



Git配置与使用

windows:

Git安装去官网安装，然后就可以用了，用git bash命令行。

第一次安装之后配置user.name和user.email。

```
git config --global user.name "Your Name"
```

```
git config --global user.e mail "Your Name"
```

在bash中切换到需要用到git的目录之后，git init初始化仓库。

文件操作

- **git add <file>**：将指定文件添加到暂存区，准备进行提交。如果想添加所有修改过的文件，可以使用 **git add.**（添加当前目录下所有文件）或 **git add -A**（添加工作区所有文件，包括删除的文件）。
- **git rm <file>**：从暂存区和工作目录中删除指定文件，并记录这次删除操作到下次提交中。
- **git mv <old_file> <new_file>**：重命名或移动文件，它会自动记录文件的变更，相当于先执行 **mv** 命令，再执行 **git add**。
- **git status**：查看当前 Git 仓库的状态，包括哪些文件被修改、哪些文件在暂存区、哪些文件未被跟踪等信息。
- **git branch**：列出当前仓库的所有分支，当前所在分支前会有一个 ***** 标记。
- **git branch <branch_name>**：创建一个名为 **<branch_name>** 的新分支，但不会切换到新分支。
- **git checkout <branch_name>**：切换到指定的分支，同时更新工作目录的文件内容为该分支的最新状态。
- **git checkout -b <new_branch>**：创建一个新分支并立即切换到该分支，相当于 **git branch <new_branch>** 和 **git checkout <new_branch>** 的组合操作。
- **git branch -d <branch_name>**：删除指定的分支，前提是该分支的内容已经被合并到其他分支中。如果要强制删除未合并的分支，可以使用 **git branch -D <branch_name>**。

