

# UI自动化测试-Git安装和使用

## 下载

Git下载地址请访问[这里](#),根据对应的操作系统下载并安装即可

The screenshot shows the Git website homepage. At the top, the Git logo and tagline "--fast-version-control" are visible. A search bar is on the right. The left sidebar contains links for "About", "Documentation", "Downloads", "GUI Clients", "Logos", and "Community". The main content area is titled "Downloads" and features a large monitor graphic displaying the latest source release "2.37.1" with a "Download for Windows" button. Below this, there are sections for "GUI Clients" (listing git-gui, gitk, and third-party tools) and "Logos" (listing various formats). A "Git via Git" section provides a terminal command to clone the repository. The footer includes a copyright notice and a link to the Software Freedom Conservancy.

**git** --fast-version-control

Search entire site...

**About**

**Documentation**

**Downloads**

GUI Clients

Logos

**Community**

The entire **Pro Git book** written by Scott Chacon and Ben Straub is available to [read online for free](#). Dead tree versions are available on [Amazon.com](#).

## Downloads

Older releases are available and the Git source repository is on GitHub.

**macOS** **Windows** **Linux/Unix**

Latest source Release  
**2.37.1**  
Release Notes (2022-07-04)  
[Download for Windows](#)

### GUI Clients

Git comes with built-in GUI tools (**git-gui**, **gitk**), but there are several third-party tools for users looking for a platform-specific experience.

[View GUI Clients →](#)

### Logos

Various Git logos in PNG (bitmap) and EPS (vector) formats are available for use in online and print projects.

[View Logos →](#)

### Git via Git

If you already have Git installed, you can get the latest development version via Git itself:

```
git clone https://github.com/git/git
```

You can also always browse the current contents of the git repository using the [web interface](#).

</> About this site  
Patches, suggestions, and comments are welcome.

Git is a member of Software Freedom Conservancy

## 配置

git账号权限申请去找 @利世(zhao,赵学军) 处理

安装完成后, 打开安装好的Git Bash 请参考下方配置操作即可



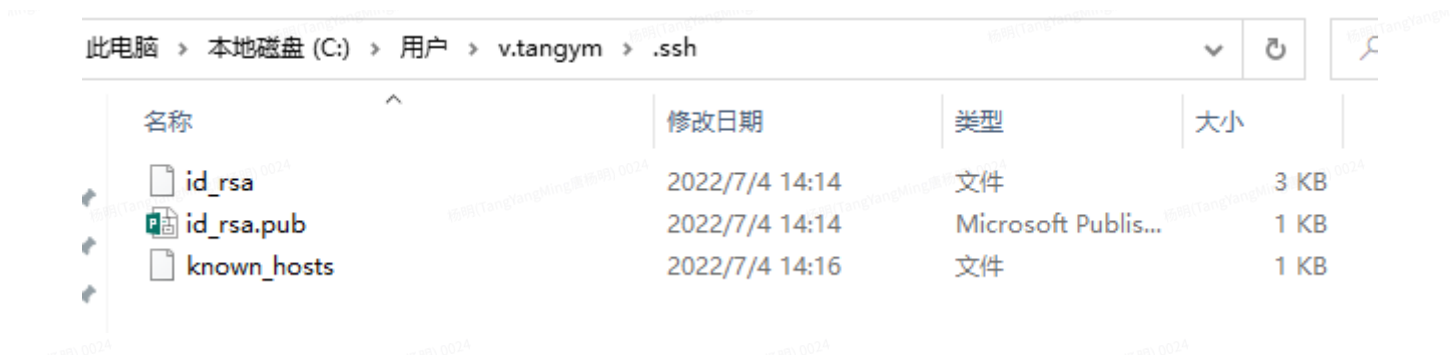
## 1) 设置用户名和邮箱

- 1 `git config --global user.name "v.tangym"` 填写自己的git用户名
- 2 `git config --global user.email "v.tangym@yoozoo.com"` 填写自己的git邮箱账号

