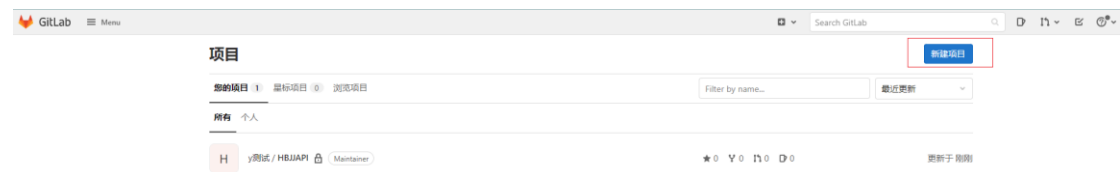


# 一、版本控制

1、首先，将 pytest 自动化测试脚本纳入版本控制系统中，也就是常见的 git 中，一般公司内部都有自己的 git 系统，如果没有则可以传到码云上（需要注册 Gitee 账号）

2、拿我的举例，我将代码上传到公司内部的 gitlab 代码仓库系统，打开 gitlab 系统，如下步骤所示：

## 1) 新建一个项目



## 2) 创建空白项目



## 3) 添加项目名称、设置项目可见性级别



## 创建空白项目

创建一个空白项目来存放您的文件，规划您的工作，并在代码等方面进行协作。

新建项目 · 创建空白项目

### 项目名称

apiautotest

1

### 项目 URL

http://192.168.105.36:8100/ yzmttest

### 项目标识

apiautotest

想要在同一命名空间下存放几个依赖项目？请[创建一个群组](#)。

### 项目描述(可选)

测试

### 可见性级别

2

☒ 私有

项目访问必须明确授予每个用户。如果此项目是在一个群组中，群组成员将会获得访问权限。

☐ 内部

除外部用户外，任何登录用户均可访问该项目。

☐ 公开

该项目允许任何人访问。

☐ 使用自述文件初始化仓库

允许您立即克隆这个项目的仓库。如果您计划推送一个现有的仓库，请跳过这个步骤。

3

新建项目

取消

## 项目

新建项目

您的项目 2

星标项目 0

浏览项目

Filter by name...

最近更新

所有 个人

A

y测试 / apiautotest 私有 Maintainer

测试

★ 0 ♿ 0 🗒 0 📄 0

更新于 1分钟前

H

y测试 / HBIIAPI 私有 Maintainer

★ 0 ♿ 0 🗒 0 📄 0

更新于 17分钟前

## 4) 点击项目，进入项目详细页面

y测试 > apiautotest

A

apiautotest 私有

项目 ID: 35

🔗

🔔

★ 星标

0

测试

### Invite your team

Add members to this project and start collaborating with your team.

