



## 1.2 创建本地仓库

- 在电脑的任意位置创建一个空目录（例如test）作为我们的本地Git仓库  
进入这个目录中，点击右键打开Git bash窗口  
执行命令 **git init**  
如果创建成功后可在文件夹下看到隐藏的.git目录  
**git init**

```
Hongyi@MSI MINGW64 /e/develop/test
$ git init
Initialized empty Git repository in E:/develop/test/.git/

Hongyi@MSI MINGW64 /e/develop/test (master)
$ ll
total 8
drwxr-xr-x 1 Hongyi 197121 0 Jan 18 14:47 ./
drwxr-xr-x 1 Hongyi 197121 0 Jan 18 14:47 ../
drwxr-xr-x 1 Hongyi 197121 0 Jan 18 14:47 .git/

Hongyi@MSI MINGW64 /e/develop/test (master)
$
```

## 1.3 基础操作指令

- status查看修改的状态  
作用：查看修改的状态（暂存区、工作区）  
命令形式：  
**git status**
- add添加工作区到暂存区  
作用：添加工作区一个或多个文件的修改到暂存区  
命令形式：git add 单个文件名 | 通配符  
**git add file.txt # 添加单个文件**  
**git add . # 将所有修改加入暂存区**
- commit提交暂存区到本地仓库  
作用：提交暂存区内容到本地仓库的当前分支  
命令形式：git commit -m “注释内容”  
**git commit -m "XXX update"**
- log查看提交日志  
配置的别名git-log就包含了这些参数，所以后续可以直接使用指令 **git-log**  
作用：查看提交记录  
命令形式：git log [option] 或者 git-log  
—all 显示所有分支  
—pretty=oneline 将提交信息显示为一行  
—abbrev-commit 使得输出的commitId更简短  
—graph 以图的形式显示

- 版本回退  
作用：版本切换  
命令形式：

**git reset --hard commitID** # commitID 可以使用 git-log 或 git log 指令查看

- 添加文件至忽略列表  
一般我们总会有些文件无需纳入Git的管理，也不希望它们总出现在未跟踪文件列表。通常都是些自动生成的文件，比如日志文件，或者编译过程中创建的临时文件等。在这种情况下，我们可以在工作目录中创建一个名为 **.gitignore** 的文件（文件名称固定），列出要忽略的文件模式。

#### 1.4命令使用示例

- 利用上面初始化好的本地仓库test，新建一个文件file01.txt，并用status查看状态：

**touch file01.txt**  
**git status**

```
Hongyi@MSI MINGW64 /e/develop/test (master)
$ touch file01.txt

Hongyi@MSI MINGW64 /e/develop/test (master)
$ ll
total 8
drwxr-xr-x 1 Hongyi 197121 0 Jan 18 14:55 ./
drwxr-xr-x 1 Hongyi 197121 0 Jan 18 14:47 ../
drwxr-xr-x 1 Hongyi 197121 0 Jan 18 14:47 .git/
-rw-r--r-- 1 Hongyi 197121 0 Jan 18 14:55 file01.txt

Hongyi@MSI MINGW64 /e/develop/test (master)
$ git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
       file01.txt

nothing added to commit but untracked files present (use "git add" to track)

Hongyi@MSI MINGW64 /e/develop/test (master)
$ |
```

- 添加到暂存区

**git add file01.txt**

或者

**git add .**

```
Hongyi@MSI MINGW64 /e/develop/test (master)
$ git add .

Hongyi@MSI MINGW64 /e/develop/test (master)
$ git status
On branch master 在分支branch上

No commits yet

Changes to be committed: 即将被提交
  (use "git rm --cached <file>..." to unstage)
       new file:   file01.txt 发现add后，文件名变绿了
```

- 提交至本地仓库并查看状态

**git commit -m "commit file01.txt"**  
**git status**

```
Hongyi@MSI MINGW64 /e/develop/test (master)
$ git commit -m "commit file01.txt"
[master (root-commit) b6a4a39] commit file01.txt
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 file01.txt

Hongyi@MSI MINGW64 /e/develop/test (master)
$ git status
On branch master
nothing to commit, working tree clean
```

- 查看日志

**git log**

```
Hongyi@MSI MINGW64 /e/develop/test (master)
$ git log
commit b6a4a39178e0fc6ab4db78a96d2fab3928b9bc77 (HEAD -> master)
Author: 
Date: Tue Jan 18 15:13:05 2022 +0800

    commit file01.txt
```

- 修改file01.txt文件内容并查看状态

```
Hongyi@MSI MINGW64 /e/develop/test (master)
$ vi file01.txt

Hongyi@MSI MINGW64 /e/develop/test (master)
$ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
    modified:   file01.txt

no changes added to commit (use "git add" and/or "git commit -a")
```

- 添加至暂存区并查看状态

```
Hongyi@MSI MINGW64 /e/develop/test (master)
$ git add .
warning: LF will be replaced by CRLF in file01.txt.
The file will have its original line endings in your working directory

Hongyi@MSI MINGW64 /e/develop/test (master)
$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    modified:   file01.txt
```

- 提交至本地仓库并查看日志

```
Hongyi@MSI MINGW64 /e/develop/test (master)
$ git commit -m "commit file01.txt update_count = 1"
[master 9135087] commit file01.txt update_count = 1
1 file changed, 1 insertion(+)

Hongyi@MSI MINGW64 /e/develop/test (master)
$ git log
commit 9135087c689f25bc6aa6356fa07320e1dbf30fb4 (HEAD -> master)
Author: 
Date: Tue Jan 18 15:19:43 2022 +0800

    commit file01.txt update_count = 1

commit b6a4a39178e0fc6ab4db78a96d2fab3928b9bc77
Author: 
Date: Tue Jan 18 15:13:05 2022 +0800

    commit file01.txt
```

