Git使用指南

1. **Git简史**
2. **Git介绍**

Git是一个开源的分布式版本控制系统，用以有效、高速的处理从很小到非常大的项目版本管理。

Git是Linus Torvalds为了帮助管理Linux内核开发而开发的一个开放源码的版本控制软件。

Git具有以下特性：

* 直接记录快照，而非差异比较
* 完全分布式，近乎所有的操作都是在本地执行
* 时刻保持数据完整性（SHA-1哈希值）
* 分支功能简单实用
* 更少的仓库污染

1. **集中式vs分布式**
2. 集中型：

对于版本控制，大家比较熟悉的是CVS和SVN，这些都属于集中化的版本控制系统CVCS，它能有效的解决不同系统上的开发者协同工作，不同的人都是直接面向中心库进行操作：



集中型版本控制，工作区直接跟版本库打交道，为两区结构：



A：从版本库中检出内容到工作区。

B：从工作区提交内容到版本库。

1. 分布式：

Git是属于分布式版本控制系统DVCS，特点是所有的仓库都是完整的镜像，假如版本控制服务器发生了故障，可以用任何一个本地仓库进行恢复。



Git分布式为三区结构：工作区—暂存区—版本库，跟二区的不同在中间加了一个暂存区。



工作区直接跟暂存区打交道，暂存区跟版本库打交道：

A：从工作区中添加内容到暂存区。

B：将暂存区的内容提交到版本库。

C：从版本库中检出内容到工作区。

1. **软件安装**
2. **Server端软件**

选定GitLab作为Git服务器（Linux），下载一键安装介质，进行程序安装及配置。

程序包：bitnami-gitlab-11.5.4-0-linux-x64-installer.run

1. **Client端软件**

程序包如下：

Git-2.20.1-64-bit.exe 下载地址：<https://git-scm.com/downloads>

TortoiseGit-2.7.0.0-64bit.msi下载地址：<https://tortoisegit.org/download/>

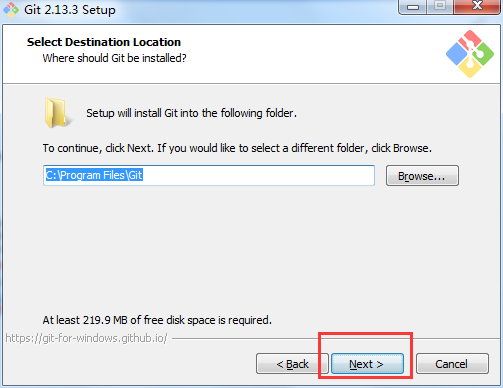
TortoiseGit-LanguagePack-2.7.0.0-64bit-zh\_CN.msi 汉化包

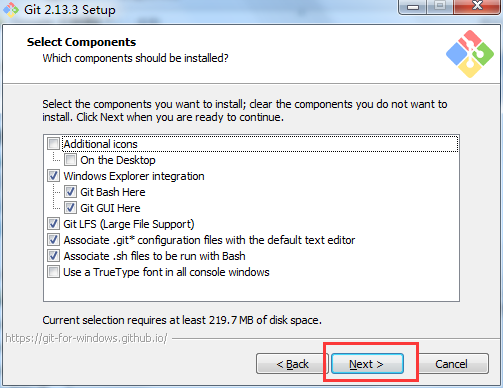
SourceTreeSetup-3.0.15.exe下载地址：

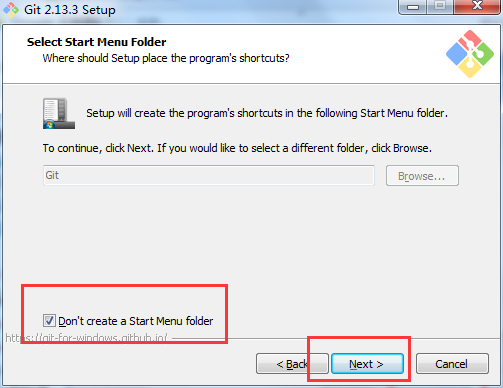
以上安装包在svn上都有备份，地址：https://10.133.194.170:8443/svn/bddj/8-工具文档/Tools/Git