Invite members

### 该项目仓库当前为空

您可以通过克隆仓库开始或开始以以下方式之一添加文件。

克隆

新建文件

添加自述文件

添加LICENSE

添加更新日志

添加贡献信息

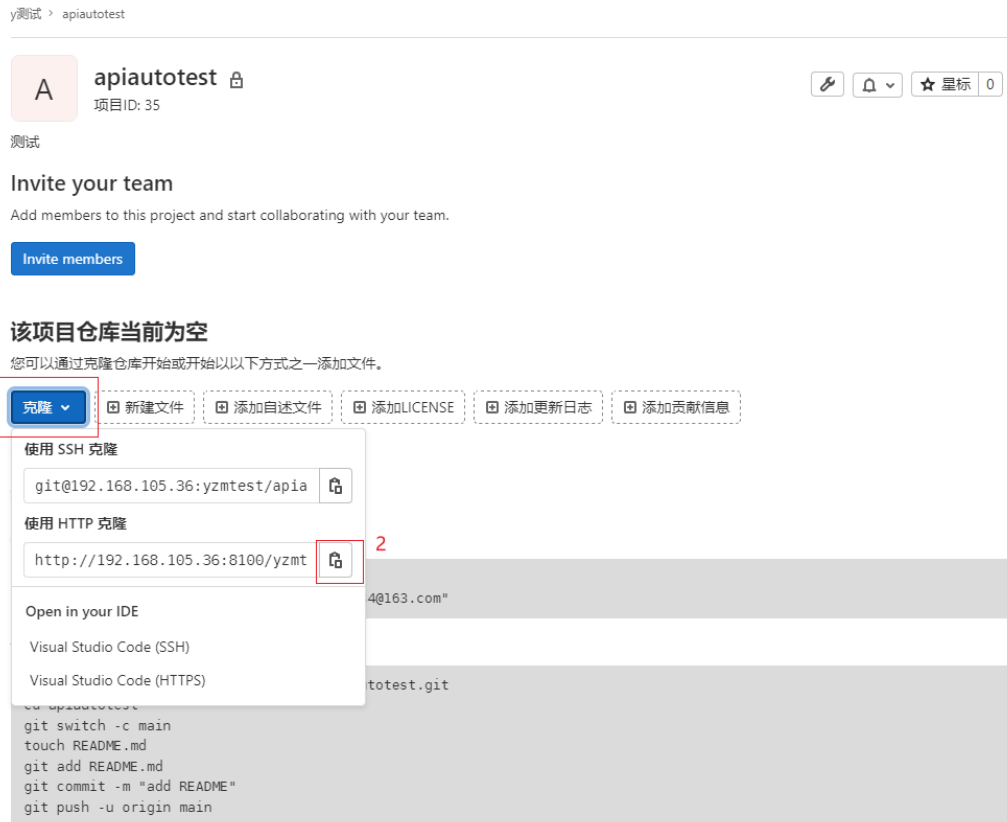
### 命令行指引

您还可以按照以下说明从计算机中上传现有文件。

#### Git 全局设置

```
git config --global user.name "y测试"
git config --global user.email "15810102634@163.com"
```

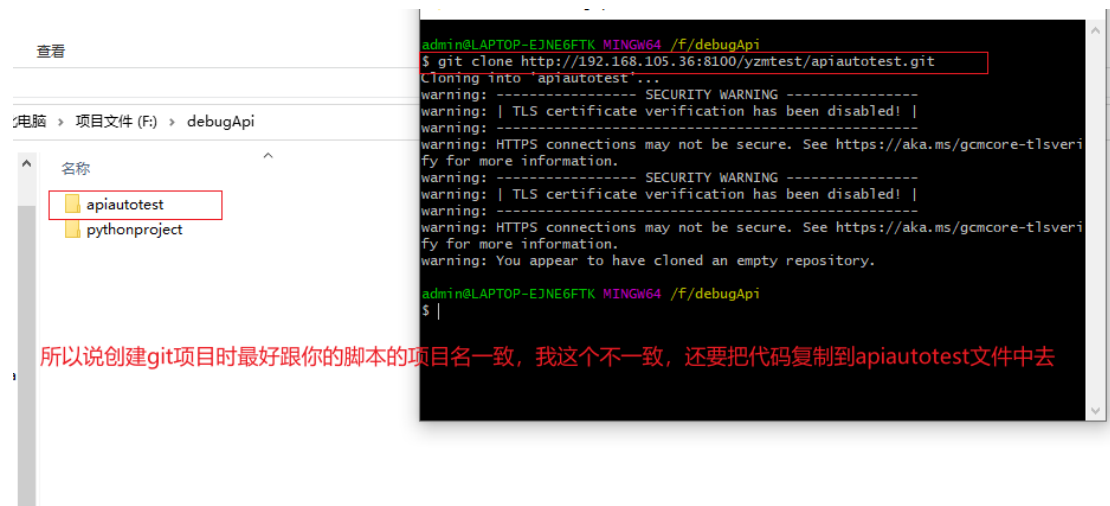
## 5) 复制克隆地址



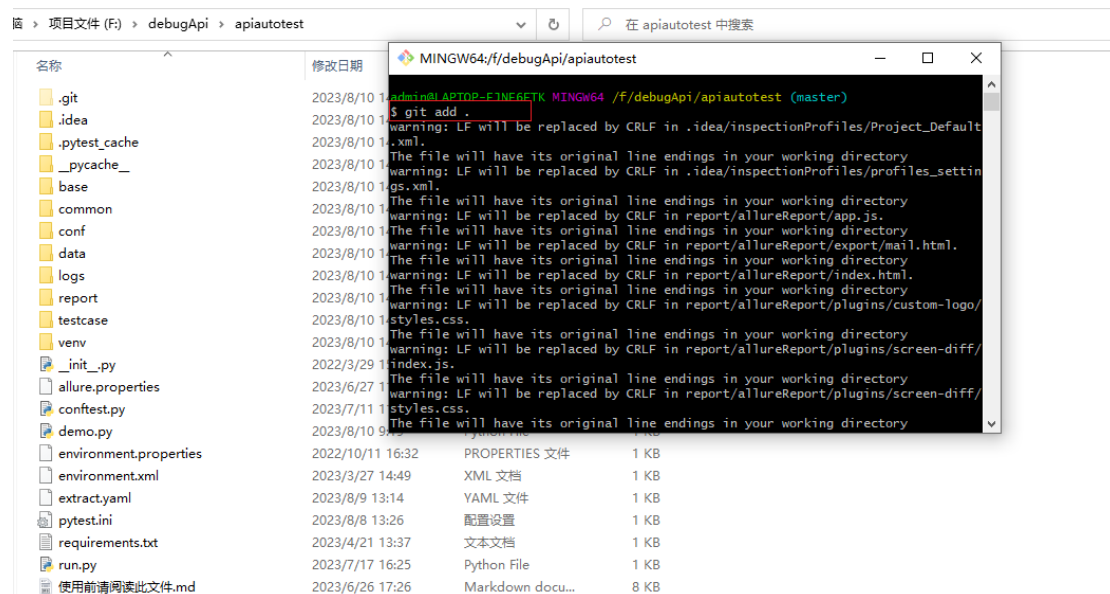
6) 在电脑盘中找一个盘根目录下，右键点击 git bash here 打开 git 窗口（前提是电脑已经安装 Git 软件）