- 利用git-log查看提交日志，并回退至第一个版本

**git-log**

**git reset --hard XXXXX**

```
Hongyi@MSI MINGW64 /e/develop/test (master)
$ git-log
* 9135087 (HEAD -> master) commit file01.txt update_count = 1
* b6a4a39 commit file01.txt
版本号
Hongyi@MSI MINGW64 /e/develop/test (master)
$ git reset --hard b6a4a39
HEAD is now at b6a4a39 commit file01.txt

Hongyi@MSI MINGW64 /e/develop/test (master)
$ git-log
* b6a4a39 (HEAD -> master) commit file01.txt
```

- 点击打开file01.txt，发现其中内容没有了。

同样，可以再次回到第二个版本：

```
Hongyi@MSI MINGW64 /e/develop/test (master)
$ git reset --hard 9135087
HEAD is now at 9135087 commit file01.txt update_count = 1

Hongyi@MSI MINGW64 /e/develop/test (master)
$ git-log
* 9135087 (HEAD -> master) commit file01.txt update_count = 1
* b6a4a39 commit file01.txt
```

- 新建文件file02.txt，加入.gitignore中

**touch file02.txt**

**touch .gitignore**

**vi .gitignore # 添加内容： file02.txt**

.gitignore内容

**file02.txt**

也可以使用通配符，例如

**\*.txt**

**\*.iml**

...

## 1.5分支

几乎所有的版本控制系统都以某种形式支持分支。使用分支意味着你可以把你的工作从开发主线上分离开来，进行重大的Bug修改、开发新的功能，以免影响开发主线。

- 查看本地分支

**git branch**

- 创建本地分支

**git branch 分支名**

- 切换分支

**git checkout 分支名**

- 我们还可以直接切换到一个不存在的分支（创建并切换）：

**git checkout -b 分支名**

- 合并分支

一个分支上的提交可以合并到另一个分支

**git merge 分支名**

- 删除分支

不能删除当前分支，只能删除其他分支

**git branch -d 分支名 # 删除分支时，需要做各种检查**

**git branch -D 分支名 # 不做任何检查，强制删除**

## 1.6解决冲突

当两个分支上对文件的修改可能会存在冲突，例如同时修改了同一个文件的同一行，这时就需要手动解决冲突，解决冲突步骤如下：

1. 处理文件中冲突的地方
2. 将解决完冲突的文件加入暂存区(add)
3. 提交到仓库(commit)

## 1.7命令使用示例

- 示例1

- 查看分支，并新建分支dev01

**git branch**

**git branch dev01**

```
Hongyi@MSI MINGW64 /e/develop/test (master)
$ git branch 查看分支
* master

Hongyi@MSI MINGW64 /e/develop/test (master)
$ git-log 也可以通过git-log查看，记住这个指令很好用!
* 9135087 (HEAD -> master) commit file01.txt update_count = 1
* b6a4a39 commit file01.txt

Hongyi@MSI MINGW64 /e/develop/test (master)
$ git branch dev01 新建分支dev01

Hongyi@MSI MINGW64 /e/develop/test (master)
$ git branch 查看分支
dev01
* master

Hongyi@MSI MINGW64 /e/develop/test (master)
$ git-log
* 9135087 (HEAD -> master, dev01) commit file01.txt update_count = 1
* b6a4a39 commit file01.txt
```

其中HEAD指向的是当前工作区所处的分支，例如此时处在master的分支上。

- 将上一节中的.gitignore提交，并查看分支：

```
Hongyi@MSI MINGW64 /e/develop/test (master)
$ git-log
* 98080db (HEAD -> master) gitignore commit
* 9135087 (dev01) commit file01.txt update_count = 1
* b6a4a39 commit file01.txt
```

- 将当前分支切换到dev01

**git checkout dev01**

```
Hongyi@MSI MINGW64 /e/develop/test (master)
$ git checkout dev01
Switched to branch 'dev01'

Hongyi@MSI MINGW64 /e/develop/test (dev01)
$ git-log
* 98080db (master) gitignore commit
* 9135087 (HEAD -> dev01) commit file01.txt update_count = 1
* b6a4a39 commit file01.txt
```

同时观察到，test目录下的.gitignore消失了

- 重新切换到master分支，.gitignore又出现了。新建分支dev02，并同时切换到上面

**git checkout -b dev02**

```
Hongyi@MSI MINGW64 /e/develop/test (master)
$ git checkout -b dev02
Switched to a new branch 'dev02'

Hongyi@MSI MINGW64 /e/develop/test (dev02)
$ git-log
* 98080db (HEAD -> dev02, master) gitignore commit
* 9135087 (dev01) commit file01.txt update_count = 1
* b6a4a39 commit file01.txt
```

- 切换到dev01分支，并新建文件file03.txt，并提交

```
Hongyi@MSI MINGW64 /e/develop/test (dev01)
$ git-log
* 8b48187 (HEAD -> dev01) file03.txt commit
| * 98080db (master, dev02) gitignore commit
|/
* 9135087 commit file01.txt update_count = 1
* b6a4a39 commit file01.txt
```