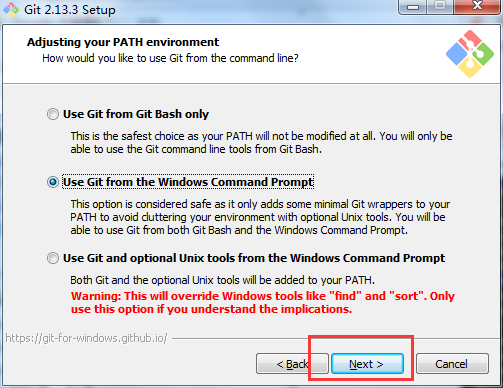
1. **Git客户端安装**

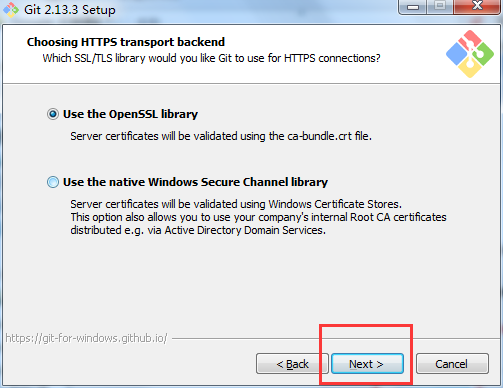


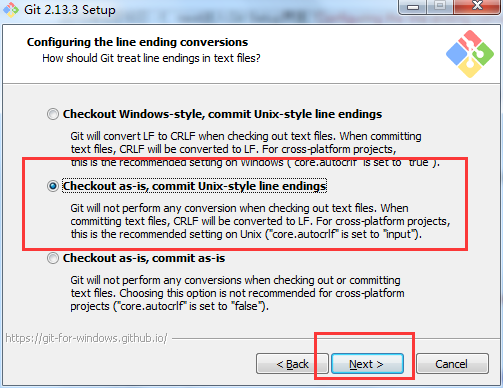


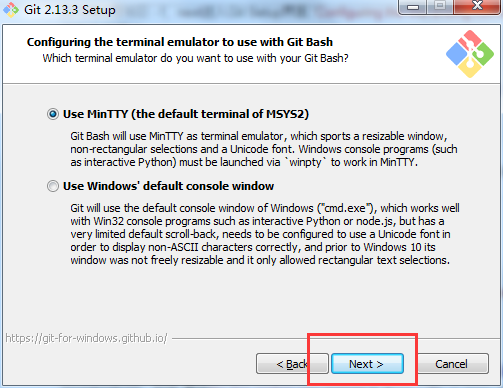


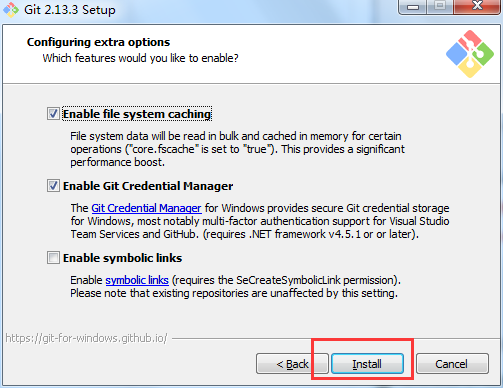