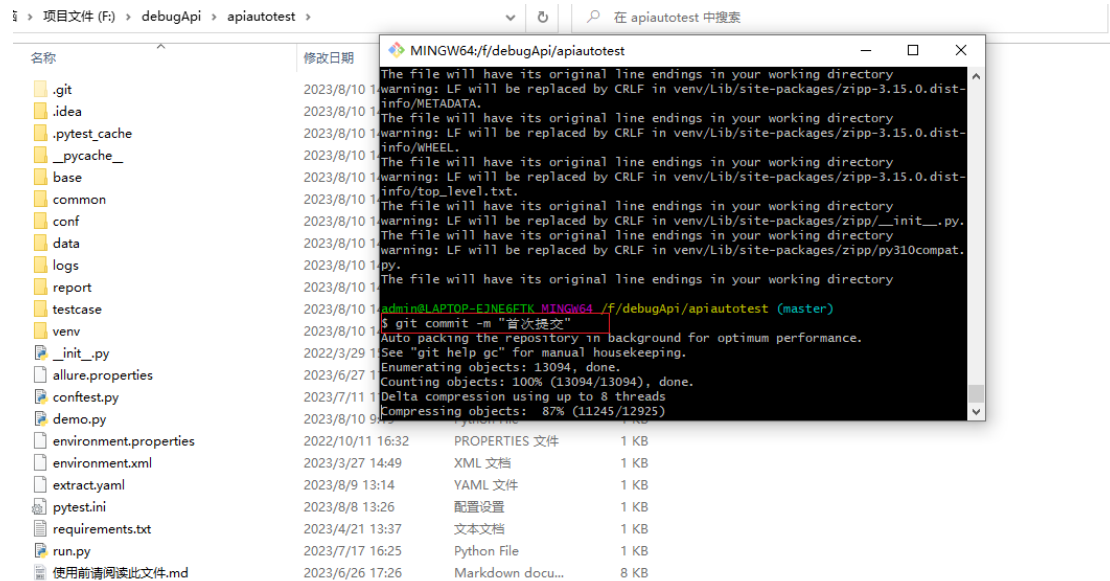
7) 执行第 5 步复制的克隆地址，执行：git clone 克隆地址



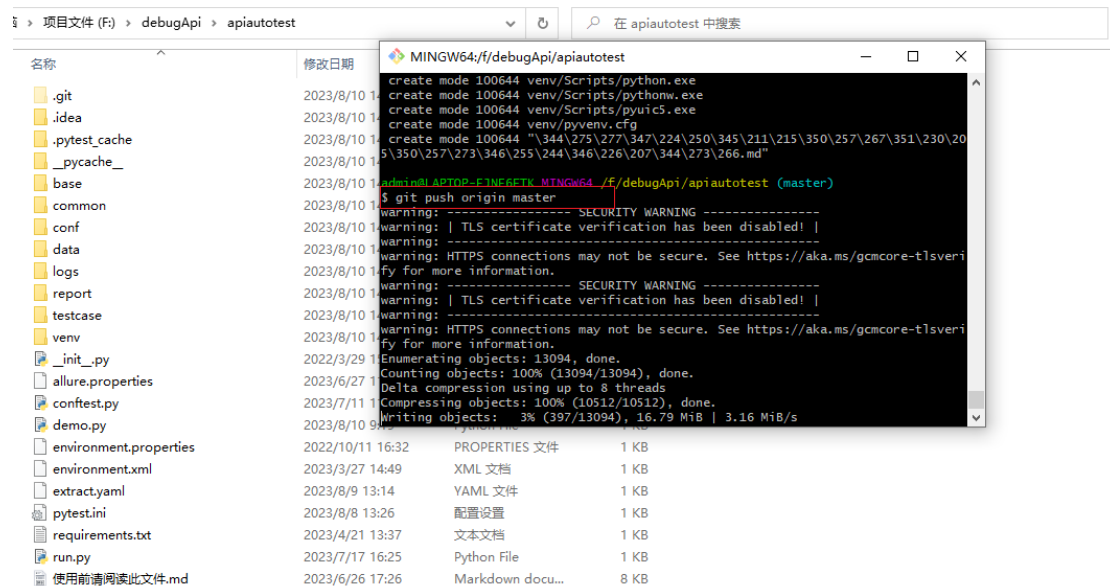
8) 将你本地的代码上传到 gitlab 仓库, 进入 apiautotest 目录并打开 git 窗口, 执行: git add . 将本地代码文件提交到暂存区。



9) 还是 git 窗口, 在执行: git commit -m "说明/描述", 将暂存区的文件提交到 master 分支上



10) 继续上一个窗口下，在执行：git push origin master，将分支上的数据推送到远程仓库中去。



11) 到这步，我们就已经把代码放到 git 仓库系统中去了，可以去 gitlab 下查看

A

**apiautotest**

星标 0

派生 0

项目ID: 35

1 次提交

1 个分支

0 个标签

104.6 MB 文件

104.6 MB 存储

测试

master

apiautotest /

+

历史

查找文件

Web IDE

克隆

首次提交

y测试 编辑于 6 minutes ago

89573c82

启用Auto DevOps

添加自述文件

添加LICENSE

添加更新日志

添加贡献信息

添加 Kubernetes 集群

| 名称          | 最后提交 | 最后更新          |
|-------------|------|---------------|
| .idea       | 首次提交 | 6 minutes ago |
| __pycache__ | 首次提交 | 6 minutes ago |
| base        | 首次提交 | 6 minutes ago |
| common      | 首次提交 | 6 minutes ago |
| conf        | 首次提交 | 6 minutes ago |
| data        | 首次提交 | 6 minutes ago |
| logs        | 首次提交 | 6 minutes ago |

到这，第一步的版本控制步骤已经完成。下面就需要去部署 jenkins 持续集成了

## 二、Jenkins 部署

### 1) 首先 jenkins 的插件是必备的，有些插件需要自己手动安装，位置：系统管理->插件管理

Q filter

可更新

可选插件

已安装

高级

安装

名称 ↓

版本

Released

已安装

☐

Allure

构建报告

This plugin integrates Allure reporting tool into Jenkins.