- 切换到master分支，并将dev01的提交合并到master上

**git checkout master**

**git merge dev01**

```
Hongyi@MSI MINGW64 /e/develop/test (master)
$ git-log
* 8b48187 (dev01) file03.txt commit
| * 98080db (HEAD -> master, dev02) gitignore commit
|/
* 9135087 commit file01.txt update_count = 1
* b6a4a39 commit file01.txt

Hongyi@MSI MINGW64 /e/develop/test (master)
$ git merge dev01
Merge made by the 'recursive' strategy.
file02.txt | 0
file03.txt | 0
2 files changed, 0 insertions(+), 0 deletions(-)
create mode 100644 file02.txt
create mode 100644 file03.txt

Hongyi@MSI MINGW64 /e/develop/test (master)
$ git-log
* 2bc3664 (HEAD -> master) Merge branch 'dev01'
| \
| * 8b48187 (dev01) file03.txt commit
| * | 98080db (dev02) gitignore commit
|/
* 9135087 commit file01.txt update_count = 1
* b6a4a39 commit file01.txt
```

合并前

合并后

此时，可以看到在master分支下的仓库中，有了file03.txt

- 删除dev02分支

**git branch -d dev02**

```
Hongyi@MSI MINGW64 /e/develop/test (master)
$ git branch -d dev02
Deleted branch dev02 (was 98080db).

Hongyi@MSI MINGW64 /e/develop/test (master)
$ git-log
* 2bc3664 (HEAD -> master) Merge branch 'dev01'
| \
| * 8b48187 (dev01) file03.txt commit
| * | 98080db gitignore commit
|/
* 9135087 commit file01.txt update_count = 1
* b6a4a39 commit file01.txt
```

## • 示例2—解决冲突

- 删除分支dev01，创建并切换到分支dev上。此时master和dev分支上的file01.txt文件中内容都为update\_count = 1

**git branch -d dev01**

**git checkout -b dev**

```
Hongyi@MSI MINGW64 /e/develop/test (master)
$ git branch -d dev01
Deleted branch dev01 (was 8b48187).

Hongyi@MSI MINGW64 /e/develop/test (master)
$ git-log
* 2bc3664 (HEAD -> master) Merge branch 'dev01'
| \
| * 8b48187 file03.txt commit
| * | 98080db gitignore commit
|/
* 9135087 commit file01.txt update_count = 1
* b6a4a39 commit file01.txt
```

```
Hongyi@MSI MINGW64 /e/develop/test (master)
$ git checkout -b dev
Switched to a new branch 'dev'

Hongyi@MSI MINGW64 /e/develop/test (dev)
$ git-log
* 2bc3664 (HEAD -> dev, master) Merge branch 'dev01'
| \
| * 8b48187 file03.txt commit
| * | 98080db gitignore commit
| /
* 9135087 commit file01.txt update_count = 1
* b6a4a39 commit file01.txt
```

- 在dev分支上修改file01.txt文件内容为update\_count = 2，并提交

```
Hongyi@MSI MINGW64 /e/develop/test (dev)
$ git add .

Hongyi@MSI MINGW64 /e/develop/test (dev)
$ git commit -m "dev file01 count = 2"
[dev 45e7c54] dev file01 count = 2
1 file changed, 1 insertion(+), 1 deletion(-)

Hongyi@MSI MINGW64 /e/develop/test (dev)
$ git-log
* 45e7c54 (HEAD -> dev) dev file01 count = 2
* 2bc3664 (master) Merge branch 'dev01'
| \
| * 8b48187 file03.txt commit
| * | 98080db gitignore commit
| /
* 9135087 commit file01.txt update_count = 1
* b6a4a39 commit file01.txt
```

- 在master分支上修改file01.txt文件内容为update\_count = 3，并提交

```
Hongyi@MSI MINGW64 /e/develop/test (master)
$ git add .

Hongyi@MSI MINGW64 /e/develop/test (master)
$ git commit -m "master file01 count = 3"
[master 583e08a] master file01 count = 3
1 file changed, 1 insertion(+), 1 deletion(-)

Hongyi@MSI MINGW64 /e/develop/test (master)
$ git-log
* 583e08a (HEAD -> master) master file01 count = 3
| * 45e7c54 (dev) dev file01 count = 2
| /
| * 2bc3664 Merge branch 'dev01'
| \
| * 8b48187 file03.txt commit
| * | 98080db gitignore commit
| /
* 9135087 commit file01.txt update_count = 1
* b6a4a39 commit file01.txt
```

- 将dev分支合并到master分支上，发现报错。查看file01.txt内容

**git merge dev**

```
Hongyi@MSI MINGW64 /e/develop/test (master)
$ git merge dev
Auto-merging file01.txt
CONFLICT (content): Merge conflict in file01.txt
Automatic merge failed; fix conflicts and then commit the result.
<----- HEAD
update_count = 3
=====
update_count = 2
>>>>>> dev
```