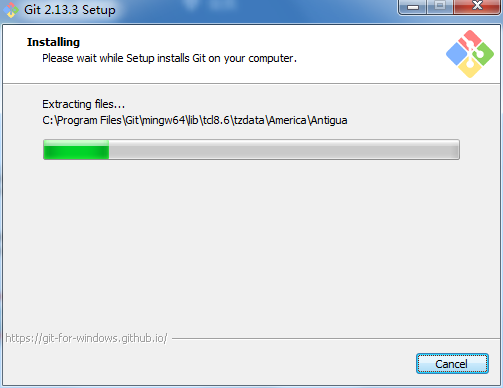


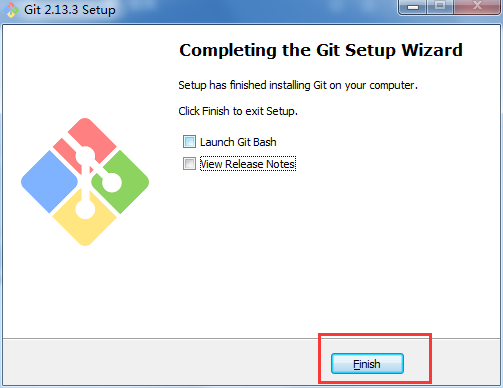




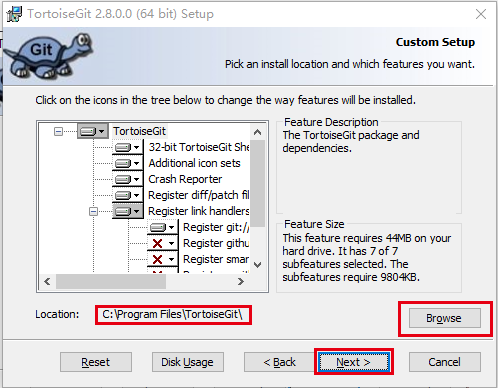
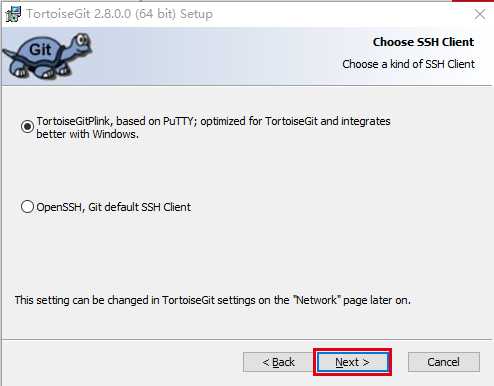
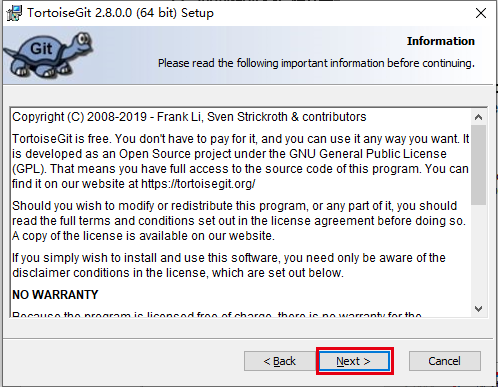
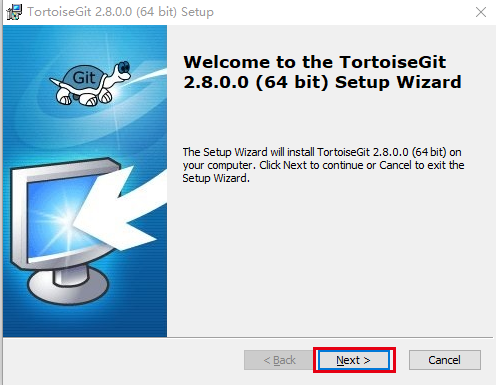




