Git和GitHub

1. **git本地库初始化**

   `git init`

2. **设置用户签名（用来区分不同的用户）**

   项目级别的设置

   `git config user.name tom`

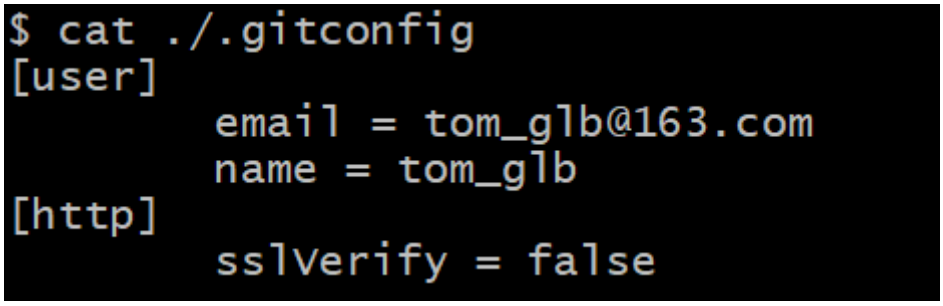
   `git config user.email tom@163.com`

   配置保存在./git/config文件中

   系统用户级别的设置

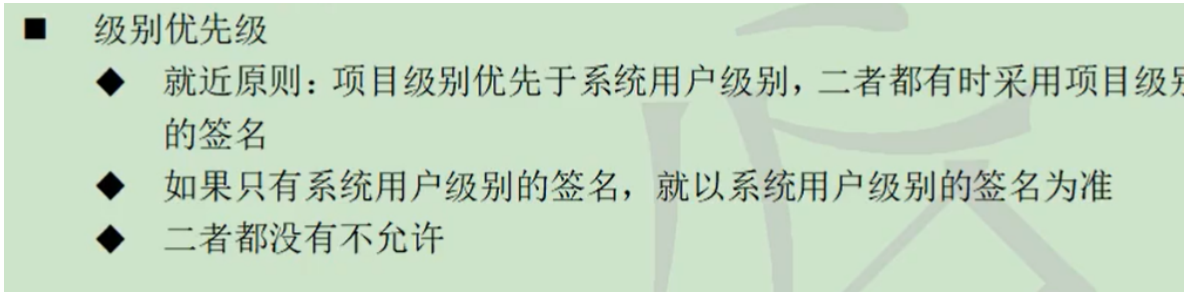
   `git config --global user.name tom_glb`

   `git config --global user.email tom_glb@163.com`



   配置保存在~/.gitconfig文件中

3. **两个配置的优先级**



4.

```
$ git commit new.txt
warning: LF will be replaced by CRLF in new.txt.
The file will have its original line endings in your working directory
[master (root-commit) 7536a32] My first commit.new file new.txt
 1 file changed, 4 insertions(+)
 create mode 100644 new.txt
```

```
rower@DESKTOP-9E1JUBA MINGW64 ~/Desktop/git (master)
$ git commit -m "My seconde commit,modify new.txt" new.txt
warning: LF will be replaced by CRLF in new.txt.
The file will have its original line endings in your working directory
[master f24c43f] My seconde commit,modify new.txt
 1 file changed, 1 insertion(+)
```

5.基本操作

5.1状态查看操作

git status

查看工作区、暂存区状态

5.2添加操作

git add [file name]

将工作区**新建/修改**的文件添加到暂存区

1. **提交操作**

   git commit -m "备注信息" [file name]

   将暂存区的文件提交到本地库

2. **查看历史提交**

   git log

```
$ git log
commit f24c43fd4d440e8be68395df2b8b0e24d8a57195 (HEAD -> master)
Author: tom <tom@163.com>
Date:   Wed Jul 22 20:44:38 2020 +0800

    My seconde commit,modify new.txt

commit 7536a3237e7f0553f70b98a42bda72073134db65
Author: tom <tom@163.com>
Date:   Wed Jul 22 20:35:56 2020 +0800

    My first commit.new file new.txt
```

   git log --pretty=oneline

```
$ git log --pretty=oneline
f24c43fd4d440e8be68395df2b8b0e24d8a57195 (HEAD -> master) My seconde commit,modify new.txt
7536a3237e7f0553f70b98a42bda72073134db65 My first commit.new file new.txt
```

   git log --oneline

```
$ git log --oneline
f24c43f (HEAD -> master) My seconde commit,modify new.txt
7536a32 My first commit.new file new.txt
```

   git reflog

```
$ git reflog
f24c43f (HEAD -> master) HEAD@{0}: commit: My seconde commit,modify new.txt
7536a32 HEAD@{1}: commit (initial): My first commit.new file new.txt
```

