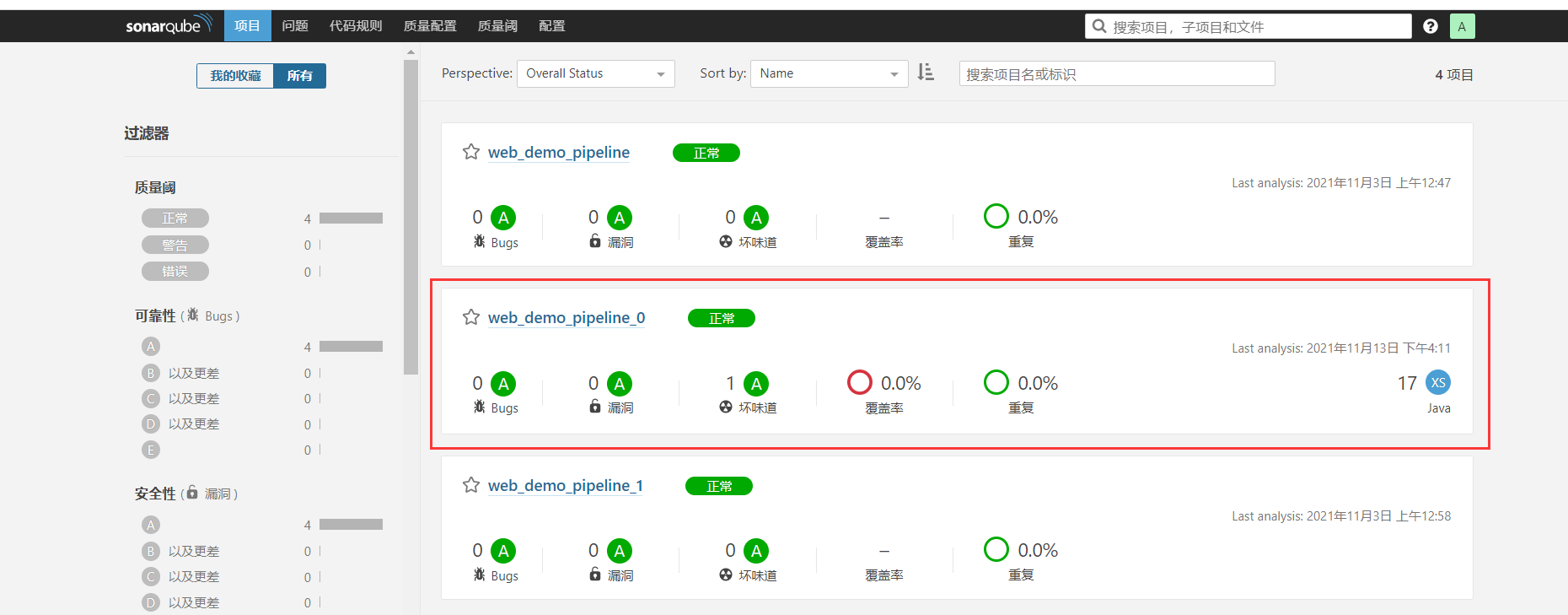


图37-sonarqube中查看代码审查结果

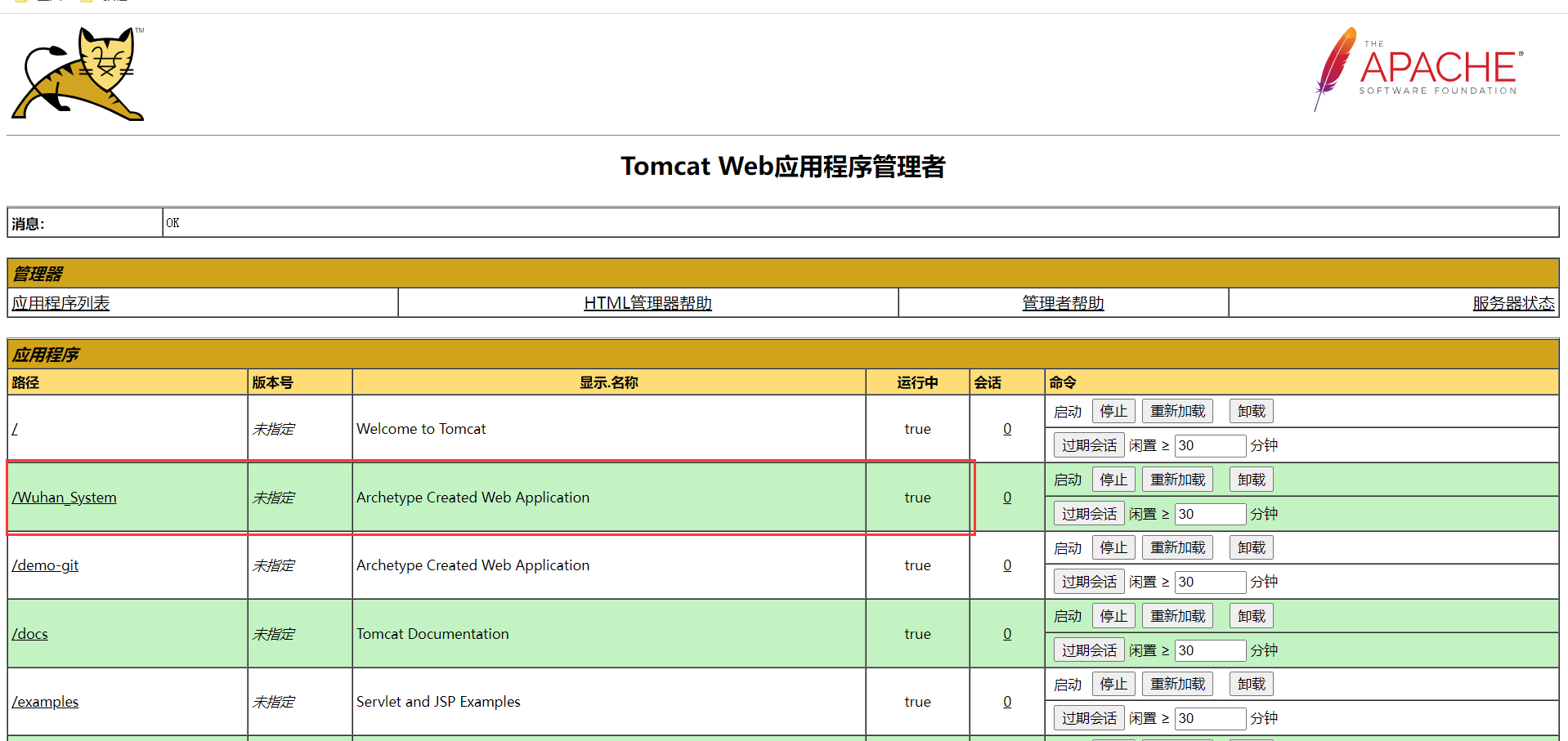


图38-Tomcat后台管理中查看项目部署情况