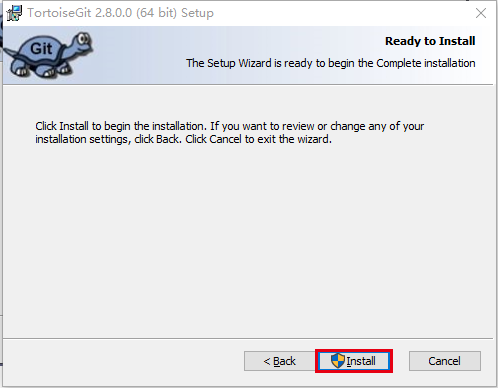




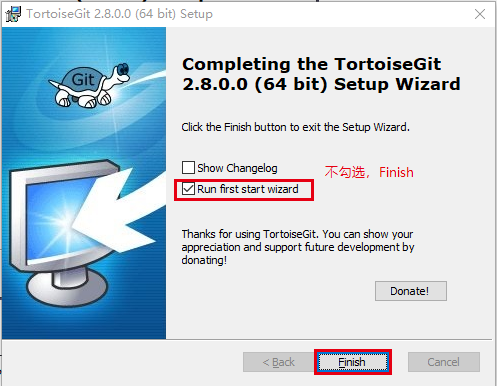
1. **TortoiseGit客户端安装**



如需更换安装路径，选择Browse选择自己电脑中的需要安装存放的路径Next

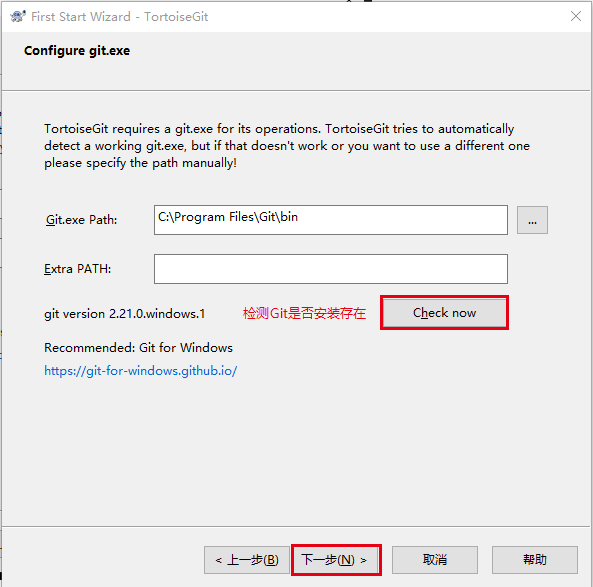
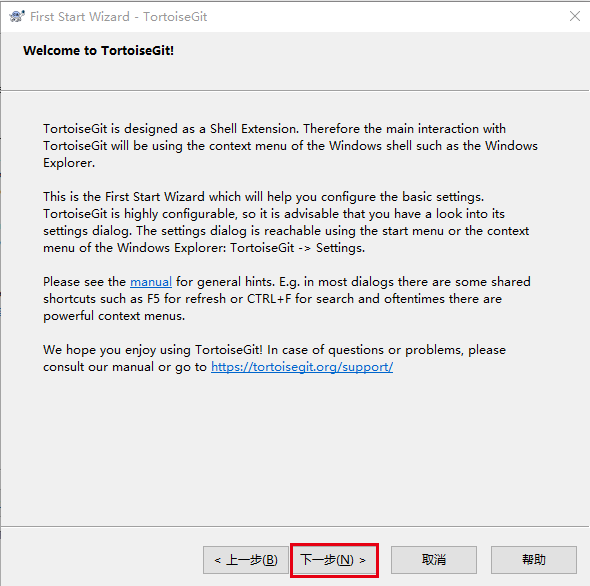
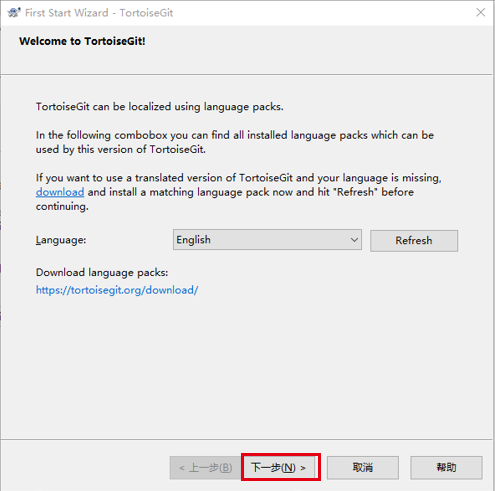