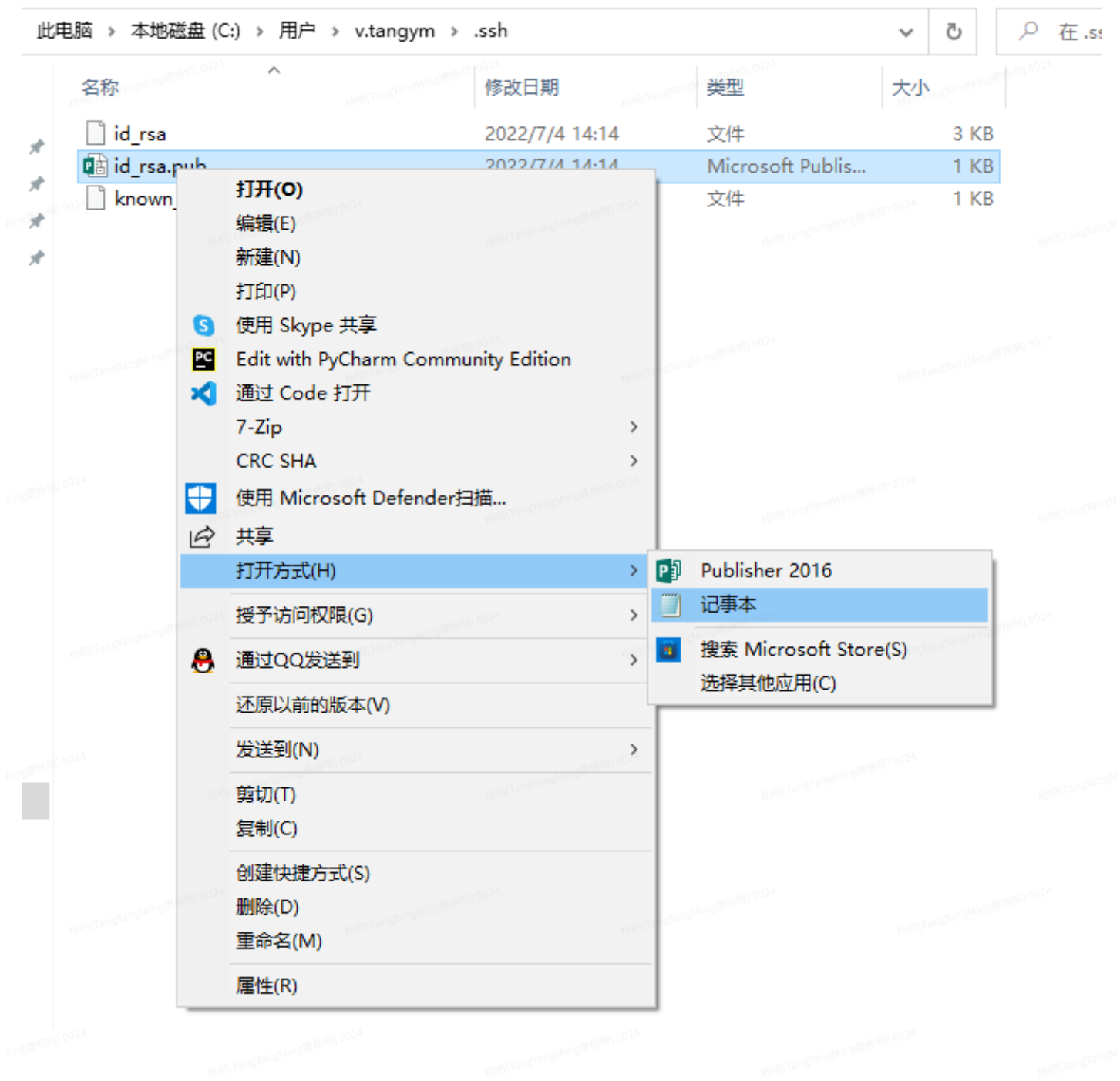
## 2) 生成公钥

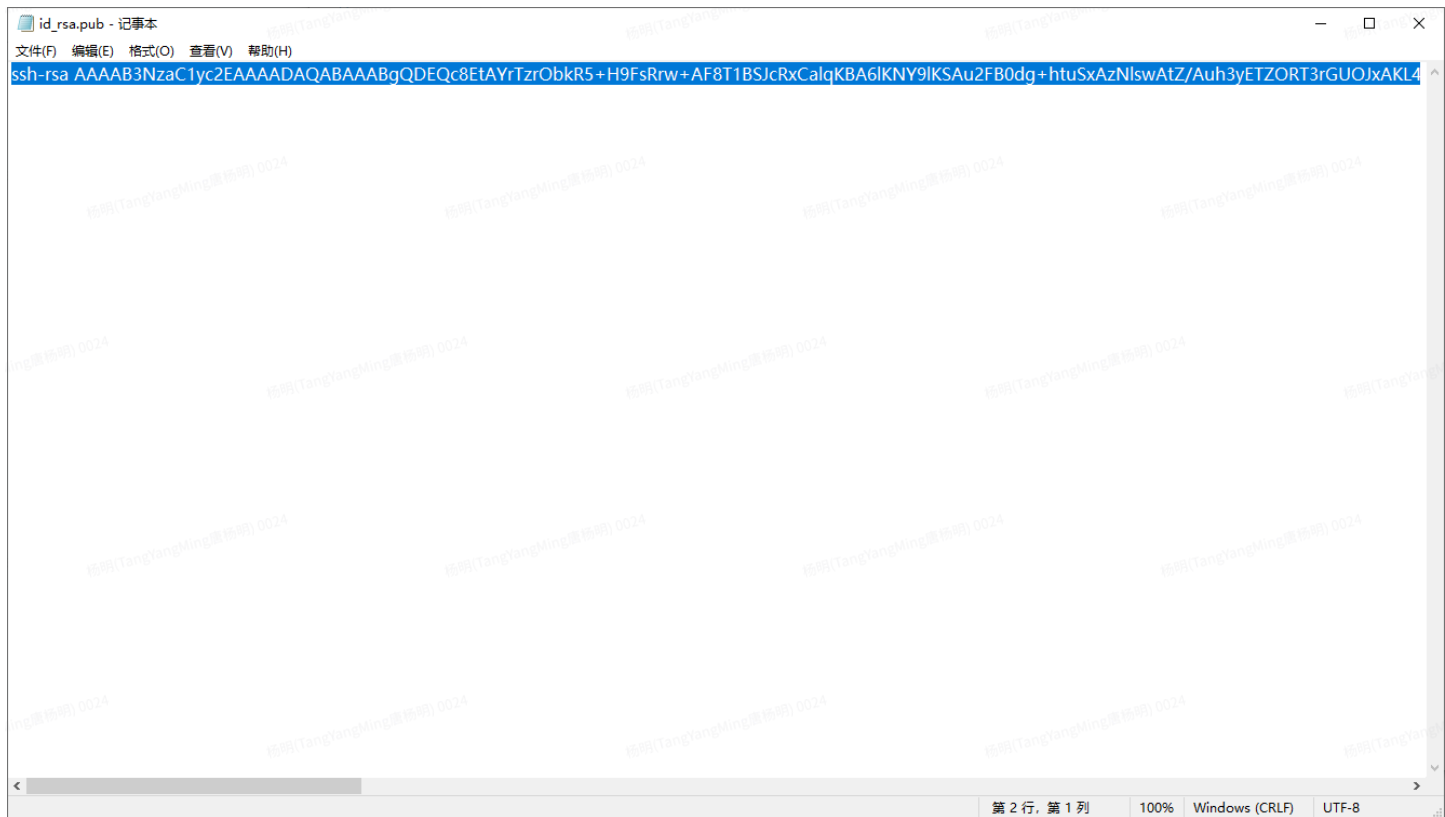
- 1 `ssh-keygen -t rsa -C "v.tangym@yoozoo.com"` 填写自己的git邮箱账号 需要敲击回车三次