图39-Jenkins中查看TestNG测试报告

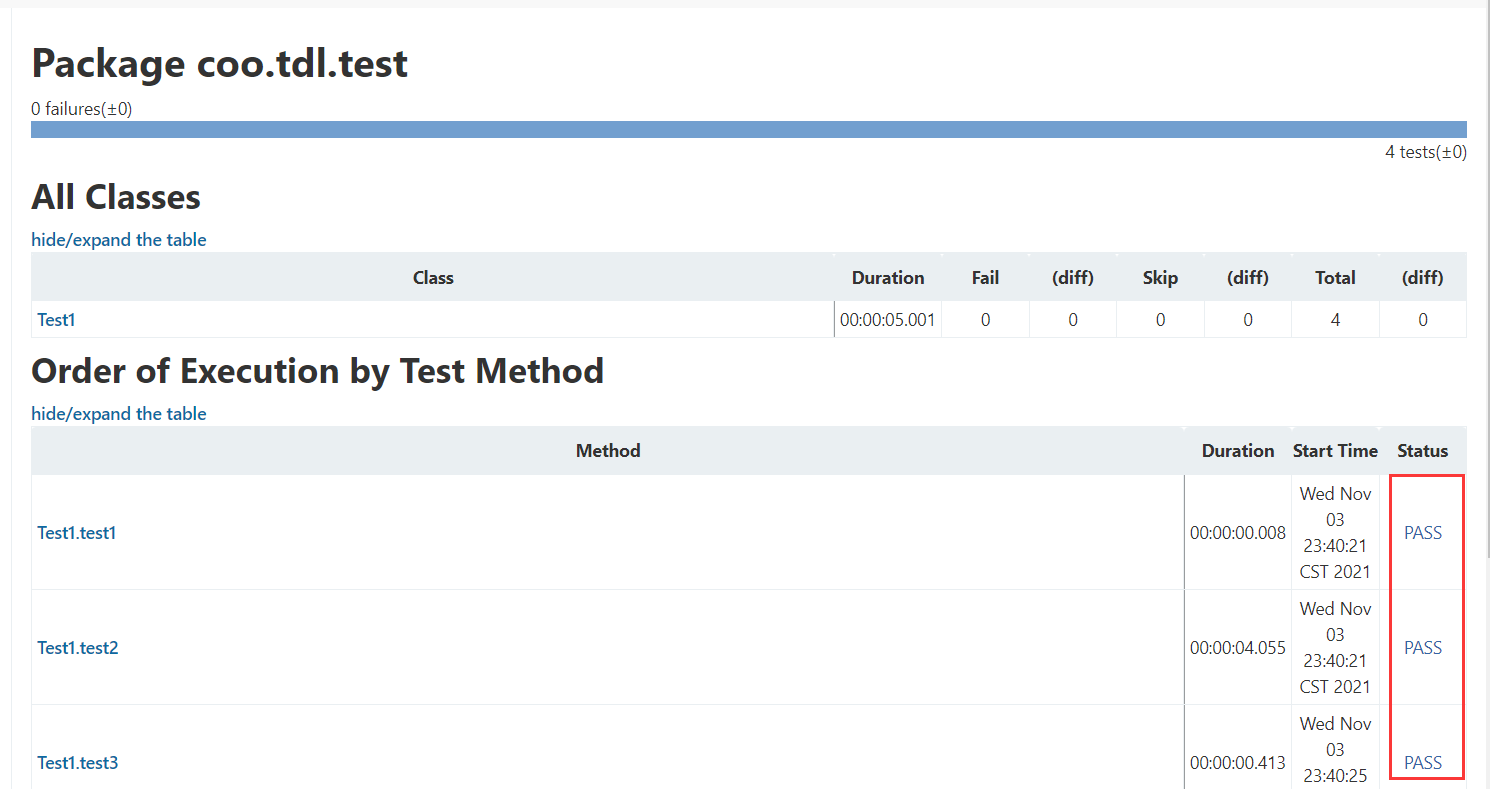


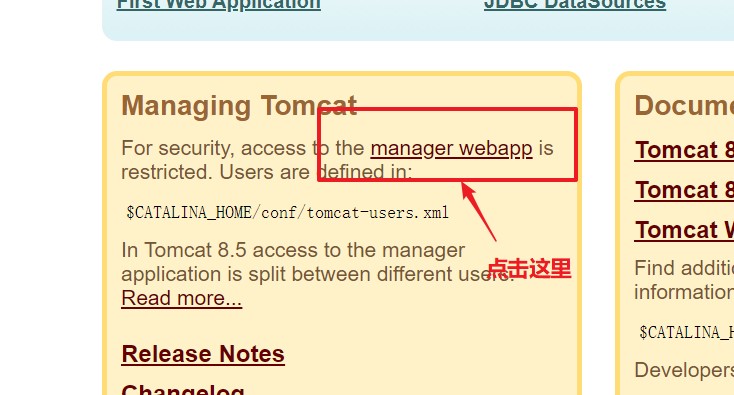
图40-查看测试报告结果