不多说Install，等待安装进度

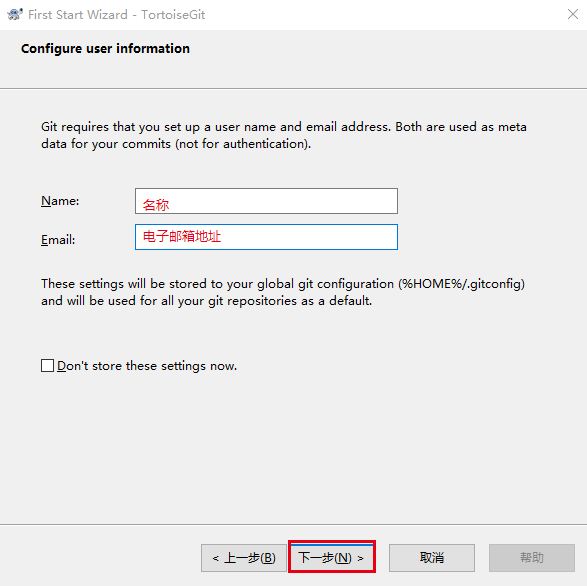


点击Finish，不运行开始向导直接完成即可

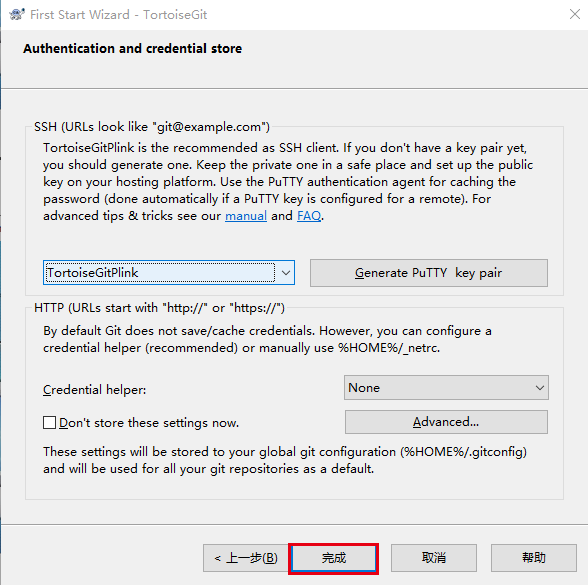
开始向导



检验Git是否安装成功，如果检测不成功，重新安装Git(3.1Git客户端的安装)



在上述框中填入要使用的Git的账号信息，下一步



点击完成，开始向导完成

1. **TortoiseGit-LanguagePack安装**

\*该安装包可选择进行安装，不使用汉化处理，依然可以使用上述安装Git的相关软件，可直接忽略该过程进行配置。



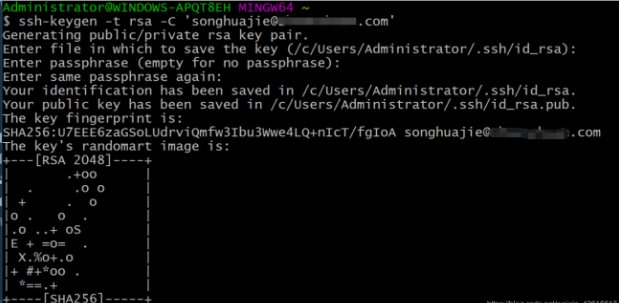
点击完成，TortoiseGit-LanguagePack安装结束。

1. **Git配置**
   1. **创建本地SSH KEY认证配置：**

1．Git Bash配置：

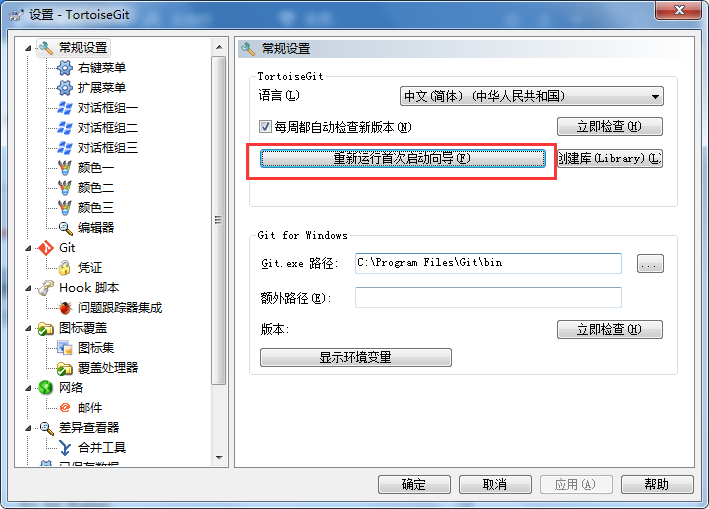
打开Git Bash命令行工具，执行下列命令生成一个密钥对，由于安装TortoiseGit客户端，我们也可以使用可视化界面进行密钥对的生成，如下图：