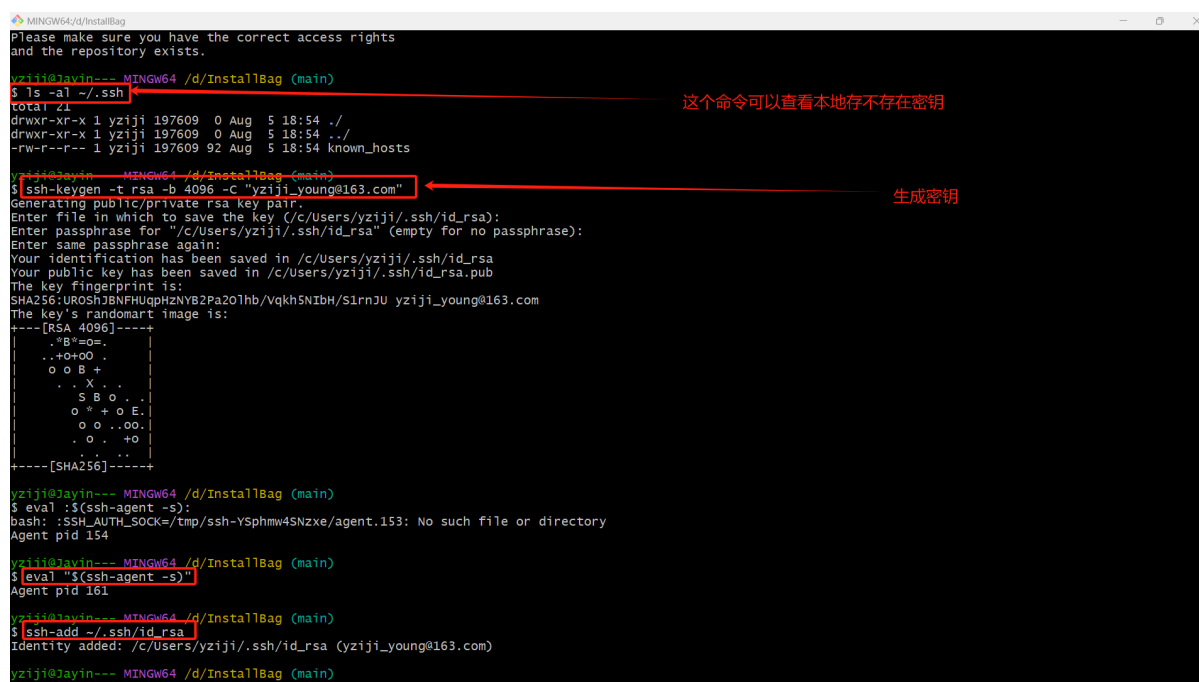
- `git commit -m "<message>"`：将暂存区的文件提交到本地仓库，`m` 选项后面跟着提交说明，用于描述本次提交的主要内容。
- `git commit --amend`：修改最近一次提交的提交说明或添加 / 移除文件，会用新的提交替换原来的提交。
- `git log`：查看提交历史记录，默认按照时间倒序显示每次提交的哈希值、作者、日期和提交说明等信息。常用选项有 `-pretty=oneline`（单行显示提交信息）、`-graph`（以图形化展示分支合并情况）、`n`（限制显示的提交数量，如 `git log -3` 表示显示最近 3 条提交记录）。
- `git reflog`：查看本地引用（如分支指针）的更新历史，记录了 `HEAD` 指针的移动情况，在误操作后恢复到之前的状态时非常有用。
- `git remote`：列出当前仓库配置的远程仓库名称，默认远程仓库名称为 `origin`。
- `git remote add <remote_name> <remote_url>`：添加一个远程仓库，`<remote_name>` 是自定义的远程仓库名称，`<remote_url>` 是远程仓库的地址。
- `git push <remote_name> <local_branch>:<remote_branch>`：将本地分支 `<local_branch>` 推送到远程仓库 `<remote_name>` 的 `<remote_branch>` 分支上。如果本地分支和远程分支名称相同，可以简化为 `git push <remote_name> <branch_name>`。常见的 `git push origin main` 表示将本地的 `main` 分支推送到名为 `origin` 的远程仓库的 `main` 分支。
- `git pull <remote_name> <branch_name>`：从远程仓库 `<remote_name>` 的 `<branch_name>` 分支拉取最新代码，并自动合并到本地当前分支，相当于 `git fetch` 和 `git merge` 的组合操作。
- `git fetch <remote_name>`：从远程仓库 `<remote_name>` 拉取最新的分支和提交信息，但不会自动合并到本地分支，方便在合并前查看远程仓库的变化。
- `git merge <branch_name>`：将指定分支 `<branch_name>` 的修改合并到当前分支。如果合并过程中出现冲突，需要手动解决冲突后再提交。
- `git rebase <base_branch>`：将当前分支的提交移动到 `<base_branch>` 的最新提交之后，使得提交历史更加线性。在多人协作中，变基可以让分支历史更清晰，但需要注意避免在公共分支上滥用，以免影响其他开发者。
- `git stash`：将当前工作区的修改暂存起来，以便切换分支或进行其他操作，之后可以使用 `git stash apply` 恢复暂存的修改，`git stash drop` 丢弃暂存的修改。
- `git reset <commit_hash>`：重置当前分支到指定的提交 `<commit_hash>`，有 `-soft`（只修改 `HEAD` 指针，暂存区和工作区不变）、`-mixed`（默认模式，修改 `HEAD` 指针和暂存区，工作区不变）、`-hard`（修改 `HEAD` 指针、暂存区和工作区，可能会丢失未提交的修改，需谨慎使用）等选项。

本地GIT如何推到GITHub中：可以作为保存文件使用

首先在git创建库之后会出现这么一段提示，其中都是在本地文件夹中实现的

```
echo "# MyBag" >> README.md
git init //初始化本地
git add README.md //创建一个文件
git commit -m "first commit" //提交
git branch -M main //分支名必须和github库中的一致，将当前所在的分支重命名为 main
或者用这一段
git branch -m master main//更改分支名
git remote add origin git@github.com:Jayin-Zig/MyBag.git//指向远程库关联
git push -u origin main//推库
```