从HEAD向下到=处，为当前分支(master)提交的内容

从dev向上到=处，为其他分支(dev)提交的内容

- 解决冲突：将file01.txt提示冲突的地方修改成我们想要的内容，例如再次修改成update\_count = 3，并再次提交即可。



# Git远程仓库

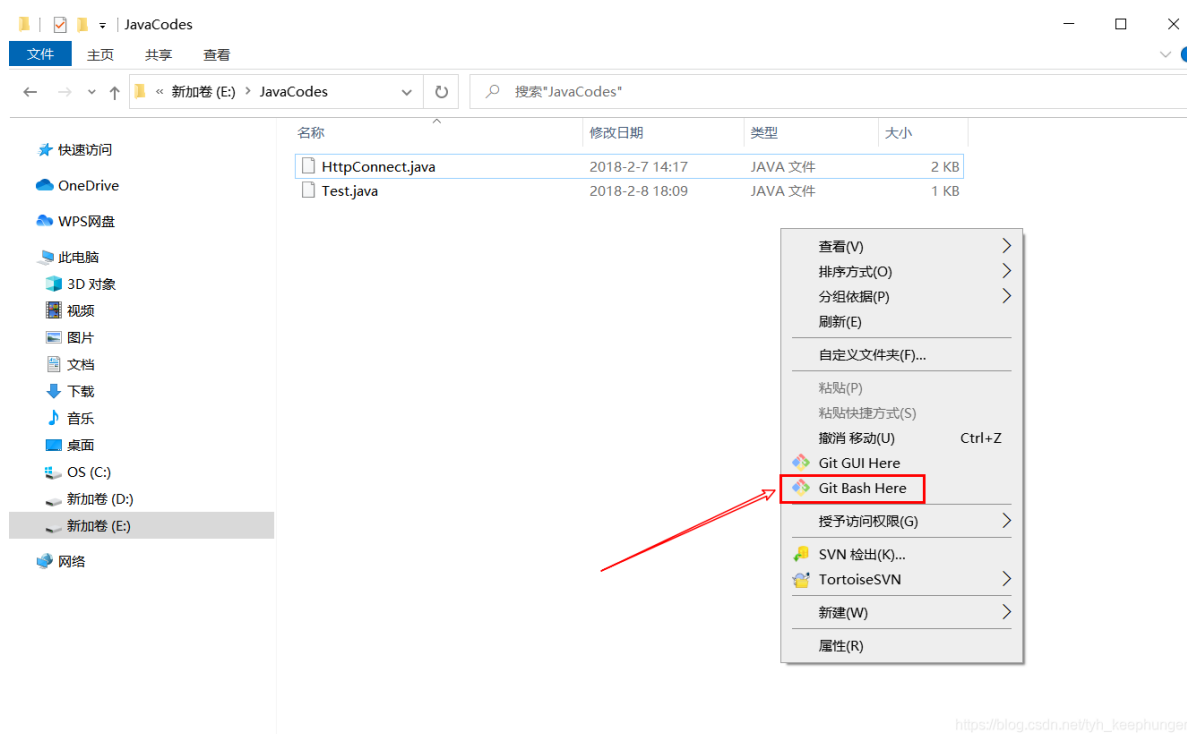
## 1.常用的托管服务

前面我们已经知道了Git中存在两种类型的仓库，即本地仓库和远程仓库。那么我们如何搭建Git远程仓库呢？我们可以借助互联网上提供的一些代码托管服务来实现，其中比较常用的有GitHub、码云、GitLab等。

- GitHub（地址：<https://github.com/>）是一个面向开源及私有软件项目的托管平台，因为只支持Git作为唯一的版本库格式进行托管，故名GitHub
- 码云（地址：<https://gitee.com/>）是国内的一个代码托管平台，由于服务器在国内，所以相比于GitHub，码云速度会更快
- GitLab（地址：<https://about.gitlab.com/>）是一个用于仓库管理系统的开源项目，使用Git作为代码管理工具，并在此基础上搭建起来的web服务，一般用于在企业、学校等内部网络搭建git私服。

## 2.操作远程仓库--以GitHub为例

- 打开存放代码的文件夹目录，鼠标右键选择"Git Bash Here"，打开Git命令行窗口；

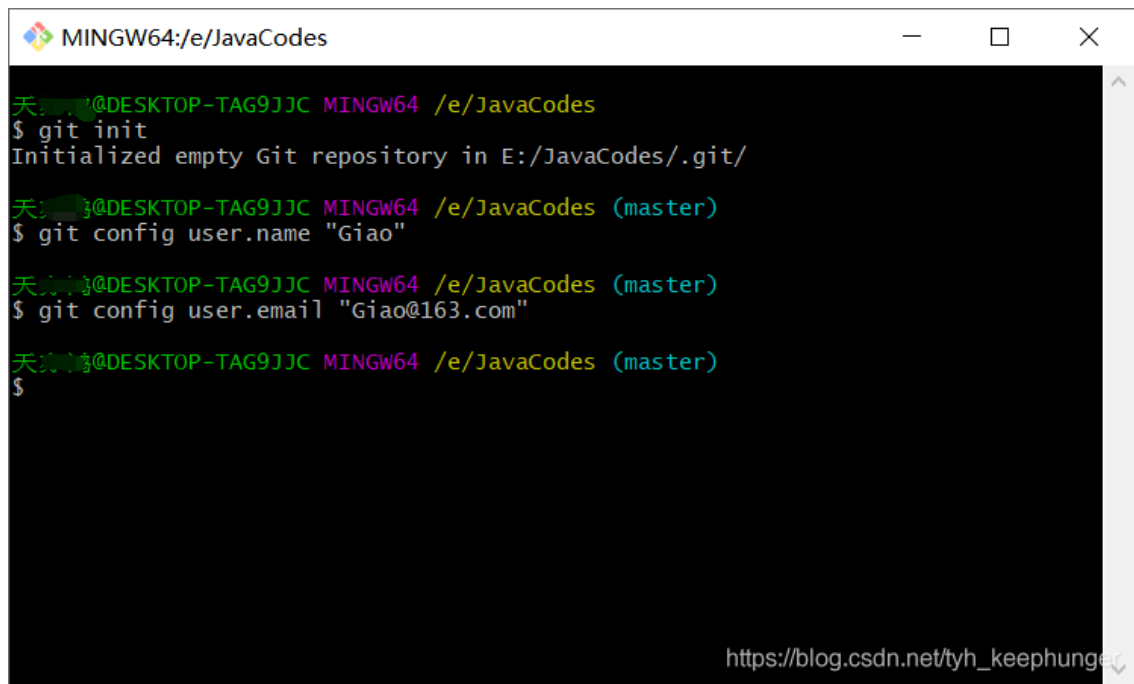


- 初始化本地仓库

### 1.git init

### 2.git --global config user.name "Zhang3"

git --global config user.email "[zhang3@163.com](mailto:zhang3@163.com)"



```
MINGW64:/e/JavaCodes
天...@DESKTOP-TAG9JJC MINGW64 /e/JavaCodes
$ git init
Initialized empty Git repository in E:/JavaCodes/.git/

天...@DESKTOP-TAG9JJC MINGW64 /e/JavaCodes (master)
$ git config user.name "Giao"

天...@DESKTOP-TAG9JJC MINGW64 /e/JavaCodes (master)
$ git config user.email "Giao@163.com"

天...@DESKTOP-TAG9JJC MINGW64 /e/JavaCodes (master)
$
```