生成公钥时，若指定输出位置，输入值为 文件前中缀

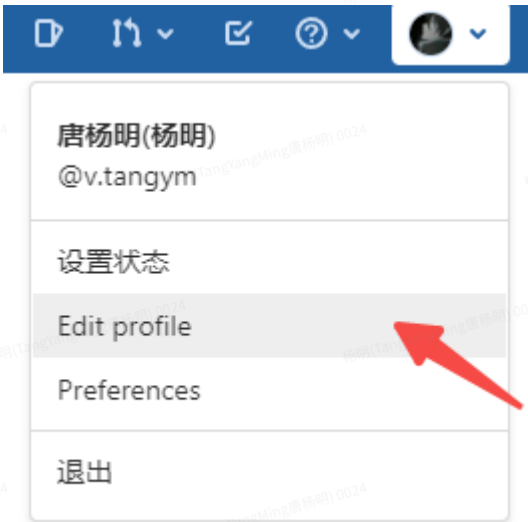


以文本方式打开.pub文件，复制内容





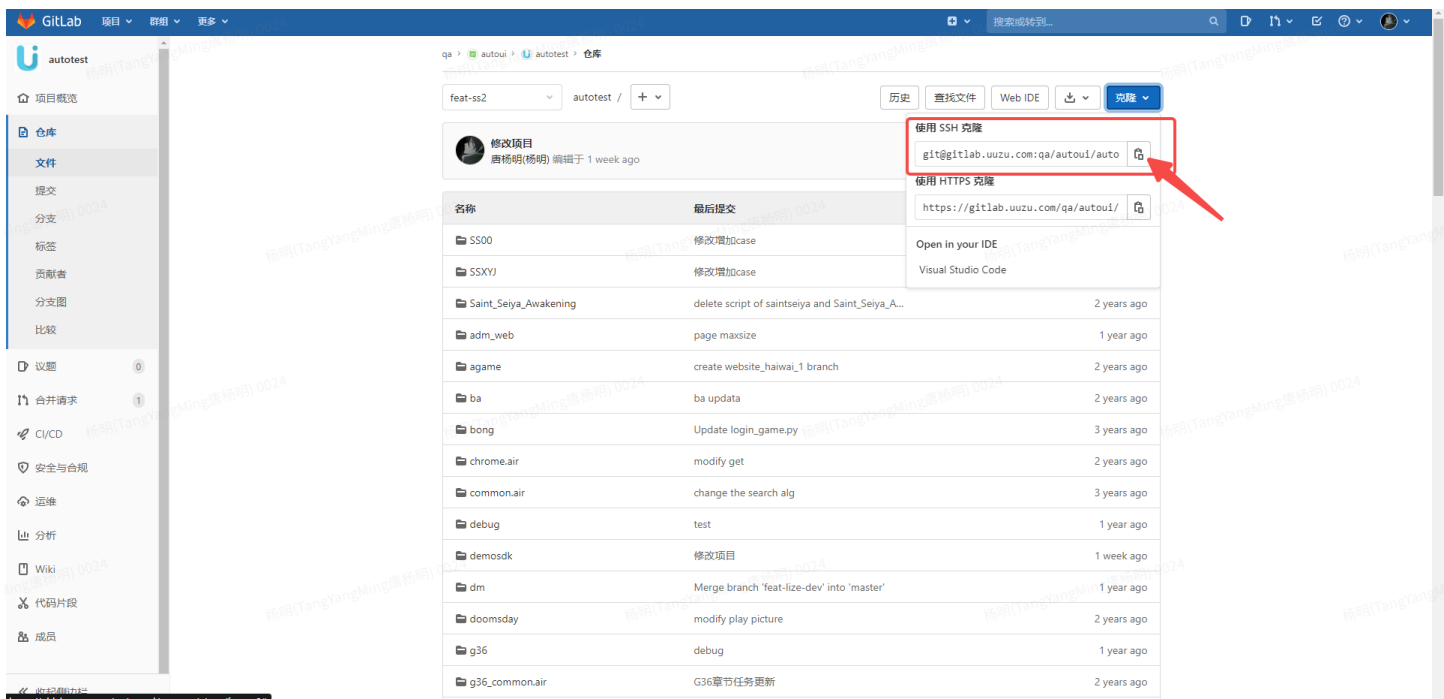
打开[Git](#)网页，进入设置



填写并应用



## 克隆库



## git命令

- 1 Job为指定目录，不指定克隆到当前工作目录
- 2 `git clone git@gitlab.uuzu.com:qa/autoui/autotest.git Job`
- 3
- 4 切换到分支 feat-ss2
- 5 `git checkout feat-ss2`
- 6
- 7