这里可能会出现连接不上的原因，应该是ssh对不上，需要看下本地ssh生成了吗？



```
MINGW64/d/InstallBag
Please make sure you have the correct access rights
and the repository exists.

yziji@Jayin--- MINGW64 /d/InstallBag (main)
$ ls -al ~/.ssh
total 21
drwxr-xr-x 1 yziji 197609  0 Aug  5 18:54 ./
drwxr-xr-x 1 yziji 197609  0 Aug  5 18:54 ../
-rw-r--r-- 1 yziji 197609 92 Aug  5 18:54 known_hosts

yziji@Jayin--- MINGW64 /d/InstallBag (main)
$ ssh-keygen -t rsa -b 4096 -C "yziji_young@163.com"
Generating public/private rsa key pair.
Enter file in which to save the key (/c:/Users/yziji/.ssh/id_rsa):
Enter passphrase for "/c:/Users/yziji/.ssh/id_rsa" (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /c:/Users/yziji/.ssh/id_rsa
Your public key has been saved in /c:/Users/yziji/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:UROSJBNFHUqHhZNYB2Pa20lhb/Vqkh5NIBH/S1rn3U yziji_young@163.com
The key's randomart image is:
+-----[RSA 4096]-----+
|
|..+o+o+
|o o B +
|.. X .
|S B o .
|o * + o E.
|o o ..oo.
|. o . +o
|
+-----[SHA256]-----+

yziji@Jayin--- MINGW64 /d/InstallBag (main)
$ eval :$(ssh-agent -s):
bash: :SSH_AUTH_SOCK=/tmp/ssh-YSpHm4SNzxe/agent.153: No such file or directory
Agent pid 154

yziji@Jayin--- MINGW64 /d/InstallBag (main)
$ eval "$(ssh-agent -s)"
Agent pid 161

yziji@Jayin--- MINGW64 /d/InstallBag (main)
$ ssh-add ~/.ssh/id_rsa
Identity added: /c:/Users/yziji/.ssh/id_rsa (yziji_young@163.com)

yziji@Jayin--- MINGW64 /d/InstallBag (main)
```

这个命令可以查看本地存不存在密钥

生成密钥

`ls -al ~/.ssh` 中可以看有没有id_rsa和id_rsa.pub。没有的话说明本地没有密钥

```
yziji@Jayin--- MINGW64 /d/InstallBag (main)
$ ls -al ~/.ssh
total 37
drwxr-xr-x 1 yziji 197609 0 Aug 5 19:00 ./
drwxr-xr-x 1 yziji 197609 0 Aug 5 18:54 ../
-rw-r--r-- 1 yziji 197609 3389 Aug 5 18:58 id_rsa
-rw-r--r-- 1 yziji 197609 745 Aug 5 18:58 id_rsa.pub
-rw-r--r-- 1 yziji 197609 828 Aug 5 19:00 known_hosts
-rw-r--r-- 1 yziji 197609 92 Aug 5 18:54 known_hosts.old
```

那么接下来就通过命令创建密钥 `ssh-keygen -t rsa -b 4096 -c "邮箱"`.

然后执行 `eval "$(ssh-agent -s)"` 和 `ssh-add ~/.ssh/id_rsa`

之后用

`cat ~/.ssh/id_rsa.pub`

复制密钥，添加到github中就好。

The screenshot shows the GitHub 'SSH keys' management interface. On the left, the 'SSH and GPG keys' option is selected in the sidebar. The main content area shows a list of SSH keys. The 'Jayin' key, added on Aug 5, 2025, is highlighted with a red box. A 'New SSH key' button is located in the top right corner. Below the SSH keys section, there are sections for 'GPG keys' (showing no keys) and 'Vigilant mode' (with a checkbox for 'Flag unsigned commits as unverified').