可以看到整个流程是完成了的，完成了自动化持续集成的过程。

## Tomcat测试环境

### Tomcat用户角色权限

默认情况下Tomcat是没有配置用户角色权限的



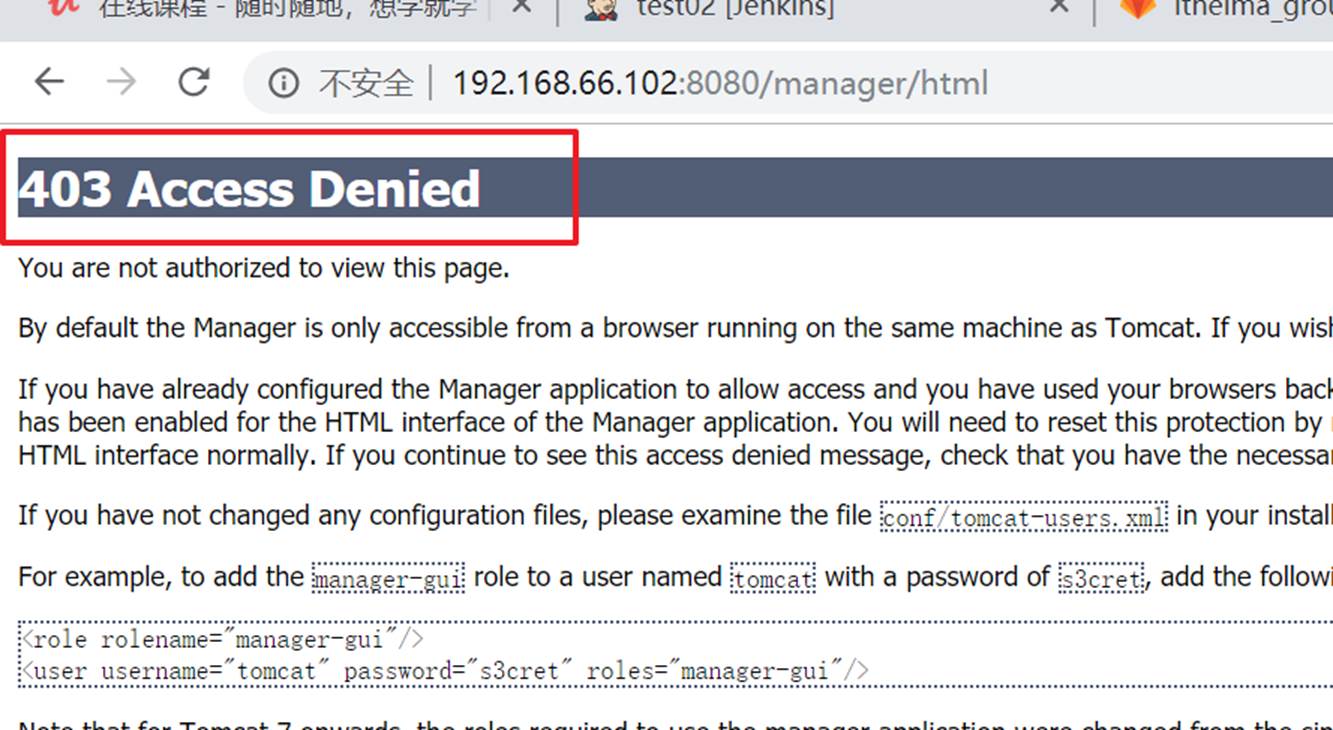


