GIT的使用

[1. Git配置](#_Toc25466)

[1.1. 安装GitBash](#_Toc12752)

[1.2. Git初始化](#_Toc28445)

[2. Git命令](#_Toc16250)

[2.1. Git 起步](#_Toc29190)

[2.1.1. 三种状态](#_Toc31272)

[2.1.2. Git 工作流程](#_Toc8797)

[2.2. Git基础](#_Toc18464)

[2.2.1. Git 基础 - 获取 Git 仓库](#_Toc6947)

[2.2.2. Git 基础 - 修改更新到仓库](#_Toc14054)

[2.2.3. Git 基础 - 查看提交历史](#_Toc16431)

[2.2.4. Git 基础 - 撤消操作](#_Toc32205)

[2.3. Git分支](#_Toc10995)

[2.3.1. Git 分支 - 分支的查看、新建与合并](#_Toc23083)

[2.4. submodule 命令](#_Toc11898)

# Git配置

## 安装GitBash

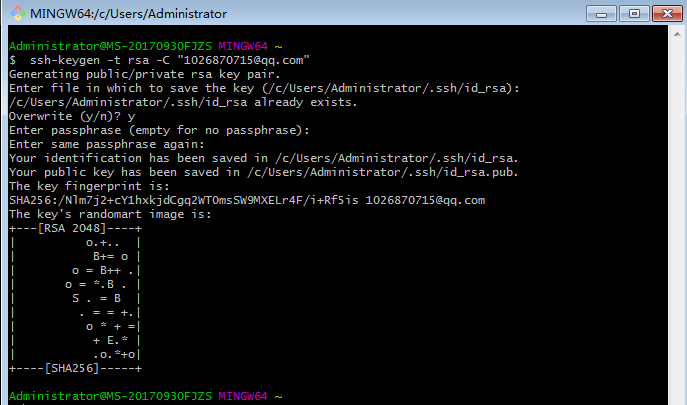
(已下载Git-2.13.3-64-bit.exe)下载地址<https://git-for-windows.github.io/>

