# 安装git

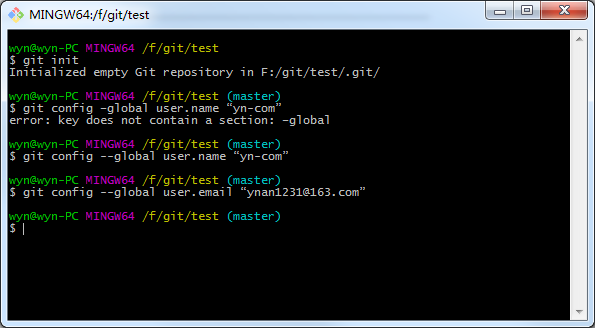
下载地址 <http://msysgit.github.io/>

默认安装即可.

完成后设置名字和邮箱,因为是分布式的,相当于自身名字.

git config --global user.name “yn-com”

git config --global user.email “[ynan1231@163.com](mailto:ynan1231@163.com)”



# 基础命令

## 创建版本库

**git init**

## 配置信息config

**git config --list 显示当前git配置**

**git config –e [--global] 编辑git配置文件**

**git config [--global] user.name “name” 设置提交代码时的用户信息**

**git config [--gloabl] user.email “email@email”**

## 创建文件touch

**touch readme.txt**

或者手动创建添加

## 添加文件到暂存区add

**git add readme.txt**

**git add .**  注意最后是点,添加所有文件

# 添加指定文件到暂存区

$ git add [file1] [file2] ...

# 添加指定目录到暂存区，包括子目录

$ git add [dir]

# 添加当前目录的所有文件到暂存区

$ git add .

# 添加每个变化前，都会要求确认

# 对于同一个文件的多处变化，可以实现分次提交

$ git add -p

# 删除工作区文件，并且将这次删除放入暂存区

$ git rm [file1] [file2] ...

# 停止追踪指定文件，但该文件会保留在工作区

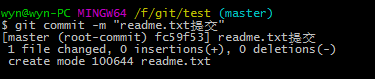
$ git rm --cached [file]

# 改名文件,重命名，并且将这个改名放入暂存区

$ git mv [file-original] [file-renamed]

## 提交文件commit

**git commit –m “readme.txt 提交”**



-m 注释内容

# 提交暂存区到仓库区

$ git commit -m [message]

# 提交暂存区的指定文件到仓库区

$ git commit [file1] [file2] ... -m [message]

# 提交工作区自上次commit之后的变化，直接到仓库区

$ git commit -a

# 提交时显示所有diff信息

$ git commit -v

# 使用一次新的commit，替代上一次提交

# 如果代码没有任何新变化，则用来改写上一次commit的提交信息

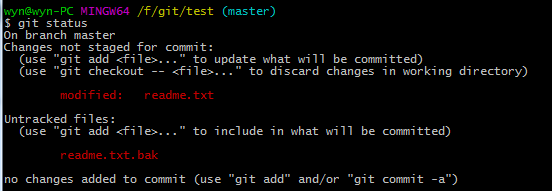
$ git commit --amend -m [message]

# 重做上一次commit，并包括指定文件的新变化

$ git commit --amend [file1] [file2] ...

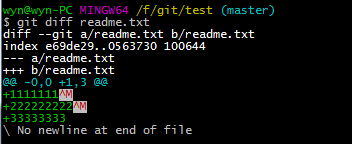
## 查看文件状态status

**git status**



## 对比更改diff

**git diff readme.txt 暂存区与工作区之间的差异**



增加了三行内容

提交修改要先git add ,然后git commit

比较两个版本的

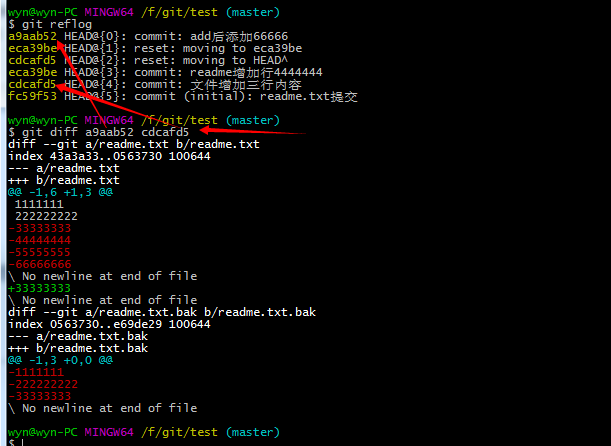
git diff --cached filename 显示暂存区与上一个commit的差异

git diff HEAD 显示工作区与当前分支最新commit之间的差异

git diff –shortstat “@{0 day ago}” 今天提交了多少代码



**git diff hash1 hash2 两次提交之间的差异**



## 查看历史log

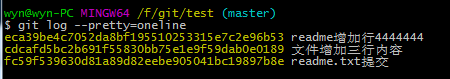
**git log**



其中,黄字为每次提交的版本号,顺序为从近到远.