图41-42 Tomcat测试环境默认情况

但是，后续Jenkins部署项目到Tomcat服务器，需要用到Tomcat的用户，所以修改tomcat以下配置，添加用户及权限

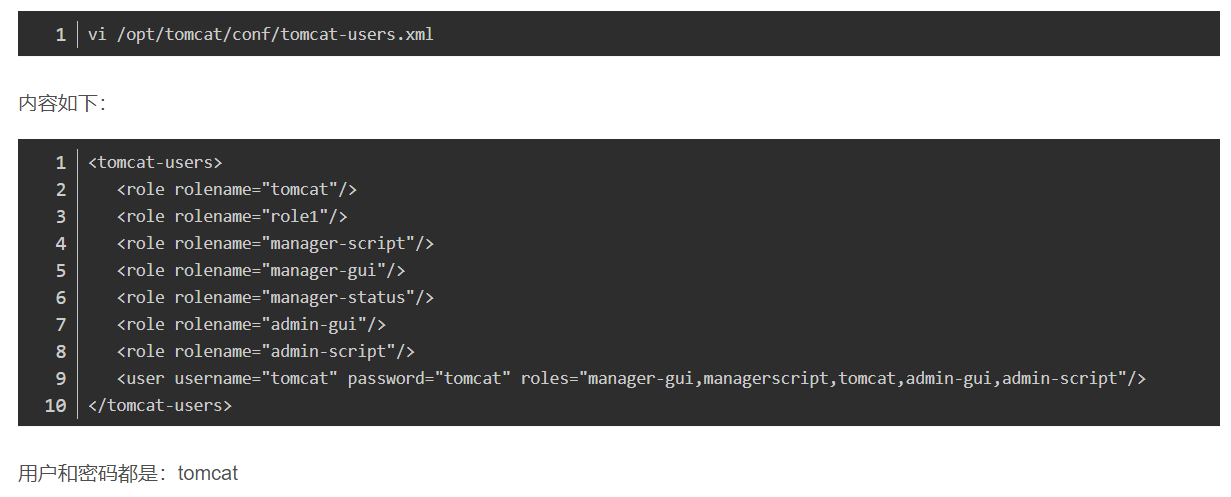