GitBash安装成功，右击鼠标会出现git操作，选择git bash here可以打开git命令行（一个类linux操作界面）



## Git初始化

右击打开Git Bash Here命令窗，输入ssh-keygen -t rsa -C "你的邮箱账号"，一路回车(单用户)。



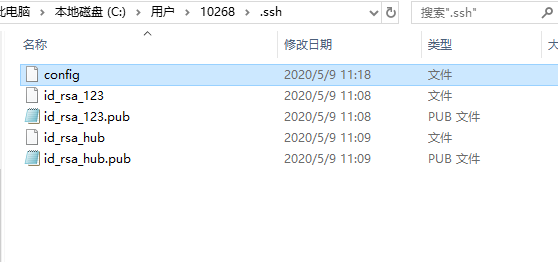
如需多个git地址账号时：

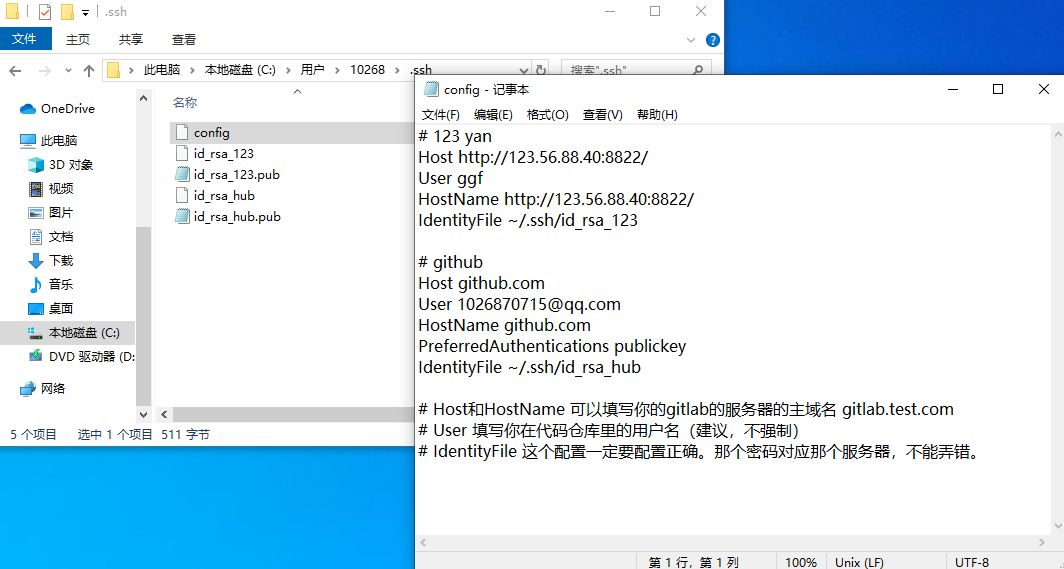
一、创建不同的ssh证书

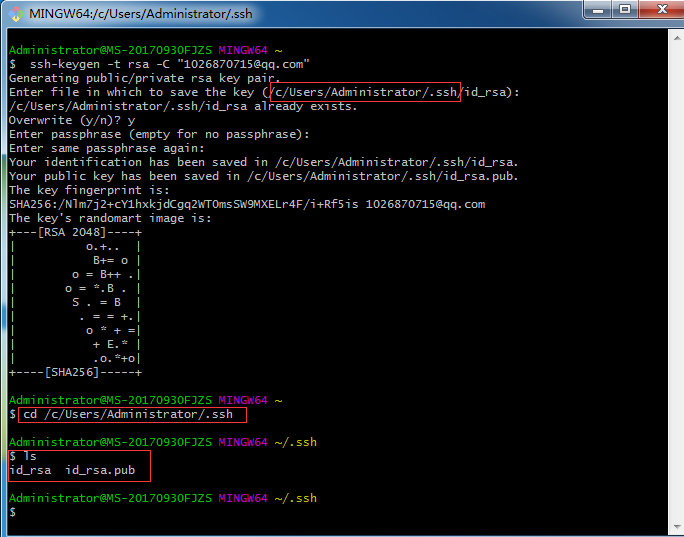


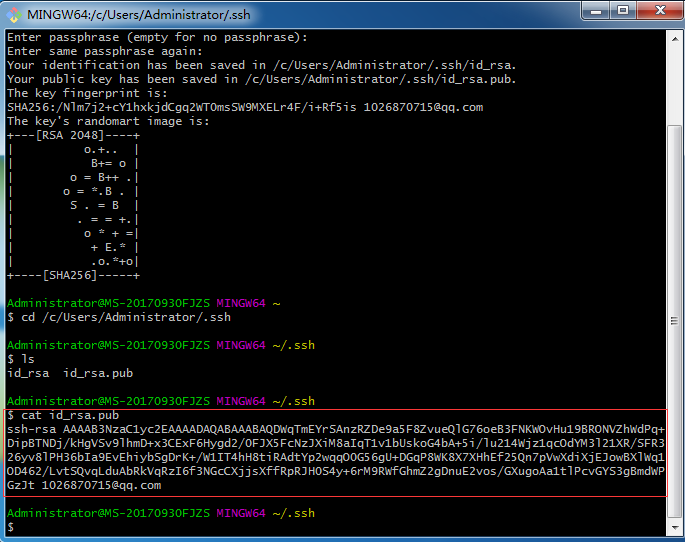


二、编辑配置文件