美化显示过去5次提交

**git log -5 --pretty --oneline**



git log 查看commit的历史

**git show** <commit-hash-id>查看某次commit的修改内容

git log -p <filename> 显示指定文件相关的每一次diff

git log -p -2查看最近2次的更新内容

git log -S [keyword] 搜索提交内容包含关键字的commit

git log --stat 查看commit历史，以及每次commit发生变更的文件

git log HEAD --grep add 搜索提交说明里包含add的提交

git log HEAD –pretty=format:%s 格式化显示提交说明

git whatchanged filename 查看某个文件的提交人员、日期、版本号等记录信息，不包括修改详情。

git shortlog –sn 显示所有提交过的用户,按提交次数排序

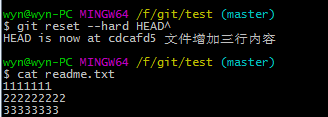
git blame filename 显示指定文件什么人在什么时间修改过

## 回退reset

**git reset --hard HEAD^** 工作区退回上个版本

**git reset --hard HEAD^^** 工作区退回上上个版本

**git reset --hard HEAD~100** 工作区退回前100个版本



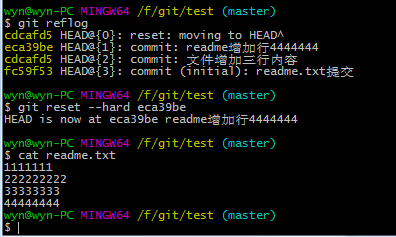
**git reset filename 暂存区退回到上一次commit,工作区不变**

**git reset --hard 重置工作区和暂存区**

**git reflog** 获得当前分支的最近几次提交

**git reset [commit] 指定当前分支为指定commit,重置暂存区,工作区不变**

**git reset --hard [commit]** 指定当前分支到commit,重置暂存区和工作区



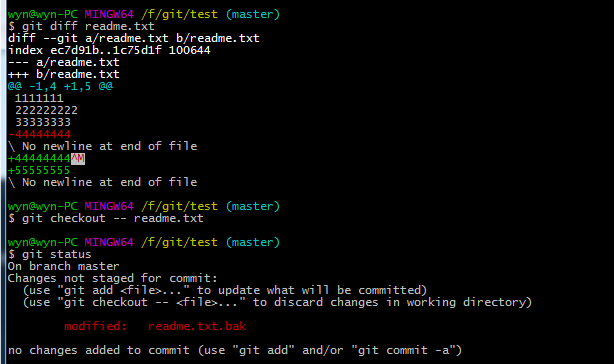
git reset –keep [commit] 重置当前HEAD为指定commit,但保持暂存区和工作区不变

git revert [commit] 新建一个commit,用来撤销指定commit,后者所有的变化都被前者抵消,应用到当前分支

## 还原

**git checkout -- readme.txt**

**--很重要,没有就变成切换分支**



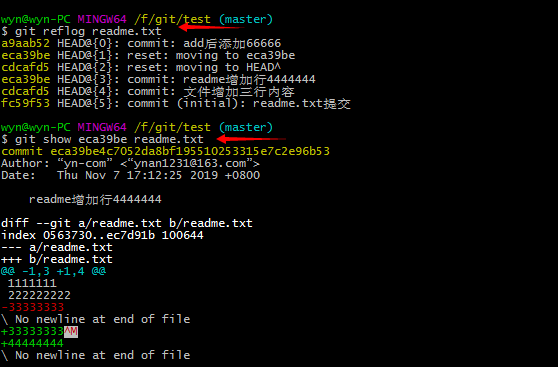
在文件未添加了一行55555555,执行命令后还原到版本库内容.

如果是已经add的文件,修改后还原,还是到add后的状态.