2.30.3

9 月 10 天 ago

2.29.0

☐

Ant

构建工具

Adds Apache Ant support to Jenkins

497.v94e7d9fffa\_b\_9

1 月 13 天 ago

1.11

警告：该插件只适用于Jenkins2.387.3或更新版本。它可能不能正常运行于你的Jenkins

Email Extension

构建工具

email

构建通知

用于发送邮件的插件

This plugin is a replacement for Jenkins's email publisher. It allows to configure every aspect of email notifications: when an email is sent, who should receive it and what the email says

☐

警告：该插件只适用于Jenkins2.387.3或更新版本。它可能不能正常运行于你的Jenkins

2.100

Applying this update will address security vulnerabilities in the currently installed version.

Warning: This plugin has dependencies on other plugins that require Jenkins 2.387.3 or newer. Jenkins will refuse to load the dependencies requiring a newer version of Jenkins, and in turn loading this plugin will fail.

Git

git 源码管理

This plugin integrates [Git](#) with Jenkins.

警告：该插件只适用于Jenkins2.387.3或更新版本。它可能不能正常运行于你的Jenkins

Applying this update will address security vulnerabilities in the currently installed version.

☐

Warning: This plugin requires newer versions of dependencies and at least one of those plugins is not compatible with the installed version. This is usually the case because its behavior changed, or it uses a different settings format than the installed version. Jobs using that plugin may need to be reconfigured, and/or you may not be able to cleanly revert to the prior version without manually restoring old settings. Consult the plugin release notes for details.

The following plugins are incompatible:

- [Credentials Binding \(1.27\)](#)
- [Git client \(3.9.0\)](#)
- [SSH Credentials \(1.19\)](#)

5.2.0

Warning: This plugin has dependencies on other plugins that require Jenkins 2.387.3 or newer. Jenkins will refuse to load the dependencies requiring a newer version of Jenkins, and in turn loading this plugin will fail.

Git client

api-plugin 插件库 (被其他插件使用)

Utility plugin for Git support in Jenkins

警告：该插件只适用于Jenkins2.387.3或更新版本。它可能不能正常运行于你的Jenkins

Applying this update will address security vulnerabilities in the currently installed version.

☐

Warning: This plugin requires newer versions of dependencies and at least one of those plugins is not compatible with the installed version. This is usually the case because its behavior changed, or it uses a different settings format than the installed version. Jobs using that plugin may need to be reconfigured, and/or you may not be able to cleanly revert to the prior version without manually restoring old settings. Consult the plugin release notes for details.

The following plugins are incompatible:

- [Mina SSHD API :: Core 0](#)
- [SSH Credentials \(1.19\)](#)
- [Trilead API \(1.0.13\)](#)

4.4.0

Warning: This plugin has dependencies on other plugins that require Jenkins 2.387.3 or newer. Jenkins will refuse to load the dependencies requiring a newer version of Jenkins, and in turn loading this plugin will fail.

JUnit

构建报告

Allows JUnit-format test results to be published.

警告：该插件只适用于Jenkins2.387.1或更新版本。它可能不能正常运行于你的Jenkins

Applying this update will address security vulnerabilities in the currently installed version.

☐

Warnings: This plugin has dependencies on other plugins that require Jenkins 2.387.3 or newer. Jenkins will refuse to load the dependencies requiring a newer version of Jenkins, and in turn loading this plugin will fail.

1217.v4297208a\_a\_b\_ce

等等。。。。。。需要用到的插件都需要安装


## 2) 新建任务，构建一个自由风格的软件项目


Dashboard > 所有 >


输入一个任务名称


aplautotest


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。

**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) 项目描述, 可填可不填

Dashboard > aplautotest >

General 源码管理 构建触发器 构建环境 构建 构建后操作

描述

测试描述

(纯文本) 预览

☐ 应用项目安全

☐ Change date pattern for the BUILD\_TIMESTAMP (build timestamp) variable

☐ GitHub 项目

☐ This build requires lockable resources

☐ Throttle builds

☐ 禁用旧的构建

☐ 参数化构建过程

☐ 关闭构建

☐ 在必要的时候并发构建

问题...

源码管理

☒ 无

☐ Git

构建触发器

☐ 触发远程构建 (例如使用脚本)

☐ 其他工程构建后触发

☐ 定时构建

保存 应用

Rolling

### 4) 配置源码管理, 将你 gitlab 上的项目地址复制到 Repository URL

再次进入到你的 gitlab 系统, 复制你的 gitlab 的项目地址



y测试 > apiautotest

A

apiautotest

项目ID: 35

1 次提交

1 个分支

0 个标签

104.7 MB 文件

104.7 MB 存储

测试

master

apiautotest /

历史

查找文件

Web IDE

克隆

首次提交

y测试 编辑于 3 hours ago

添加自述文件

添加LICENSE

添加更新日志

添加贡献信息

启用Auto

配置 CI/CD

| 名称          | 最后提交             |
|-------------|------------------|
| .idea       | 首次提交             |
| __pycache__ | 首次提交 3 hours ago |

使用 SSH 克隆

git@192.168.105.36:yzmtest/apia

使用 HTTP 克隆

http://192.168.105.36:8100/yzmt

Open in your IDE

Visual Studio Code (SSH)

Visual Studio Code (HTTPS)

复制

复制链接

源码管理

无

Git

Repositories

Repository URL

http://192.168.105.36:8100/yzmtest/apiautotest.git

无法连接仓库: Command "git ls-remote -h http://192.168.105.36:8100/yzmtest/apiautotest.git HEAD" returned status code 128:  
stdout:  
stderr: fatal: Authentication failed for 'http://192.168.105.36:8100/yzmtest/apiautotest.git/'

Credentials

- 无 -

添加

Jenkins

高级...