图43-配置Tomcat管理权限

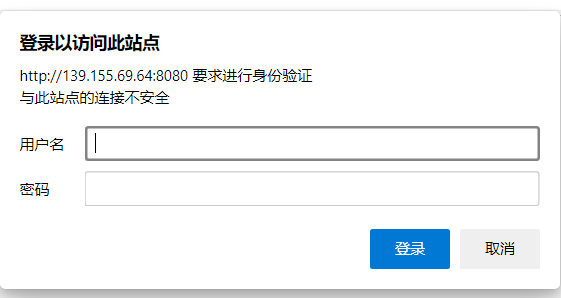


图44-登录Tomcat后台管理系统

登录之后就可以进入管理界面：

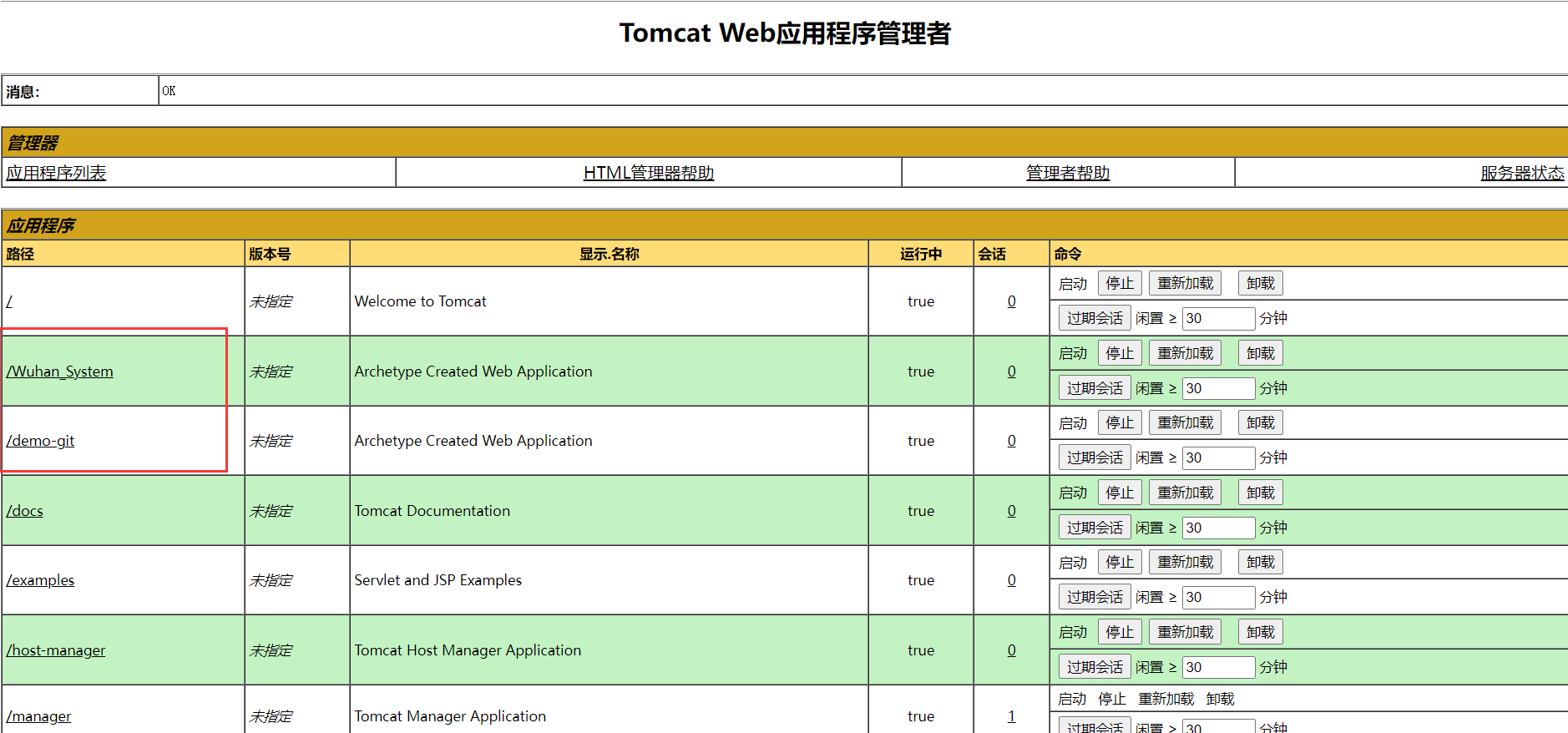