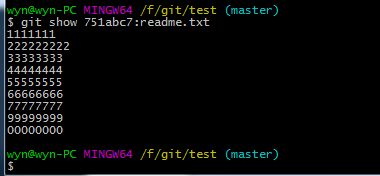
## 显示某版本更改的内容show

**git show [commit] 显示某次提交的元数据和内容变化**



git show --name-only [commit] 只显示变更的元数据,不显示内容变化

git show [commit]:[filename] 显示某次提交时文件的内容

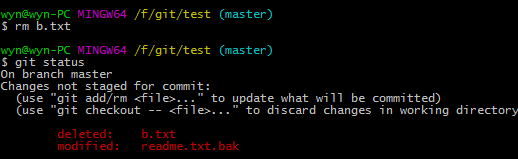


## 删除文件rm

rm filename 或直接删除

然后add、commit,就能删除版本库中的文件

Commit前回复可以用 git checkout – b.txt



# 分支命令

## 创建分支

**git branch name** 按照当前版本创建分支

**git checkout –b name** 创建并切换分支

**git branch name SHA值** 按照某个版本创建分支

# 新建一个分支，指向指定commit

$ git branch [branchname] [commit]

# 新建一个分支，与指定的远程分支建立追踪关系

$ git branch --track [branchname] [remote-branch]

在现有分支上与指定远程分支建立追踪关系

git branch --set-upstream dev3 origin/dev3

## 切换到分支checkout

**git checkout 新分支名**

**git checkout - 切换到上一工作区**

## 查看分支 branch

**git branch**

**# 列出所有本地分支**

**$ git branch**

**# 列出所有远程分支**

**$ git branch -r**

**# 列出所有本地分支和远程分支**

**$ git branch -a**

## 合并分支 merge

**git merge dev5**  合并指定分支到当前分支(将dev5分支合并到dev4)



git cherry-pick [commit] 选择一个版本的分支合并到当前分支

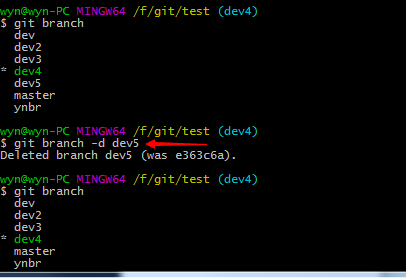
## 删除分支

git branch –d dev5 删除本地分支

删除远程分支

git push origin --delete [branch-name]

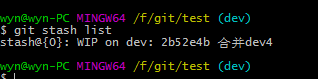
$ git branch -dr [remote/branch]