# 代码提交

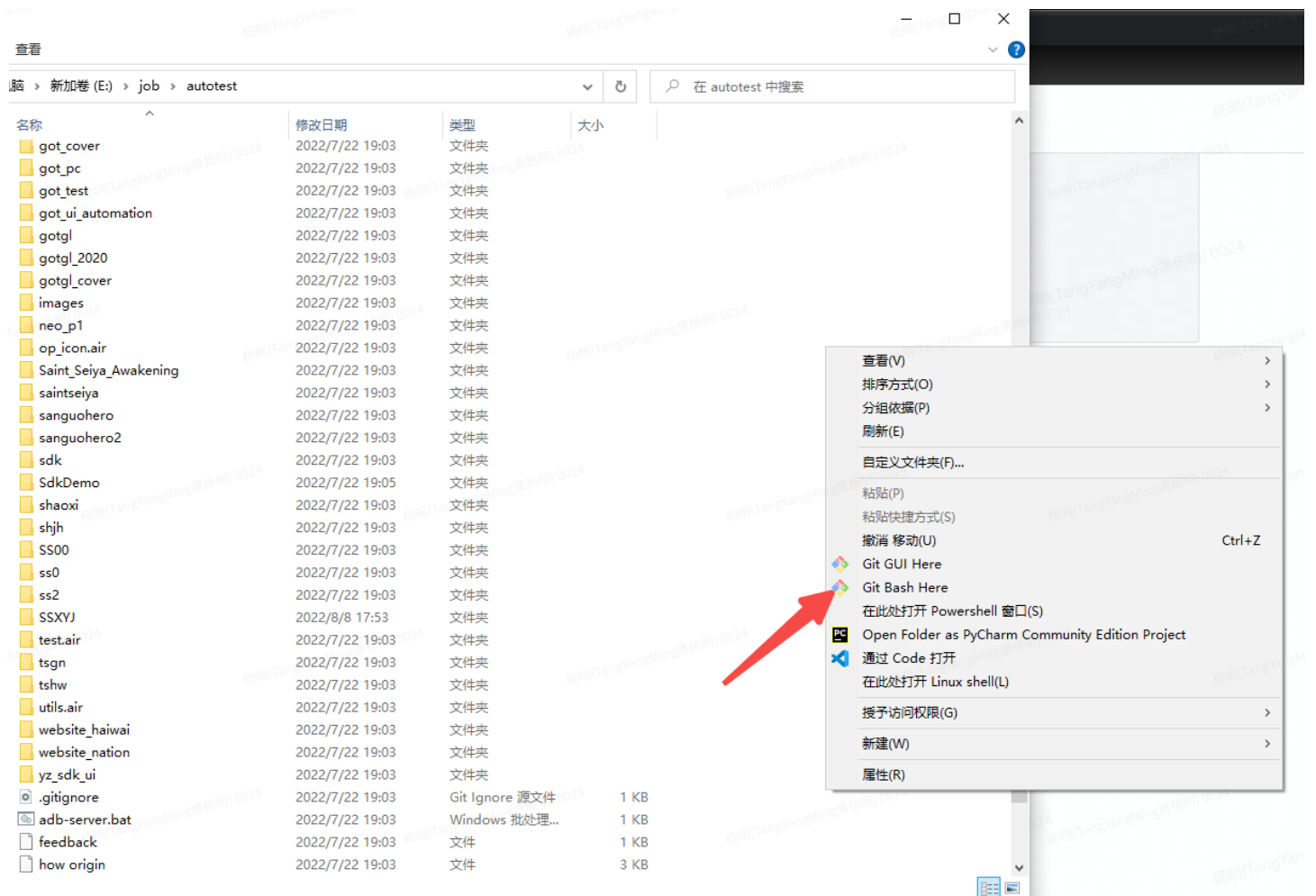
## Git命令

- 1 添加文件
- 2 `git add .` （将当前目录的所有文件到暂存区）
- 3
- 4 提交注释
- 5 `git commit -m [提交注释]` （提交暂存区到仓库区，将索引的当前内容与描述更改的用户和日志消息一起存储在新的提交中）
- 6
- 7 推送到远程
- 8 `git push`
- 9
- 10 推送到远程feat-ss2分支上面
- 11 `git push origin feat-ss2`
- 12
- 13 下拉代码
- 14 `git pull`
- 15

## 总结：

### 第一种方法：

第一步：在修改代码的文件夹目录下右键点击Git Bash Here



第二步：git add . (将当前目录的所有文件到暂存区)

```