Add Repository

上面报错，显示无法连接到仓库，认证错误，需要设置你的 gitlab 系统的登录用户和密码

Jenkins 凭据提供者: Jenkins

添加凭据

Domain

全局凭据 (unrestricted)

类型

Username with password

范围

全局 (Jenkins, nodes, items, all child items, etc)

用户名

yzmtest

☐ Treat username as secret

密码

\*\*\*\*\*

ID

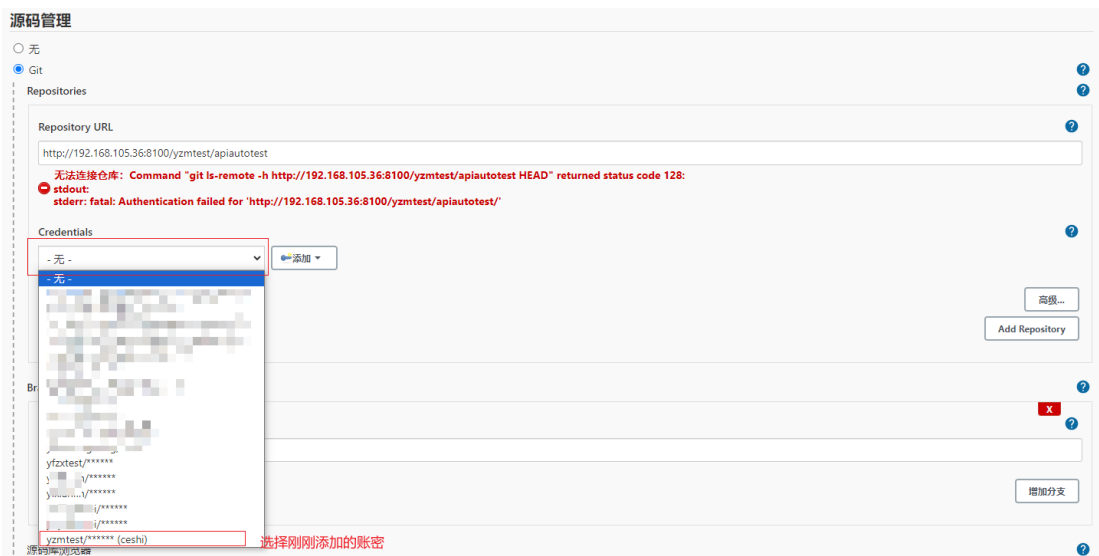
描述

ceshi

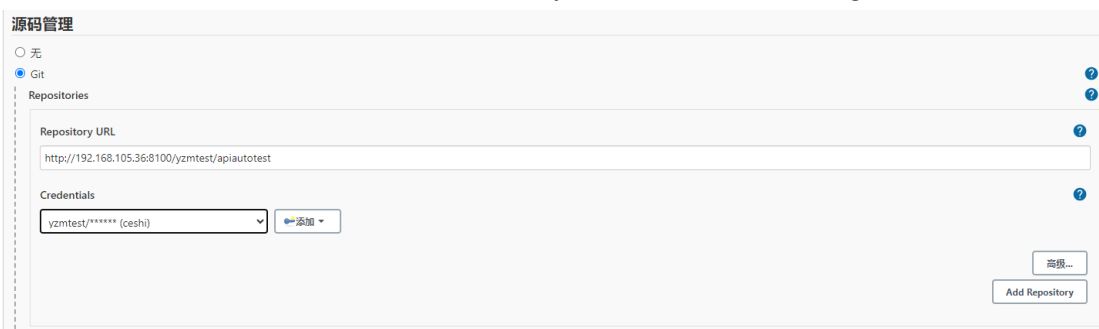
添加

取消

源码管理，选择刚刚添加的账密



选择账密后（如下图所示），这里不报错说明jenkins 已经能够访问到 gitlab 的源码了。



源码库浏览器配置，如下



## 5) 构建触发器，如设置定时构建

**构建触发器**

- ☐ 触发远程构建 (例如,使用脚本)
- ☐ 其他工程构建后触发
- ☒ 定时构建

日程表

H/5 \* \* \* \*

上次运行的时间 Thursday, August 10, 2023 at 8:45:07 AM Coordinated Universal Time; 下次运行的时间 Thursday, August 10, 2023 at 8:45:07 AM Coordinated Universal Time.