[https://blog.csdn.net/tyh\\_keeptung](https://blog.csdn.net/tyh_keeptung)

- 提交本地文件到Git服务器

3.git status 查看状态

4.git add . 添加文件

5.git commit 或者 git commit -m "注释"

6.git status 再次查看状态

7.git push -u origin master 推送本地代码到GitHub仓库

```
MINGW64:/e/JavaCodes

天...@DESKTOP-TAG9JJC MINGW64 /e/JavaCodes (master)
$ git status --查看状态, 对应第3点
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        HttpURLConnection.java
        Test.java

nothing added to commit but untracked files present (use "git add" to track)

天...@DESKTOP-TAG9JJC MINGW64 /e/JavaCodes (master)
$ git add . --添加当前目录的全部文件到暂存区, 对应第4点

天...@DESKTOP-TAG9JJC MINGW64 /e/JavaCodes (master)
$ git status --再次查看状态, 文件名颜色变成了绿色, 代表进入了暂存区
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   HttpURLConnection.java
        new file:   Test.java

天...@DESKTOP-TAG9JJC MINGW64 /e/JavaCodes (master)
$ git commit -m "20210414第一次提交" --提交文件到服务器, 对应第5点
[master (root-commit) 759393a] 20210414第一次提交
 2 files changed, 67 insertions(+)
 create mode 100644 HttpURLConnection.java
 create mode 100644 Test.java

天...@DESKTOP-TAG9JJC MINGW64 /e/JavaCodes (master)
$ git status --再次查看状态, 确认暂存区没有待提交的内容
On branch master
nothing to commit, working tree clean
```

[https://blog.csdn.net/tyh\\_keephung](https://blog.csdn.net/tyh_keephung)

```
@DESKTOP-KB95R1C MINGW64 /d/TyhStudy/PythonCodes (master)
$ git status
On branch master
Your branch is ahead of 'origin/master' by 1 commit.
  (use "git push" to publish your local commits)

nothing to commit, working tree clean

@DESKTOP-KB95R1C MINGW64 /d/TyhStudy/PythonCodes (master)
$ git push -u origin master 提交代码到GitHub
Warning: Permanently added the RSA host key for IP address '20.205.243.166' to the list of known hosts.
Enumerating objects: 35, done.
Counting objects: 100% (35/35), done.
Delta compression using up to 4 threads
Compressing objects: 100% (23/23), done.
Writing objects: 100% (23/23), 13.48 KiB | 690.00 KiB/s, done.
Total 23 (delta 3), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (3/3), completed with 3 local objects.
To github.com:lfh047375/Tyh_PyCodes.git
   8eb49d4..6e82a25 master -> master
Branch 'master' set up to track remote branch 'master' from 'origin'.

@DESKTOP-KB95R1C MINGW64 /d/TyhStudy/PythonCodes (master)
$
```