MINGW64:/e/job/job/autotest
v.tangym@UPC4615 MINGW64 /e/job/job/autotest (feat-ss2)
$ git add .
```

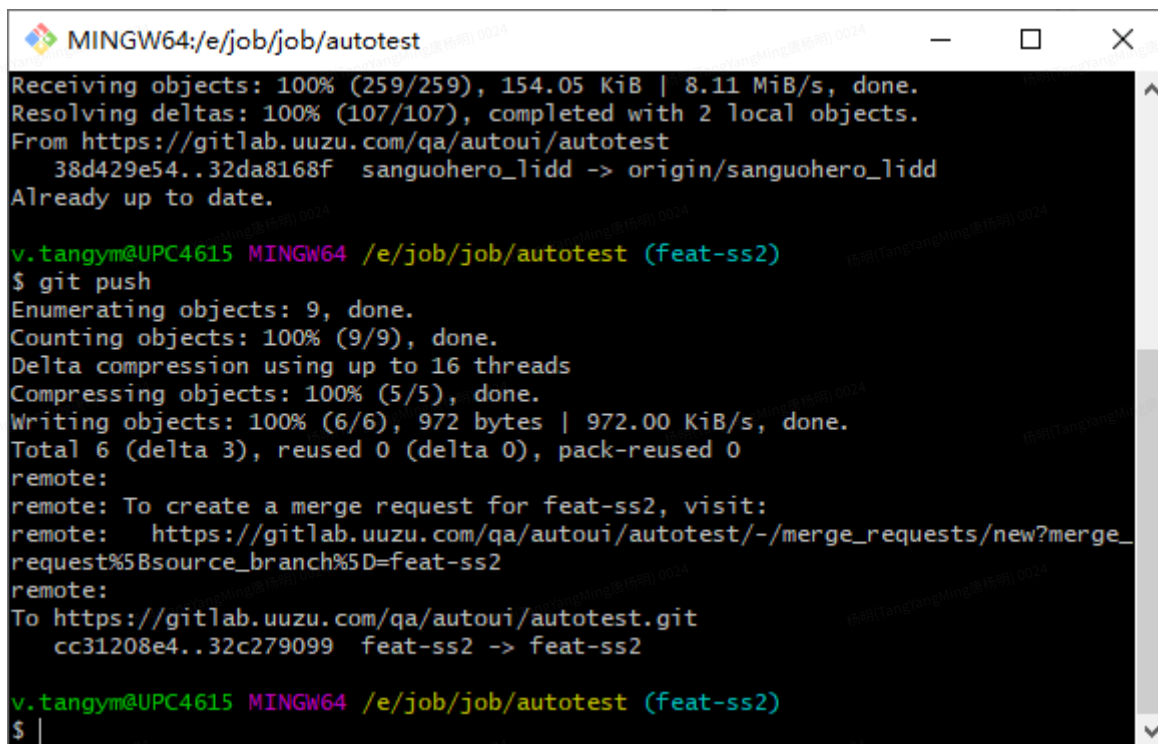
第三步：git commit -m [新增用例] (提交暂存区到仓库区，将索引的当前内容与描述更改的用户和日志消息一起存储在新的提交中)

```
v.tangym@UPC4615 MINGW64 /e/job/job/autotest (feat-ss2)
$ git commit -m [新增用例]
[feat-ss2 32c279099] [新增用例]
2 files changed, 31 insertions(+)
create mode 100644 demosdk/36ji_install.air/36ji_install.py
```