☐ GitHub hook trigger for GITScm polling

☐ 轮询 SCM

上图中表示的是每 5 分钟构建一次

具体构建时间的意思设置及含义可参考: <https://app.yinxiang.com/fx/c26b7017-c493-4a39-91d7-d04fa47ca04e>

## 6) 构建环境配置

**构建环境**

☒ Delete workspace before build starts

☐ Use secret text(s) or file(s)

☐ Abort the build if it's stuck

☒ Add timestamps to the Console Output

☐ Inspect build log for published Gradle build scans

☐ With Ant

勾选这两个即可

## 7) 构建配置，增加构建步骤—执行 shell

**构建**

增加构建步骤

- Invoke Ant
- Invoke Gradle script
- Run with timeout
- Set build status to "pending" on GitHub commit
- 执行 shell
- 调用顶层 Maven 目标

**构建**

执行 shell

命令

```
#!/bin/bash
python3 -V
pip3 -V
cd pythonProject
python3 run.py
```

查看 可用的环境变量列表

高级...

输入如下命令：

```
#!/bin/bash
python3 -V
pip3 -V
pip3 install -r requirements.txt
cd pythonProject
python3 run.py
```

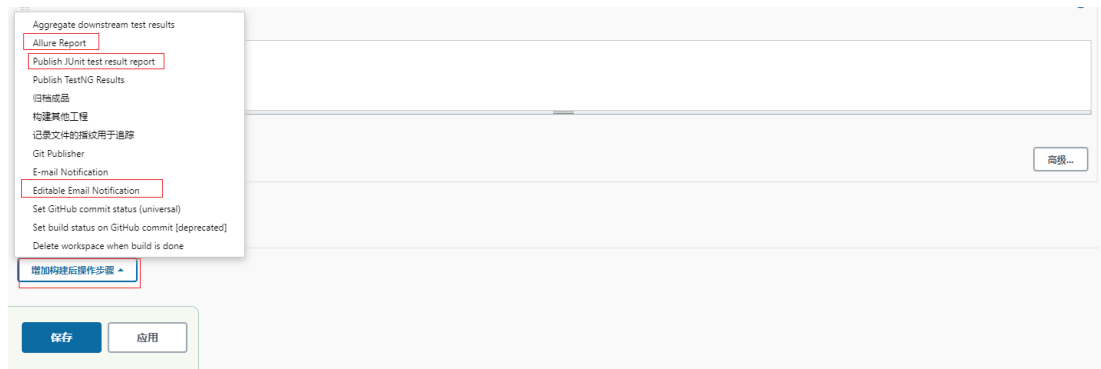
### 解释：

- #!/bin/bash 表示 shell 脚本解释并执行
- Python3 -V 表示打印 python 的版本号
- Pip3 -V 表示打印 pip 版本号
- pip3 install -r requirements.txt 表示安装项目的依赖模块

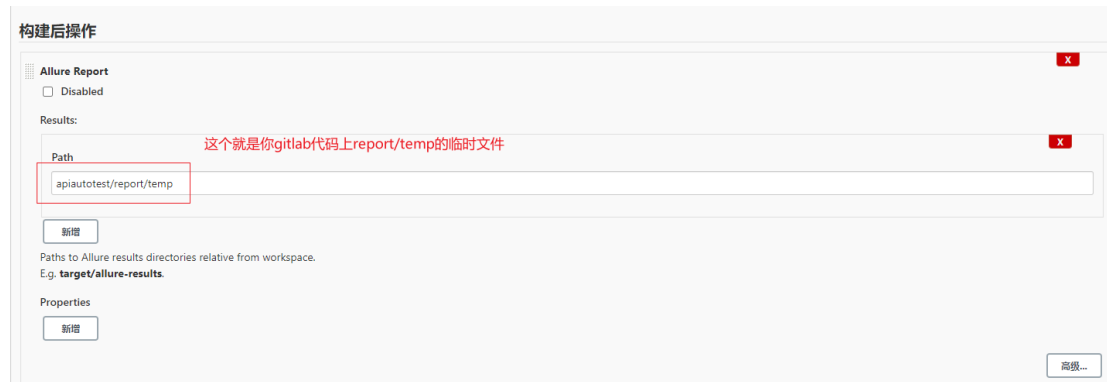
cd apiautotest      表示进入到你的项目路径下（这里填项目根目录名即可）

python3 run.py      表示执行你的自动化脚本中的主函数

## 8) 构建后操作配置，增加 allure report 报告配置



### Allure report 配置



### Publish JUnit test result report 配置



上面这个 results.xml 文件需要在脚本中设置，如下

```
import shutil

import pytest
import os
from conf import setting
from common.dingding_robot import send_dd_msg

if __name__ == '__main__':
    pytest.main(
        ['-s', '-v', '--alluredir=./report/temp', './testcase', '--clean-alluredir', '--junitxml=./report/results.xml'])

    shutil.copy('./environment.properties', './report/temp')
    send_dd_msg()
```

这里生成的路径一定要跟上面配置的那个路径一致。

Editable Email Notification 配置，这个是jenkins在测试完成之后发自动给你发邮件。

Editable Email Notification

☐ Disable Extended Email Publisher

Allows the user to disable the publisher, while maintaining the settings

Project From

Project Recipient List ?

项目收件人列表，也就是说可以加上项目组成员的邮箱

1@163.com

Comma-separated list of email address that should receive notifications for this project.

Project Reply-To List ?

这个是回复列表，当人回复邮件时哪些人接收

1@163.com

Comma-separated list of email address that should be in the Reply-To header for this project.

Content Type ?

Default Content Type

Default Subject ?