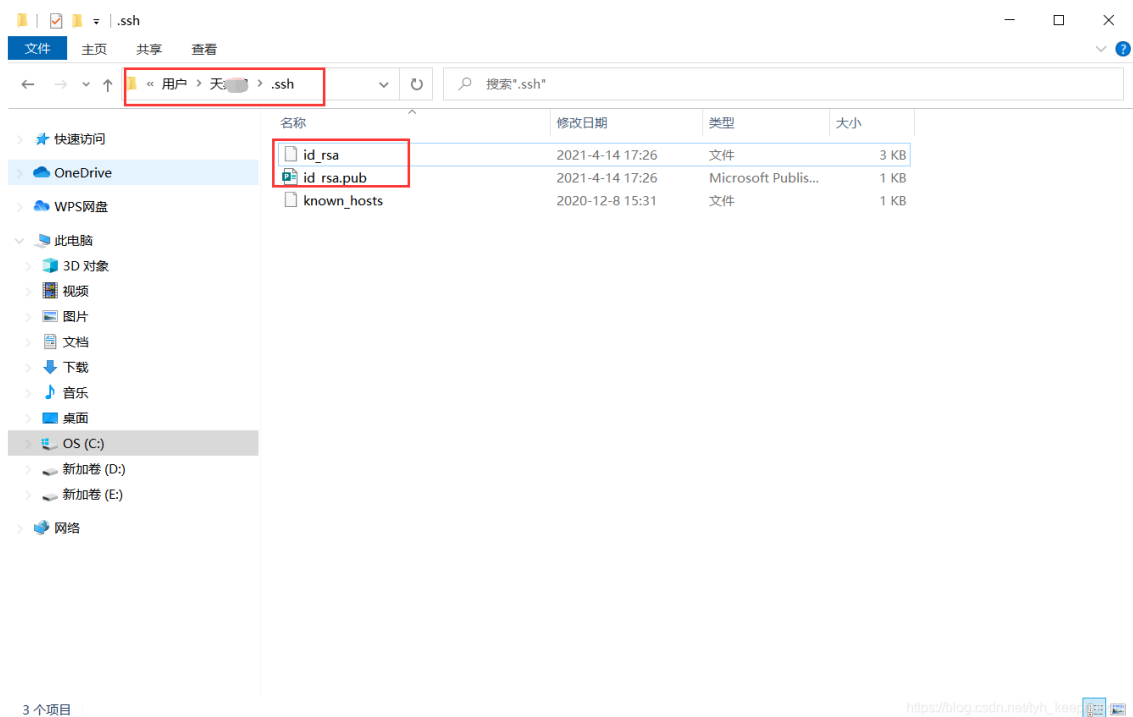
CSDN @tyh\_keepRunning

- 提交本地仓库的文件到GitHub
  - 先检查C盘用户目录下有没有.ssh目录, 有的话看下里面有没有id\_rsa和id\_rsa.pub这两个文件, 有就跳到下一步, 没有就在Git命令行输入命令: `ssh-keygen -t rsa -C "youremail@example.com"` 然后一路回车。这时你就会在用户下的.ssh目录里找到id\_rsa和id\_rsa.pub这两个文件。

```
MINGW64:/e/JavaCodes
@DESKTOP-TAG9JJC MINGW64 /e/JavaCodes (master)
$ ssh-keygen -t rsa -C "Giao@163.com"
Generating public/private rsa key pair.
Enter file in which to save the key (/c/Users/天亦鸿/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /c/Users/天亦鸿/.ssh/id_rsa
Your public key has been saved in /c/Users/天亦鸿/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:nse5FqFbDIHi2bwwRv/dq/D/cao2fb6d+x1SNrQEkr0 Giao@163.com
The key's randomart image is:
+---[RSA 3072]-----+
|
|  o . . . . .
| o * . . . . .o
| * + . . . . .Eo
| . o oS= o =
| ..oo=. . o .
| +O+. +. .
| .+. + o+*
| .++o+OBB
+---[SHA256]-----+

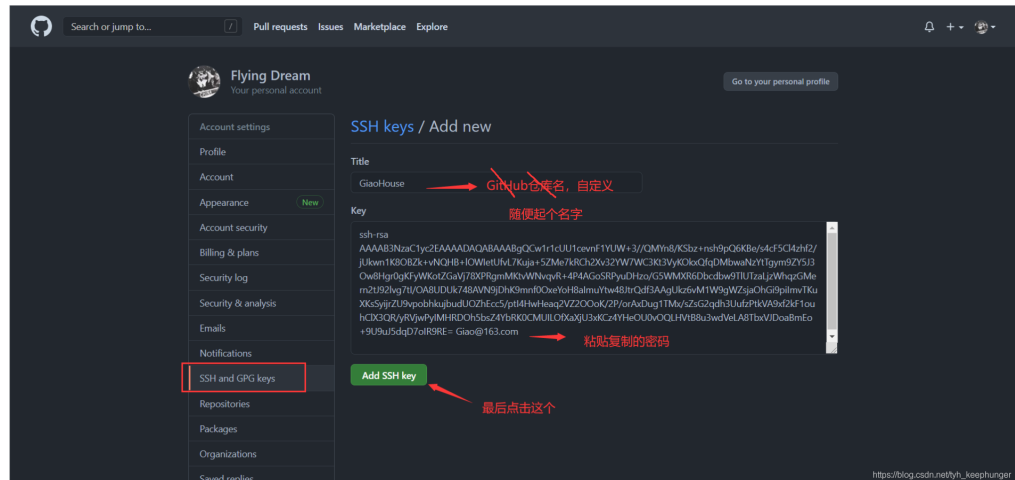
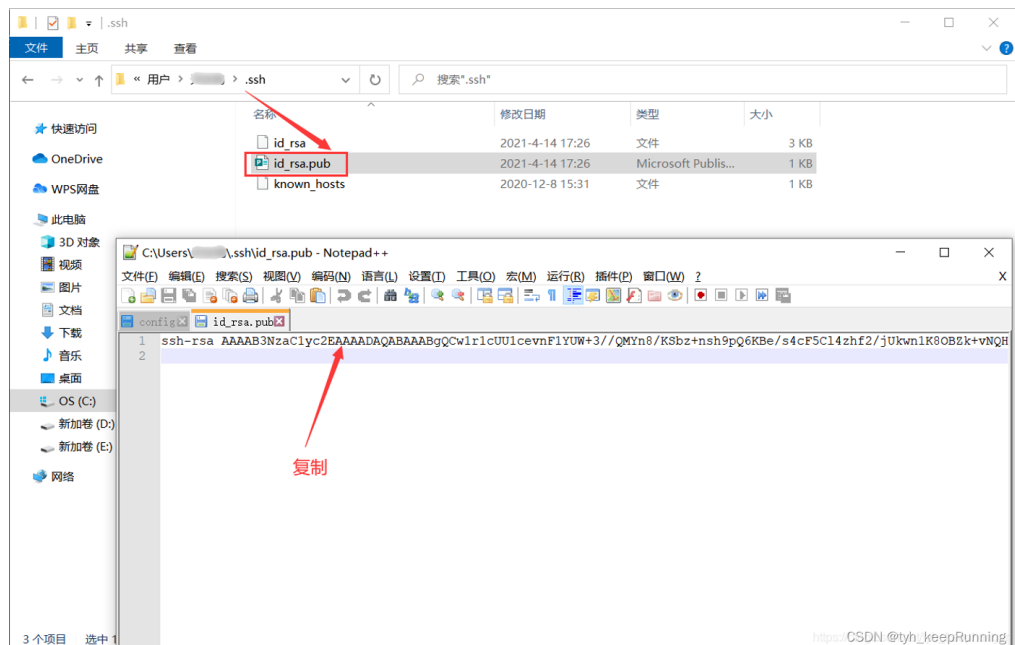
@DESKTOP-TAG9JJC MINGW64 /e/JavaCodes (master)
$
```

一路按回车键即可



打开[GitHub官网](https://github.com)