第四步：git pull (下拉代码，将远程最新的代码跟本地代码合并，完成之后打开代码查看有没有冲突，并解决，如果有冲突解决完成以后再次执行 2 和 3 的操作)

```
v.tangym@UPC4615 MINGW64 /e/job/job/autotest (feat-ss2)
$ git pull
remote: Enumerating objects: 259, done.
remote: Counting objects: 100% (259/259), done.
remote: Compressing objects: 100% (145/145), done.
Receiremote: Total 259 (delta 107), reused 218 (delta 80), pack-reused 0
Receiving objects: 100% (259/259), 154.05 KiB | 8.11 MiB/s, done.
Resolving deltas: 100% (107/107), completed with 2 local objects.
From https://gitlab.uuzu.com/qa/autoui/autotest
   38d429e54..32da8168f  sanguohero_lidd -> origin/sanguohero_lidd
Already up to date.
```

第五步：git push （将代码推至远程）

A screenshot of a terminal window titled 'MINGW64:/e/job/job/autotest'. The terminal shows the output of a 'git push' command. It starts with 'Receiving objects: 100% (259/259), 154.05 KiB | 8.11 MiB/s, done.' and 'Resolving deltas: 100% (107/107), completed with 2 local objects.' followed by 'From https://gitlab.uuzu.com/qa/autoui/autotest' and '38d429e54..32da8168f sanguohero\_lidd -> origin/sanguohero\_lidd'. Then it shows 'Already up to date.' followed by the prompt 'v.tangym@UPC4615 MINGW64 /e/job/job/autotest (feat-ss2)'. The next command is '\$ git push', followed by 'Enumerating objects: 9, done.', 'Counting objects: 100% (9/9), done.', 'Delta compression using up to 16 threads', 'Compressing objects: 100% (5/5), done.', 'Writing objects: 100% (6/6), 972 bytes | 972.00 KiB/s, done.', 'Total 6 (delta 3), reused 0 (delta 0), pack-reused 0'. Then it shows 'remote:' followed by 'remote: To create a merge request for feat-ss2, visit:', 'remote: https://gitlab.uuzu.com/qa/autoui/autotest/-/merge\_requests/new?merge\_request%5Bsource\_branch%5D=feat-ss2', 'remote: To https://gitlab.uuzu.com/qa/autoui/autotest.git', 'cc31208e4..32c279099 feat-ss2 -> feat-ss2'. Finally, it shows the prompt 'v.tangym@UPC4615 MINGW64 /e/job/job/autotest (feat-ss2)' and '\$ |'.

第二种方法：

第一步：git stash （这是将本地代码回滚值至上一次提交的时候，就是没有你新改的代码）

第二步：git pull origin master （将远程的拉下来）

第三步：git stash pop （将第一步回滚的代码释放出来，相等于将你修改的代码与下拉的代码合并）

然后解决冲突，你本地的代码将会是最新的代码

第四步：git add .

第五步：git commit -m[提交注释]

第六步：git push origin master

第七步：git pull origin master （确保远程的全部拉下来）