\$DEFAULT\_SUBJECT

Default Content ?

\$DEFAULT\_CONTENT

Attachments ?

Can use wildcards like 'module/dist/\*\*/\*.zip'. See the [@includes of Ant fileset](#) for the exact format. The base directory is [the workspace](#).

Attach Build Log ?

Do Not Attach Build Log

Content Token Reference ?

Advanced Settings...

增加构建后操作步骤 ▾

保存

应用

到这一步已经把项目的主要配置已经设置好了，但是你发送邮箱需要在全局中设置一下

9) Email Extension Plugin, 这个插件, 如果 jenkins 没有的话需要安装, 安装完成之后去设置这个插件的内容, 位置: 系统管理->系统配置->往下翻找到这个插件

### Extended E-mail Notification

**SMTP server** 这里配置你需要使用哪个邮件发送  
smtp.163.com

**SMTP Port**  
25 邮箱端口

**Default user e-mail suffix**  
@163.com 默认使用163邮箱

**Default Content Type**  
HTML (text/html)

**List ID**

☐ Add 'Precedence: bulk' E-mail Header

**Default Recipients**

**Reply To List**

**Default Subject** 邮件主题  
\$PROJECT\_NAME - Build # \$BUILD\_NUMBER - \$BUILD\_STATUS!

**Maximum Attachment Size**  
-1

**Default Content**  

```
<!DOCTYPE html>
<html>

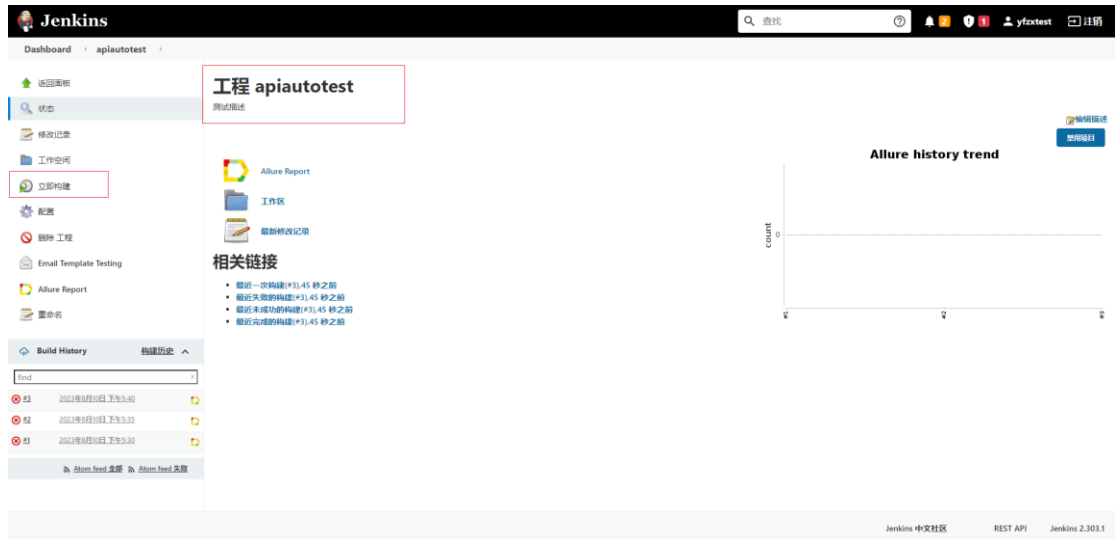
<head>
  <meta charset="UTF-8">
  <title>${ENV, var="JOB_NAME"}-第${BUILD_NUMBER}次构建日志</title>
</head>

<body leftmargin="8" marginwidth="0" topmargin="8" marginheight="4" offset="0">
  <div width="95%" cellpadding="0" cellspacing="0" style="font-size: 11pt; font-family: Tahoma, Arial, Helvetica, sans-serif">
    <div>本邮件由系统自动发出，无需回复！
    <br>
    <br>各位同事，大家好，以下为<b>${PROJECT_NAME}</b><b>自动化测试项目构建信息</b>
    <br>
    <div><h4><font color="#CC0000">构建结果 - ${BUILD_STATUS}</font></h4></div>
    </div>
    <div>
      <h4><font color="#006610B">构建信息</font></h4>
      <hr size="2" width="100%" />
      <ul>
        <li>项目名称： ${PROJECT_NAME}</li>
        <li>构建编号： 第${BUILD_NUMBER}次构建</li>
        <li>触发原因： ${CAUSE}</li>
        <li>构建状态： ${BUILD_STATUS}</li>
        <li>构建日志： <a href="${BUILD_URL}console">${BUILD_URL}console</a>
        </li>
        <li>构建地址： <a href="${BUILD_URL}">${BUILD_URL}</a>
        </li>
      </ul>
    </div>
    <div>
      <h4><font color="#006610B">测试结果</font></h4>
    </div>
  </div>
</body>
</html>
```