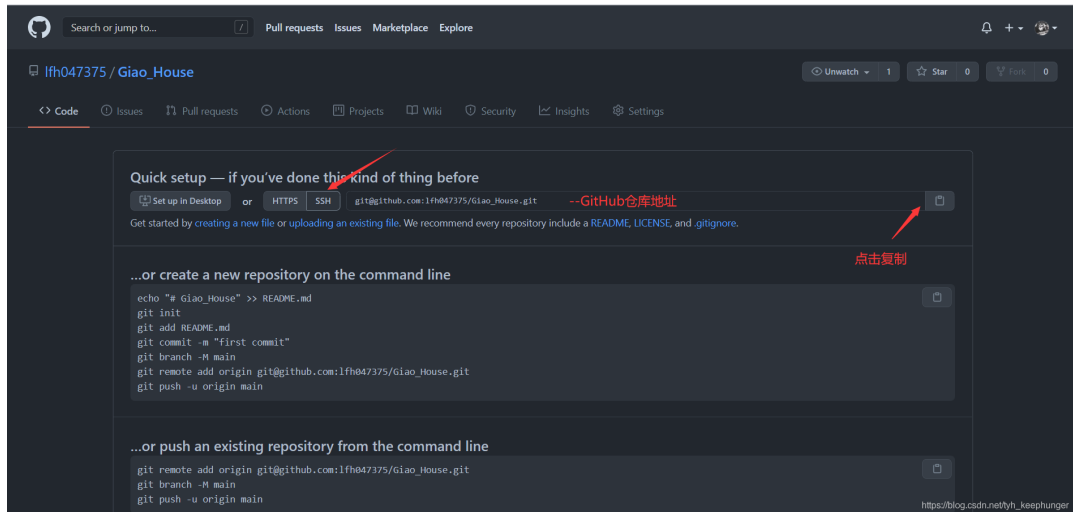
1. 登录GitHub，点开右上角头像，下拉选择Settings；
2. 选择"SSH and GPG KEYS"，点击右上角的"New SSH key"；
3. 然后Title里面随便填，再把刚才id\_rsa.pub里面的内容复制到Title下面的Key内容框里面，最后点击"Add SSH key"，这样就完成了SSH Key的加密。



- 在Github上创建一个Git仓库



- 将本地仓库代码关联到GitHub，先复制GitHub仓库地址，然后在Git命令行窗口输入命令：  
`git remote add origin GitHub仓库地址`



注意：

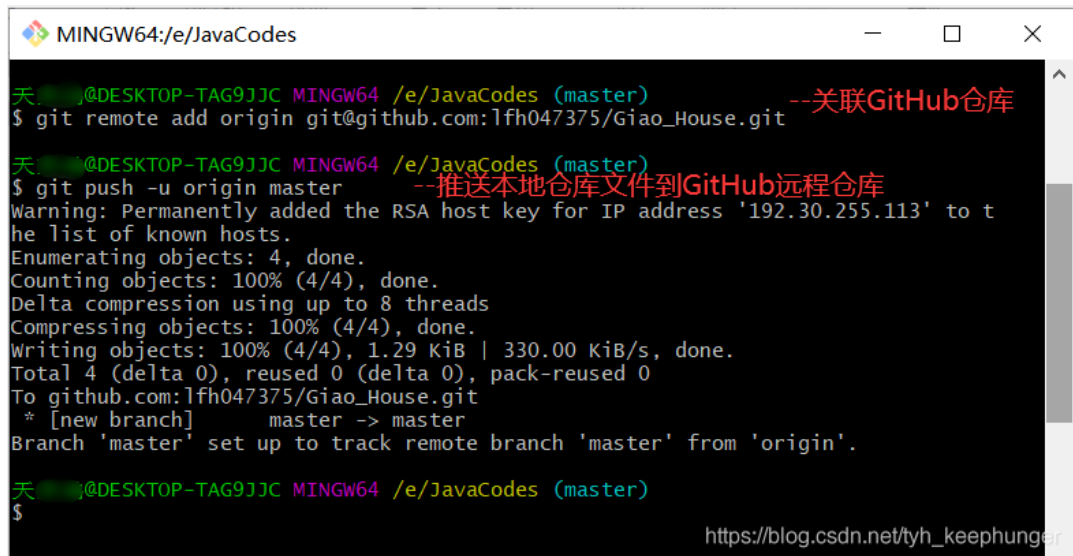
在这一步时如果出现错误：**'fatal: remote origin already exists.'**

说明本地库已经关联了origin远程库，可通过 `git remote -v` 查看关联的远程库；

如果要更改远程库，则先删除关联库，输入：`git remote rm origin`

再输入：`git remote add origin GitHub仓库地址`，就不会报错了

- 关联好之后，就到了最后一步命令，推送本地仓库项目到GitHub：`git push -u origin master`



注意：

一、执行git push命令时，如果出现错误：**'error: failed to push some refs to 'git@github.com:xxxxx'**

主要原因：GitHub仓库中的README.md文件不在本地代码目录中

解决办法：

1.先把GitHub仓库的文件与本地代码仓库的文件进行合并，输入命令：`git pull --rebase origin master`，合并后本地git代码目录下可以看到README.md文件。（如果执行git pull --rebase后，导致本地文件丢失，找回丢失文件办法：<https://blog.csdn.net/liuskyter/article/details/102745510>）