图45-Tomcat后台管理界面

## Sonarqube代码审查平台

### Jenkins代码审查

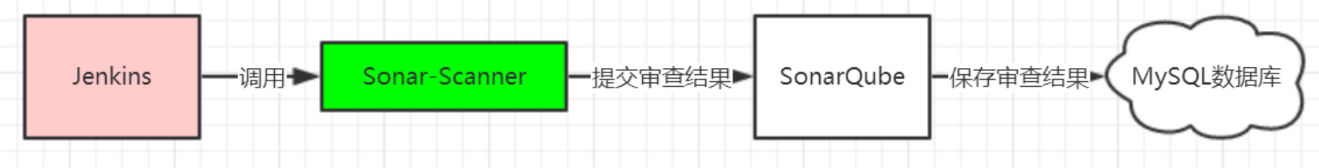


图46-sonarqube代码审查流程

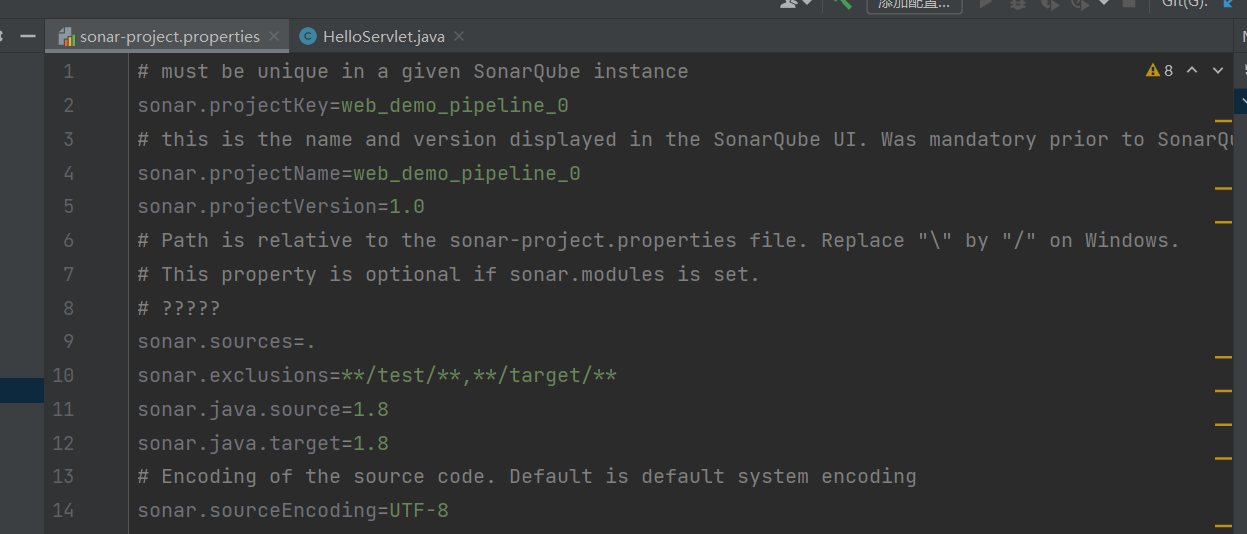


图47-代码审查脚本配置

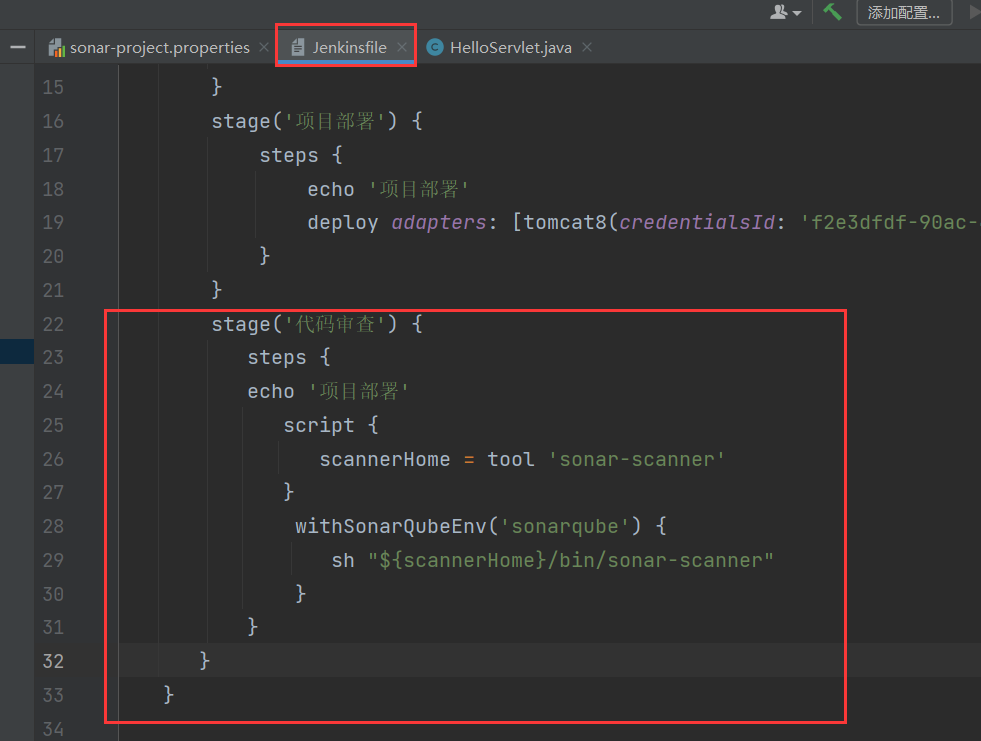
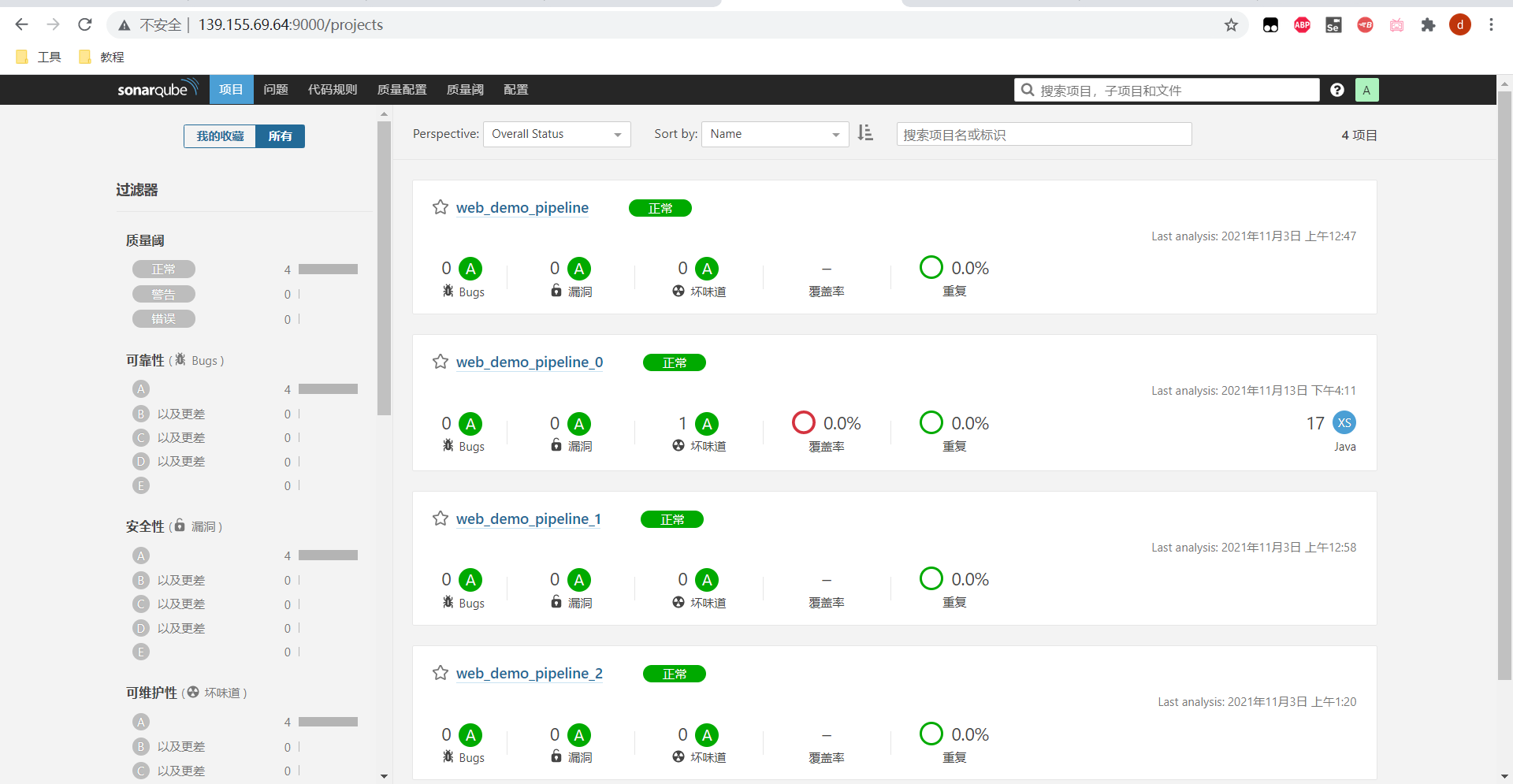