   HEAD@{移动到当前版本需要几步}

5.前进后退

- [ ] 使用索引值操作

  git reset --hard 7536a32

- [ ] 使用^，只能后退

  git reset --hard HEAD^

  注：一个^表示后退一步，n个^表示后退n步

- [ ] 使用~：只能后退

  git reset --hard HEAD~n

  表示后退n步

6.reset命令的三个参数对比

- [ ] --soft参数

  仅在本地库移动HEAD指针

- [ ] --mixed参数

  在本地库移动HEAD指针

  重置暂存区

- [ ] --hard参数

  在本地库移动HEAD指针

  重置暂存区

  重置工作区

7.删除文件并找回

前提：删除文件前，文件已经提交到本地库

操作：git reset --hard[指针位置]

删除操作已经提交到本地库：指针位置指向历史记录

删除操作没有提交到本地库：指针位置使用HEAD

8.比较文件差异

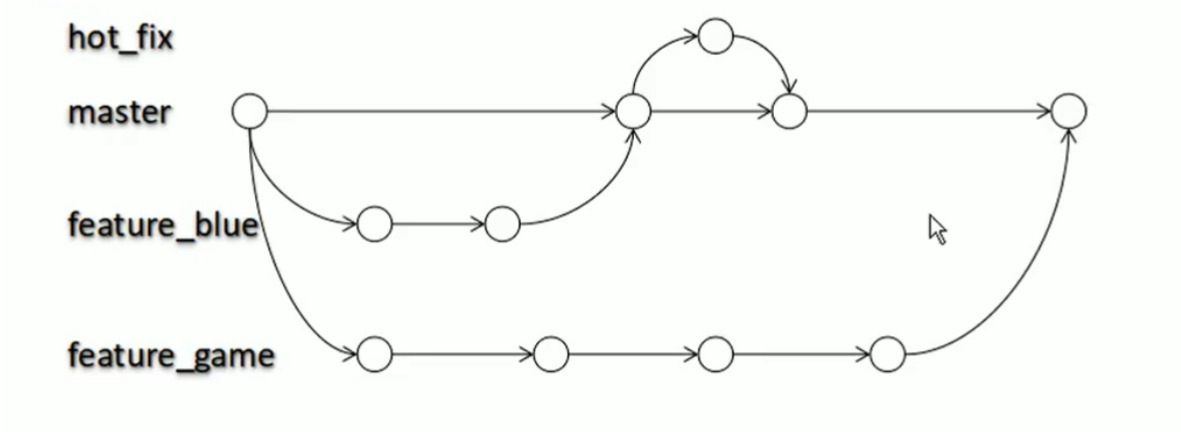
git diff[文件名]

- [ ] 将工作区中的文件和暂存区进行比较

git diff[本地库中历史版本]【文件名】

- [ ] 将工作区中的文件和本地库历史记录比较
- [ ] 不带文件名，比较多个文件

6.分支介绍



6.1分支的好处

同时并行推进多个功能开发，提高开发效率

各个分支在开发过程中，如果一个分支开发失败，不会对其他分支产生影响。

6.2分支的操作

- [ ] 创建分支

  git branch [分支名]

- [ ] 查看分支

  git branch -v

- [ ] 切换分支

  git checkout [分支名]

- [ ] 合并分支

  第一步：切换到要接受修改的分支（master）

  git checkout [分支名]

  第二步：合并分支

  git merge [分支名]