2.然后再输入命令：`git push -u origin master`，上传到GitHub仓库。

## 二、执行git push命令时，如果出现错误：‘fatal: Could not read from remote repository.’

主要原因：客户端与服务端未生成 ssh key，或者客户端与服务端的ssh key不匹配

解决办法：

1.生成新的SSH key：如果是客户端与服务端未生成ssh key，那么直接使用：`ssh-keygen -t rsa -C "youremail@example.com"` 生成新的rsa密钥即可。如果是客户端与服务端的ssh key 不匹配，此时需要先将本地生成的 id\_rsa以及id\_rsa.pub这两个文件【一般在C盘用户名目录下的.ssh文件夹下】删除掉，然后再使用上述指令生成新的rsa密钥。

2.验证key：输入命令 `ssh -T git@github.com`，当出现下图中的提示时说明配置成功，此时再次执行git push 操作将本地项目推送到远程仓库。

```
@DESKTOP-KAGT9N8 MINGW64 /e/Postgraduate/Programming/Git/learngit (master)
$ ssh -T git@github.com
The authenticity of host 'github.com ([192.168.1.112])' can't be established.
RSA key fingerprint is SHA256:nThbg6kXUpJWGI7E1IGOCspRomTxdCARLviKw6E5SY8.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'github.com ([192.168.1.112])' (RSA) to the list of known hosts.
Hi [redacted]! You've successfully authenticated, but GitHub does not provide shell access.
```

## 三、执行git push命令时，如果出现错误：‘ssh: connect to host ssh.github.com port 443: Connection refused’

主要原因：域名解析被污染了，因为使用浏览器访问github.com网站是正常的

解决办法：— 究极无敌解决方法：可以尝试换个网络，如连接手机热点，本人尝试过是可行的，就不用下面那么麻烦了。

1.可以在 <https://ipaddress.com/website/ssh.github.com> 查找github.com域名的真实ip；

2.知道ip后，接下来测试下ssh是否可以连上该ip：`ssh -T -p 443 git@140.82.113.35`；

3.如果看到最后一行 Hi xxxx 就说明没问题，然后修改用户目录下的.ssh/config 文件，将 Hostname ssh.github.com 换成ip地址 Hostname 140.82.113.35 即可。

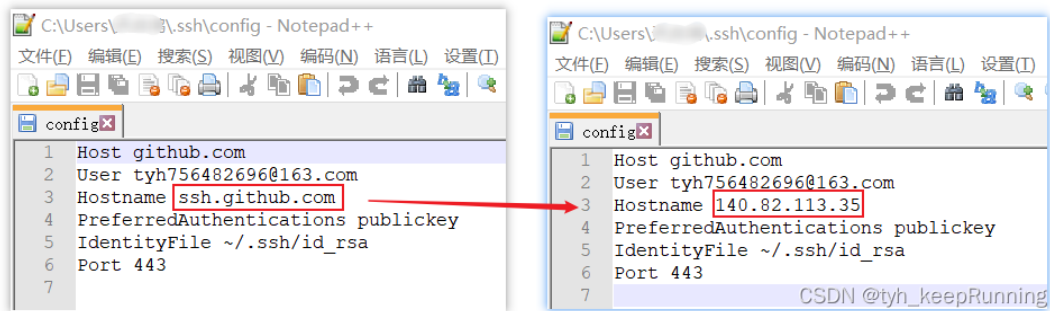
Hostname Summary	
Domain	github.com
Domain Label	github
IP Address	140.82.113.35
Web Server Location	us United States

Updated: Wed, 1 Jun 2022 01:32 GMT | Reviewed: Wed, 1 Jun, 2022 01:32 GMT | CSDN @tyh\_keepRunning

```
MINGW64:/d/TyhStudy/PythonCodes
$ ssh git@github.com
ssh: connect to host ssh.github.com port 443: Connection refused

$ ssh -T -p 443 git@140.82.113.35
The authenticity of host '[140.82.113.35]:443 ([140.82.113.35]:443)' can't be established.
ECDSA key fingerprint is SHA256:p2QAMXNIC1TJYWeIOttrVc98/R1BUFWu3/LiyKgUfQM.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '[140.82.113.35]:443' (ECDSA) to the list of known hosts.
Hi 1fh047375! You've successfully authenticated, but GitHub does not provide shell access.
```





#### 四、执行git status命令时，如果出现错误：'interactive rebase in progress; onto 6e82a25'

```
MINGW64:/d/TyhStudy/PythonCodes
@DESKTOP-TAG9JJC MINGW64 /d/TyhStudy/PythonCodes (master|REBASE 2/2)
$ git status
interactive rebase in progress; onto 6e82a25
Last commands done (2 commands done):
  pick 508716b Company 2022-3-3 周四
  pick 31b34eb Company, 2022-03-04
No commands remaining.
You are currently rebasing branch 'master' on '6e82a25'.
(all conflicts fixed: run "git rebase --continue")

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    modified:   .idea/workspace.xml

@DESKTOP-TAG9JJC MINGW64 /d/TyhStudy/PythonCodes (master|REBASE 2/2)
$ git commit -m '2022-3-8 修改'
[detached HEAD dd00894] 2022-3-8 修改
1 file changed, 65 insertions(+), 31 deletions(-)

@DESKTOP-TAG9JJC MINGW64 /d/TyhStudy/PythonCodes (master|REBASE 2/2)
$ git status
interactive rebase in progress; onto 6e82a25
```

主要原因：由于之前使用过 `git pull --rebase origin develop` 命令拉取代码，使用过 `git rebase` 执行代码覆盖，但是上一次进程还没有完成导致。查看git的提示，大概意思是你当前正在编辑的提交将要覆盖在 6e82a25 commitid 上

解决办法：

方法一(推荐)：使用 `git rebase --continue` 命令继续代码的提交，执行之后，解决当前的代码冲突之后重新提交直至没有rebase提示，就可以正常提交了；

方法二：使用 `git commit --amend` 命令修订当前的提交。

- 至此，本地项目便成功上传到GitHub~