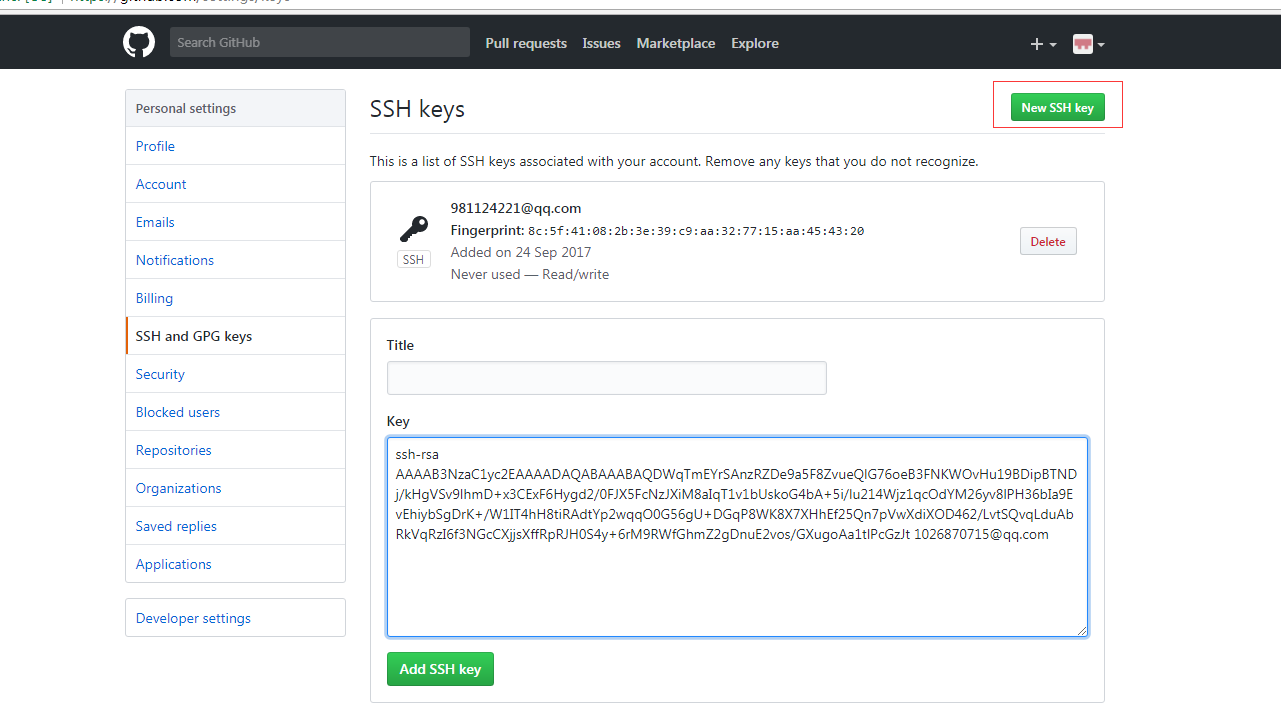








生成的公钥拷贝到github的账号settings中，点击NEW SSH key按钮



# Git命令

## Git 起步

### 三种状态

Git 项目的三个工作区域的概念：Git 仓库、工作目录以及暂存区域。



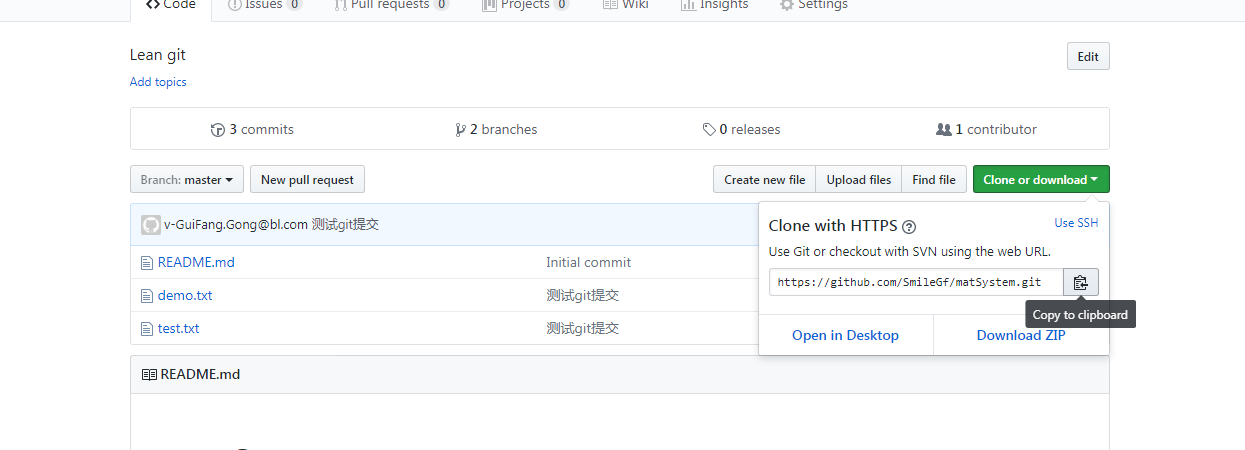
### Git 工作流程

1. 在工作目录中修改文件。
2. 暂存文件，将文件的快照放入暂存区域。
3. 提交更新，找到暂存区域的文件，将快照永久性存储到 Git 仓库目录。

如果 Git 目录中保存着的特定版本文件，就属于已提交状态。 如果作了修改并已放入暂存区域，就属于已暂存状态。 如果自上次取出后，作了修改但还没有放到暂存区域，就是已修改状态。