## 分支挂起

git stash

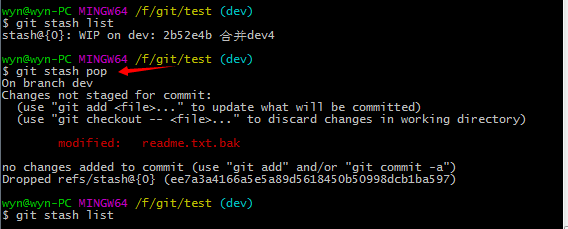
git stash list 挂起列表



**git stash apply** 恢复stash内容,但列表内容不删除

**git stash drop** 删除列表内的数据

**git stash pop** 恢复stash内容并删除列表内的数据

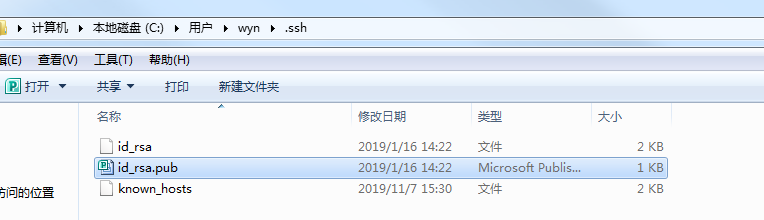


# 远程仓库

## 连接GitHub

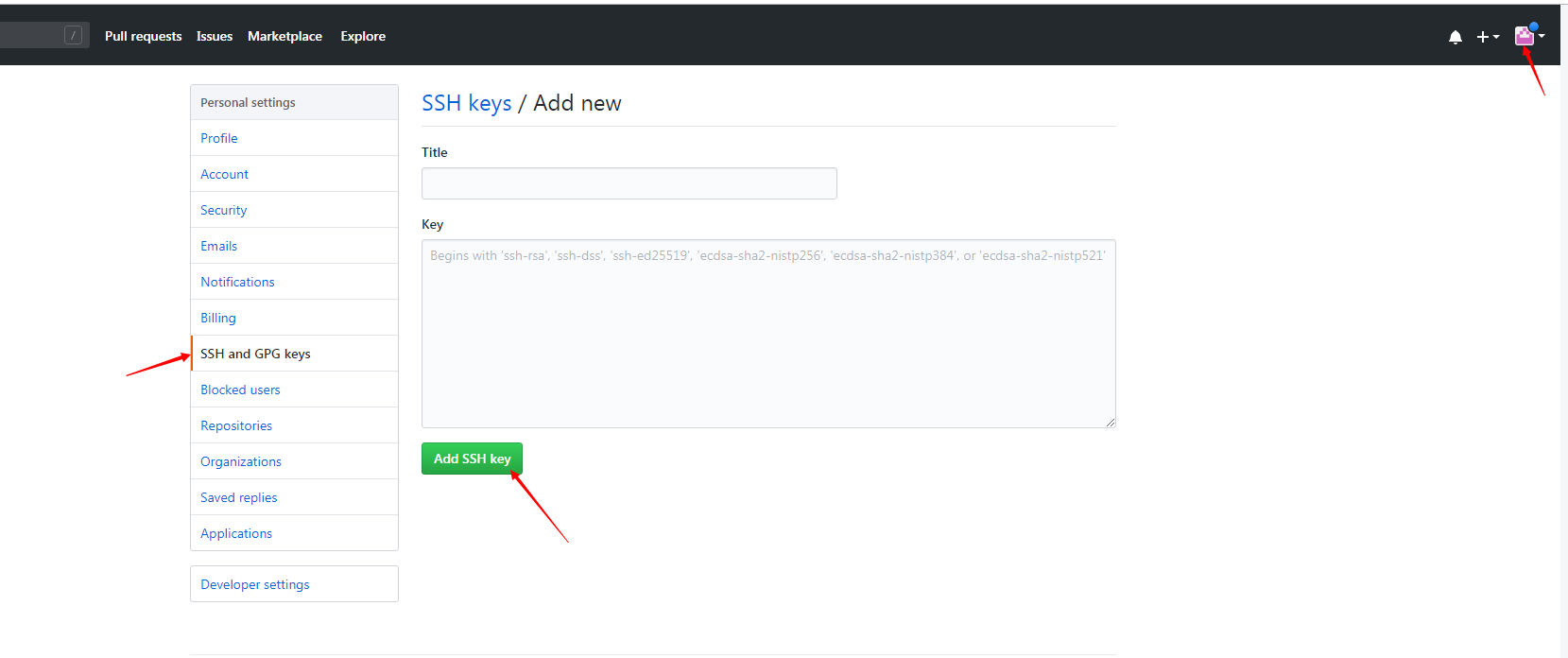
在用户主目录下,看是否有.ssh目录, 如果有，再看看这个目录下有没有id\_rsa和id\_rsa.pub这两个文件，如果有的话，直接跳过此如下命令，如果没有的话，打开命令行，输入如下命令：

ssh-keygen -t rsa –C “youremail@example.com”

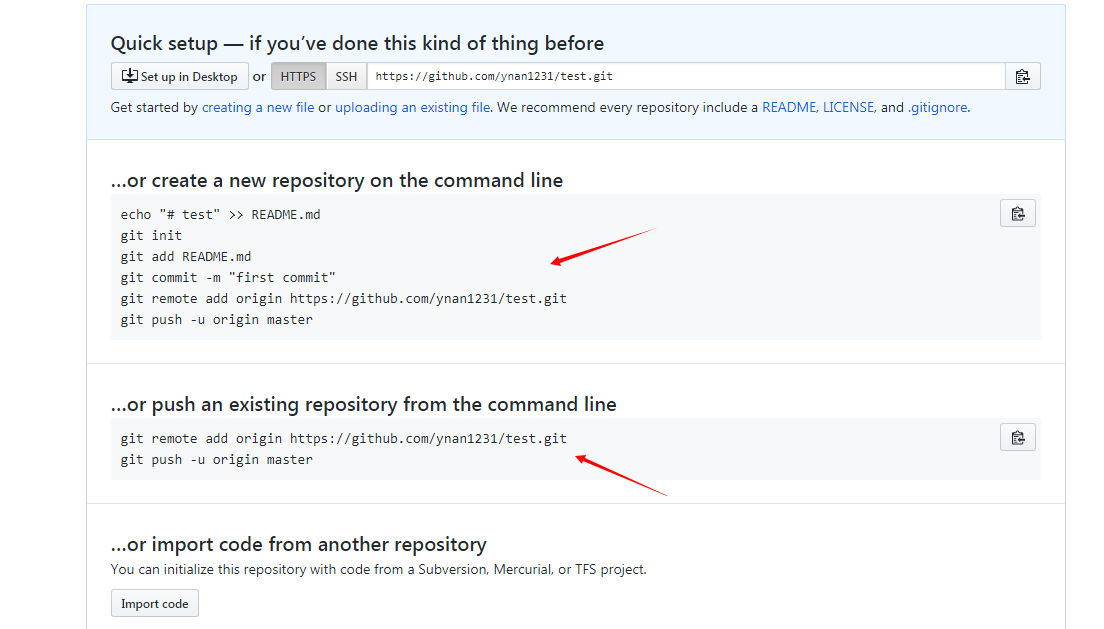


id\_rsa是私钥，不能泄露出去，id\_rsa.pub是公钥，可以放心地告诉任何人。

打开[https://github.com](https://github.com/), setting🡪SSH keys🡪add shh key ,填上任意title，在Key文本框里黏贴id\_rsa.pub文件的内容。