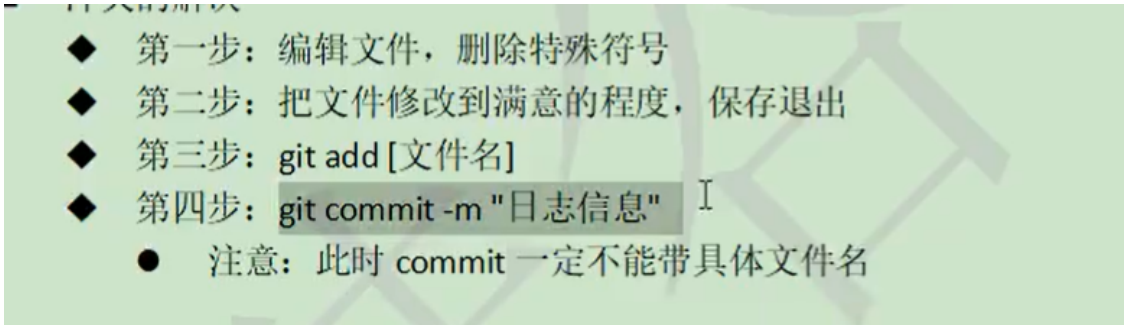
  bv

- [ ] 解决冲突

  冲突出现

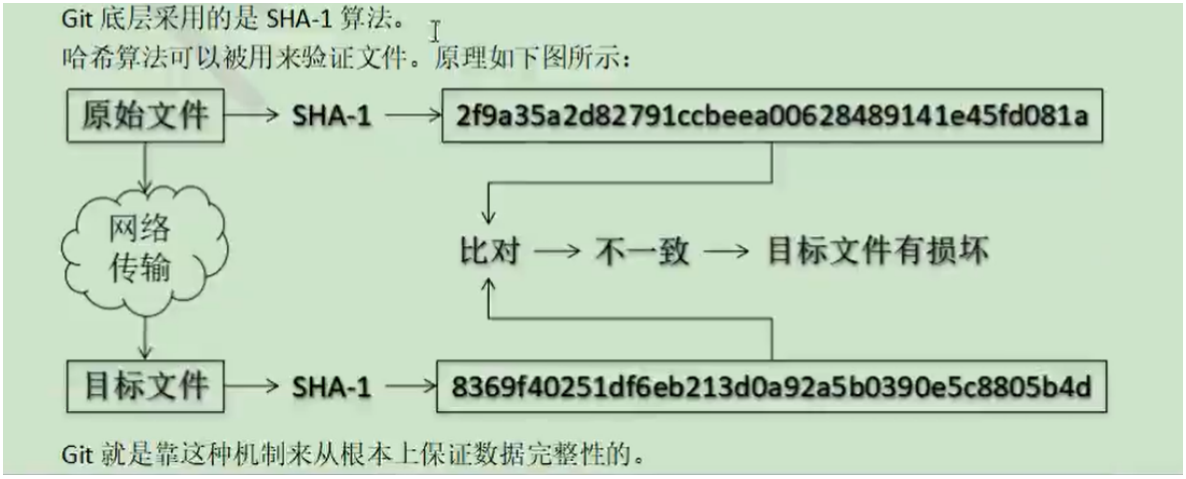
```
rower@DESKTOP-9E1JUBA MINGW64 ~/Desktop/git (hot_fix)
$ git merge master
Auto-merging a.txt
CONFLICT (content): Merge conflict in a.txt
Automatic merge failed; fix conflicts and then commit the result.
```

  解决冲突



7.git的原理

7.1



8.本地库到github远程库

8.1本地库push文件到远程库

1. 复制远程仓库的地址
2. git remote add origin https://github.com/rowerdog/testgithub2020.git

   给远程仓库起的别名
3. 提交到远程库

   git push origin master

8.2团队协作

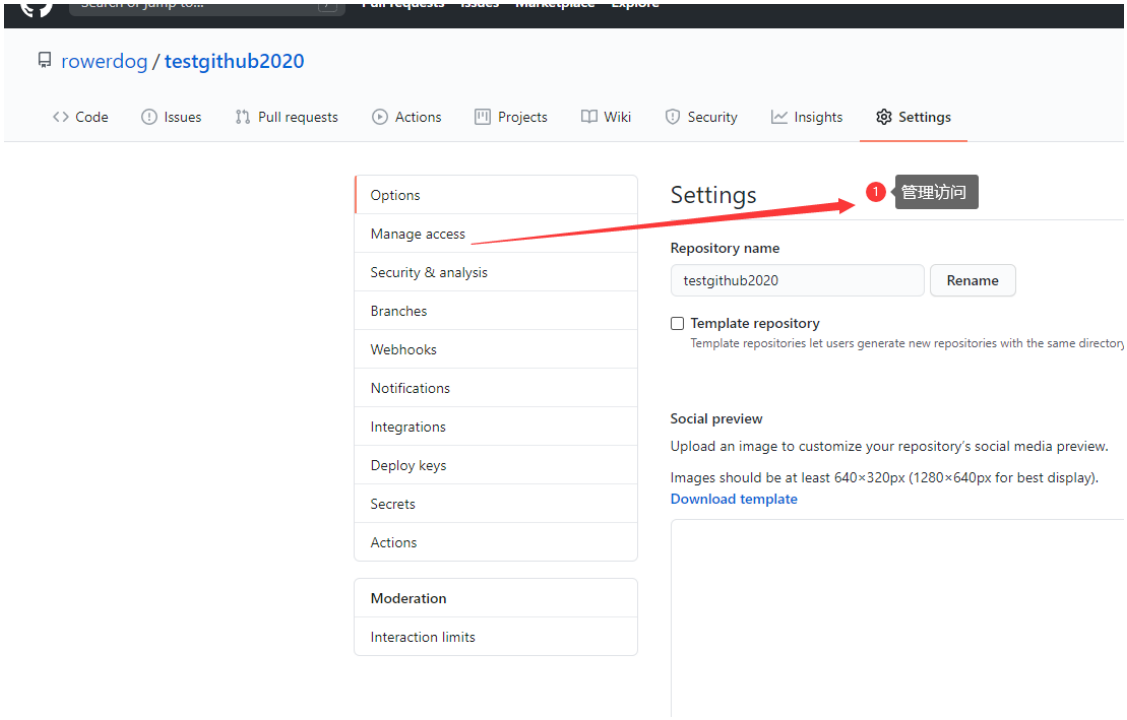
1.在文件夹中clone远程仓库的地址

git clone [远程仓库的地址]

2.clone的作用

- 完整的把远程库下载到本地
- 创建origin远程库别名
- 初始化本地库

3.邀请别人加入团队



复制邀请链接发送给要邀请的人，被邀请人打开链接接受请求

4.拉取

- pull=fetch+merge
- git fetch [远程仓库地址别名] [远程分支名]
- git merge [远程仓库地址别名/远程分支名]
- git pull [远程仓库地址别名] [远程分支名]

8.3解决冲突