## Git基础

### Git 基础 - 获取 Git 仓库

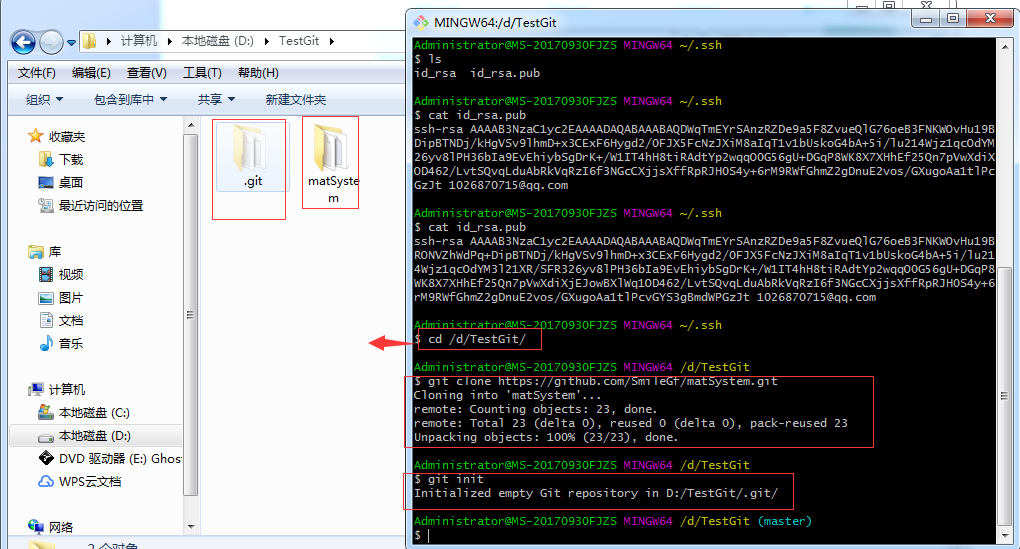


克隆git项目git clone [url]，比如：

$ git clone https://github.com/SmileGf/matSystem.git

克隆后本地TestGit是没有.git的子目录的，你需要执行以下命令初始化仓库：

$ git init



然后更新下来的iokblweb是没有子模块内容的，执行以下命令更新子模块内容：(没有子模块的项目不考虑)

$ git submodule update --recursive

### Git 基础 - 修改更新到仓库

检查当前文件状态

$ git status

暂存已修改文件

$ git add [. or 文件名]

在.gitignore中列出要忽略的文件

文件 .gitignore 的格式规范如下：

所有空行或者以 ＃ 开头的行都会被 Git 忽略。

可以使用标准的 glob 模式匹配。

匹配模式可以以（/）开头防止递归。

匹配模式可以以（/）结尾指定目录。

要忽略指定模式以外的文件或目录，可以在模式前加上惊叹号（!）取反。

$ cat .gitignore

提交更新（只提交暂存区的文件也就是git add过的文件，并且只提交到本地仓库而已）

$ git commit [-m ‘备注信息’]

当你忘记添加 .gitignore文件，不小心把一个很大的日志文件或一堆 .a 这样的编译生成文件添加到暂存区时，这一做法尤其有用。 为达到这一目的，使用 --cached 选项：

$ git rm --cached README

git rm 命令后面可以列出文件或者目录的名字，也可以使用 glob 模式。 比方说：

$ git rm log/\\*.log

注意到星号 \* 之前的反斜杠 \， 因为 Git 有它自己的文件模式扩展匹配方式，所以我们不用 shell 来帮忙展开。 此命令删除 log/ 目录下扩展名为 .log 的所有文件。 类似的比如：

$ git rm \\*~

该命令为删除以 ~ 结尾的所有文件。

推送到远程仓库

$ git push origin master

### Git 基础 - 查看提交历史

查看提交历史

$ git log

### Git 基础 - 撤消操作

撤销Git add操作

$ git reset HEAD <file> # 取消add操作并保留修改

撤销Git commit操作

$ git reset --soft <commit\_id> #可以回退到某个commit并保存之前的修改

$ git reset --hard <commit\_id> #回退到某个commit不保留之前的修改

撤销Git push操作

$ git revert <commit\_id>

## Git分支

### Git 分支 - 分支的查看、新建与合并

 查看远程分支

$ git branch -a

查看本地分支

$ git branch

创建分支

$ git branch test

把分支推到远程分支

$ git push origin test

切换分支到test

$ git checkout test

删除本地分支

$ git branch -d test

删除远程分支

$ git branch -r -d origin/test

$ git push origin :test

分支的合并

这个命令把分支"branchname"合并到了当前分支里面。

$ git merge branchname

## submodule 命令

submodule 添加

$ git submodule add [-b master ] [URL to Git repo] [local path]

Submodule内容更新

在父项目的目录下运行

$ git submodule foreach git checkout master

$ git submodule foreach git pull

在Submodule的目录下面更新

$ git pull

父工程指定的submodule版本更新

$ git submodule update --recursive

父工程更新submodule版本

$ git submodule update --remote