## 本地仓库放到远程仓库



在刚新建的F:\git\test目录执行以下语句.

git remote add origin https://github.com/ynan1231/test.git

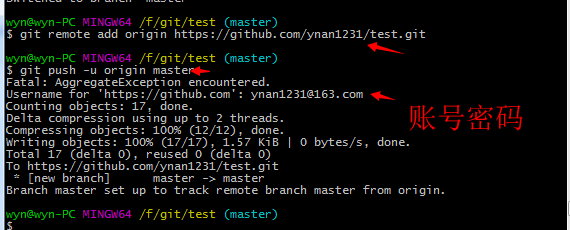
git push -u origin master

然后输入GitHub的账号密码即可.第一次推送需要-u进行关联,之后推送只需要

git push origin master

假如推送其他分支,用命令

git push origin dev



## 远程仓库下载到本地

git clone <https://github.com/ynan1231/myconfig.git>

## 远程命令

### 下载项目clone

$ git clone [url] 仅仅下载一个项目和整个历史代码,

### 添加远程仓库add

**git remote add [shortname] [url]**

git remote add origin https://github.com/ynan1231/test.git

git push -u origin master

然后输入GitHub的账号密码即可.第一次推送需要-u进行关联,之后推送只需要

git push origin master

假如推送其他分支,用命令

git push origin dev

与现有本地仓库建立联系

git branch --set-upstream dev3 origin/dev3

### 提交到远程仓库push

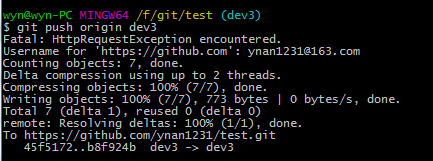
**$ git push [remote] [branch]**  remote可以是url,也可以是别名origin dev

**git push [remote] --force**  强行推送,忽略冲突

**git push [remote] --all** 推送所有分支到仓库

第一次推送 git push -u origin master

推送到远程分支dev3下,在分支dev3下,后面不需要再加本地dev3

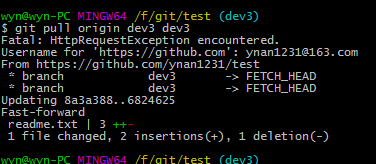


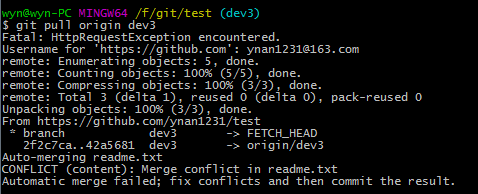
### 拉取远程仓库pull

git pull [remote] [branch] 取回远程仓库的变化，并与本地分支合并,

少用本命令

约等于 git fetch + git merge





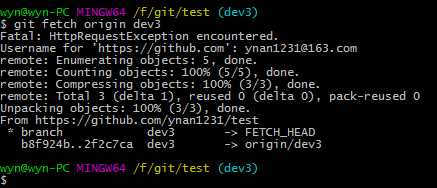
### 下载远程仓库的所有变动fetch

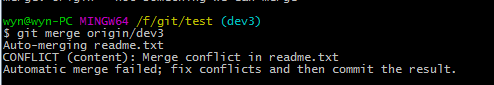
$ git fetch [remote]

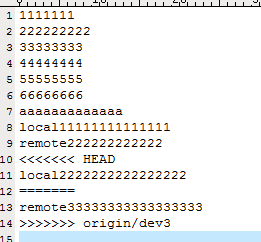
git fetch origin dev3下载远程仓库的所有变动到本地

+ git merge origin/dev3

替代git pull.







### 获取远程仓库信息

**$ git remote -v** 显示所有远程仓库

**git branch -a** 显示所有分支名称,包括远程

**$ git remote show [remote] 显示所有远程仓库所有信息**

### 对比与远程仓库的不同

**git diff origin/master master**