**保存** **应用**

邮件内容的设置内容可参考：<https://app.yinxiang.com/fx/b79b45ef-96cf-47fe-8501-9d58d1cfc480>

到这，全部的jenkins项目配置基本完成，可以点击立即构建测试一下



可以看到这个已经根据我们刚刚设置的定时任务了，每 5 分钟构建一次，但是这里构建失败，需要查找下原因。

**错误 1：**

## 控制台输出

```
18:00:00 Started by timer
18:00:00 Running as SYSTEM
18:00:00 Building in workspace /var/lib/jenkins/workspace/apiautotest
18:00:00 [WS-CLEANUP] Deleting project workspace...
18:00:00 [WS-CLEANUP] Deferred wipeout is used...
18:00:00 [WS-CLEANUP] Done
18:00:00 The recommended git tool is: NONE
18:00:00 using credential cb6fba40-6d24-433e-b0d6-c7a76184a092
18:00:00 Cloning the remote Git repository
18:00:00 Cloning repository 'https://gitlab.com/xxxxxx/apiautotest'
18:00:00 > git init /var/lib/jenkins/workspace/apiautotest # timeout=10
18:00:00 Fetching upstream changes from 'https://gitlab.com/xxxxxx/apiautotest'
18:00:00 > git --version # timeout=10
18:00:00 > git --version # 'git version 1.8.3.1'
18:00:00 using GIT_ASKPASS to set credentials ceshi
18:00:00 > git fetch --tags --progress https://gitlab.com/xxxxxx/apiautotest +refs/heads/*:refs/remotes/origin/* # timeout=10
18:00:01 ERROR: Error cloning remote repo 'origin'
18:00:01 hudson.plugins.git.GitException: Command "git fetch --tags --progress https://gitlab.com/xxxxxx/apiautotest +refs/heads/*:refs/remotes/origin/*" returned status code 128:
18:00:01 stdout:
18:00:01 stderr: error: RPC failed: result=22, HTTP code = 422
18:00:01 fatal: The remote end hung up unexpectedly
18:00:01
18:00:01 at org.jenkinsci.plugins.gitclient.CliGitAPIImpl.launchCommandIn(CliGitAPIImpl.java:2681)
18:00:01 at org.jenkinsci.plugins.gitclient.CliGitAPIImpl.launchCommandWithCredentials(CliGitAPIImpl.java:2102)
18:00:01 at org.jenkinsci.plugins.gitclient.CliGitAPIImpl.access$500(CliGitAPIImpl.java:86)
18:00:01 at org.jenkinsci.plugins.gitclient.CliGitAPIImpl$1.execute(CliGitAPIImpl.java:624)
18:00:01 at org.jenkinsci.plugins.gitclient.CliGitAPIImpl$2.execute(CliGitAPIImpl.java:853)
18:00:01 at hudson.plugins.git.GitSCM.retrieveChanges(GitSCM.java:1227)
18:00:01 at hudson.plugins.git.GitSCM.checkout(GitSCM.java:1305)
18:00:01 at hudson.scm.SCM.checkout(SCM.java:505)
```

原因：源码管理的 git url 地址不正确

解决办法：复制 gitlab 上项目的 HTTP 克隆地址即可

## 错误 2:

```
18:08:02 Avoid second fetch
18:08:02 > git rev-parse refs/remotes/origin/master {commit} # timeout=10
18:08:02 Checking out Revision 89573c825e8175db3166614a5f90c00c3c7b88cd (refs/remotes/origin/master)
18:08:02 > git config core.sparsecheckout # timeout=10
18:08:02 > git checkout -f 89573c825e8175db3166614a5f90c00c3c7b88cd # timeout=10
18:08:04 Commit message: "首次提交"
18:08:04 > git rev-list --no-walk 89573c825e8175db3166614a5f90c00c3c7b88cd # timeout=10
18:08:04 Checking for pre-build
18:08:04 Executing pre-build step
18:08:04 Checking if email needs to be generated
18:08:04 No emails were triggered.
18:08:04 [apiautotest] $ /bin/bash /tmp/jenkins3665520232591984467.sh
18:08:04 Python 3.9.6
18:08:05 pip 21.3.1 from /var/lib/jenkins/.local/lib/python3.9/site-packages/pip (python 3.9)
18:08:05 /tmp/jenkins3665520232591984467.sh: line 4: cd: apiautotest: No such file or directory
18:08:05 ImportError while loading conftest '/var/lib/jenkins/workspace/apiautotest/conftest.py'.
18:08:05 conftest.py:6: in <module>
18:08:05     from common.readyaml import ReadYamlData
18:08:05 common/readyaml.py:4: in <module>
18:08:05     import chardet
18:08:05 E ModuleNotFoundError: No module named 'chardet'
18:08:05 sh: allure: command not found
18:08:05 [apiautotest] $ /var/lib/jenkins/allure-2.13.9/bin/allure generate -c -o /var/lib/jenkins/workspace/apiautotest/allure-report
18:08:06 allure-results does not exist
18:08:07 Report successfully generated to /var/lib/jenkins/workspace/apiautotest/allure-report
18:08:07 Allure report was successfully generated.
18:08:07 Creating artifact for the build.
18:08:07 Artifact was added to the build.
18:08:07 Recording test results
18:08:07 ERROR: Step 'Publish JUnit test result report' failed: No test report files were found. Configuration error?
18:08:07 Checking for post-build
18:08:07 Performing post-build step
18:08:07 Checking if email needs to be generated
18:08:07 Email was triggered for: Failure - Any
18:08:07 Sending email for trigger: Failure - Any
```

原因：jenkins 上的模块没有安装 chardet 模块，如果报的是这种错误，那就说明你的环境基本没啥大问题了，只要把这些模块安装好就可以。