图48-代码审查脚本配置



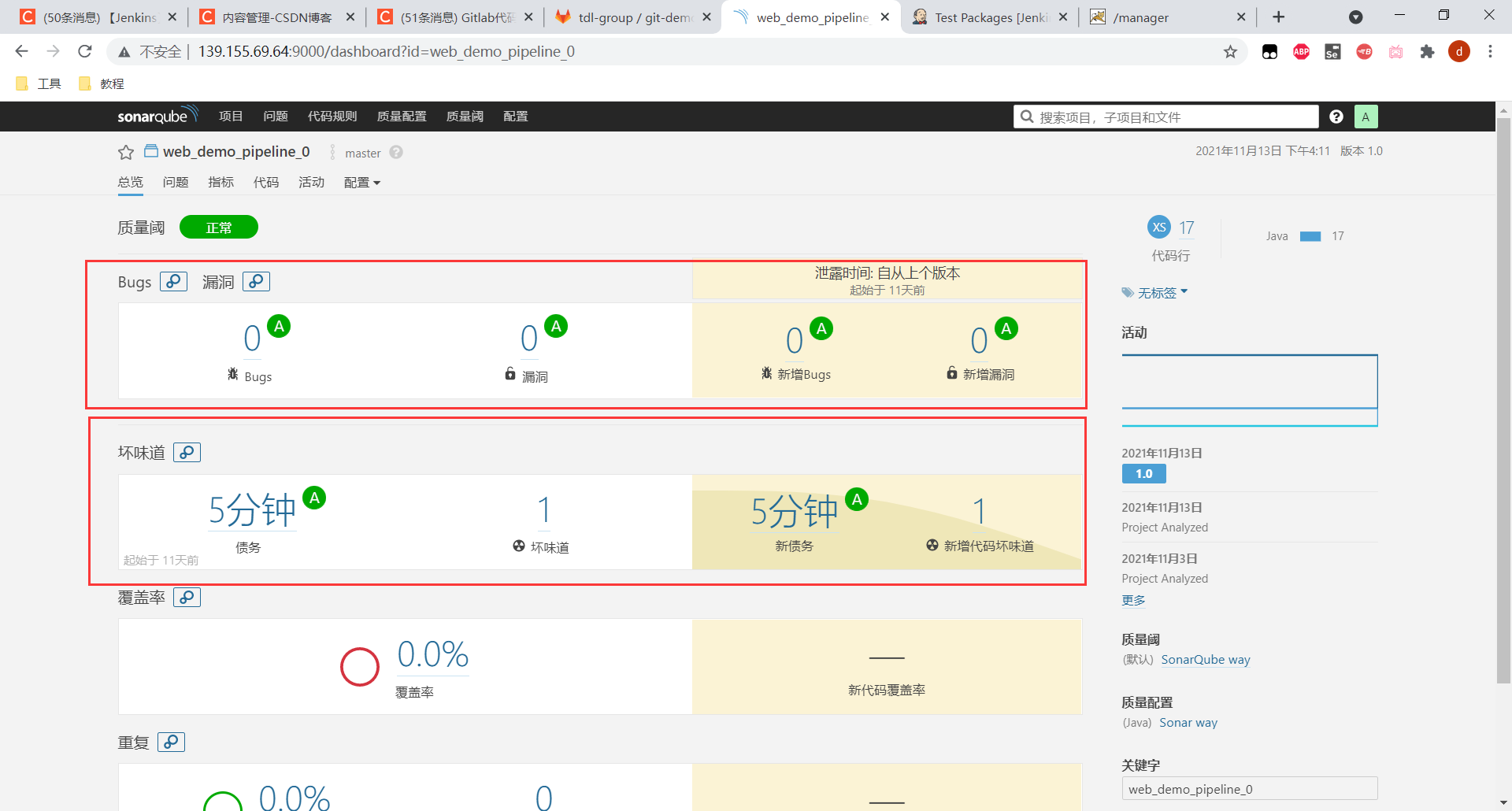


图49-50 sonarqube平台查看审查项目结果

# 附件

GitLab 代码仓库地址：<http://47.92.139.79:82/>

**账号：zhangsan**

**密码: zhangsan**

测试项目url:

[git@47.92.139.79:root/wuhan\_system.git](mailto:git@47.92.139.79:root/wuhan_system.git)

<http://47.92.139.79:82/root/wuhan_system.git>

Jenkins自动化测试框架地址： <http://139.155.69.64:8888/>

**账号：tdl**

**密码: 123**

Tomcat测试环境地址：<http://112.124.33.37:8080/manager/html>

**账号：tomcat**

**密码：tomcat**

Sonarqube代码审查平台地址：<http://139.155.69.64:9000/>

**账号：admin**

**密码：admin**