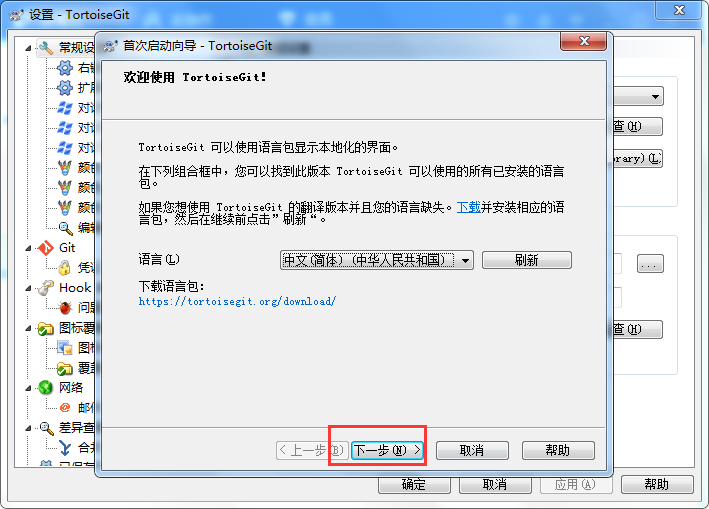
$ ssh-keygen -t rsa -C [your\_email@youremail.com](mailto:your_email@youremail.com)



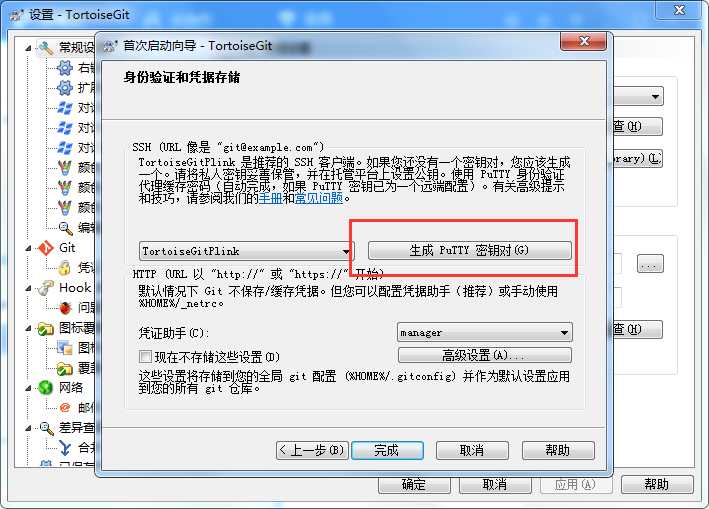
出现上述结果，说明在个人电脑中c：/Users/Administrator/.ssh/ 会生成两个密钥文件id\_rsa和id\_rsa.pub

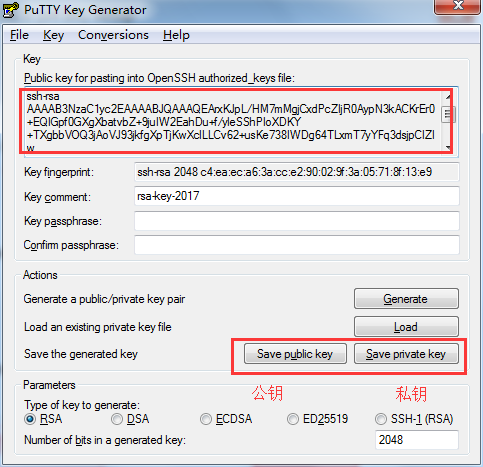
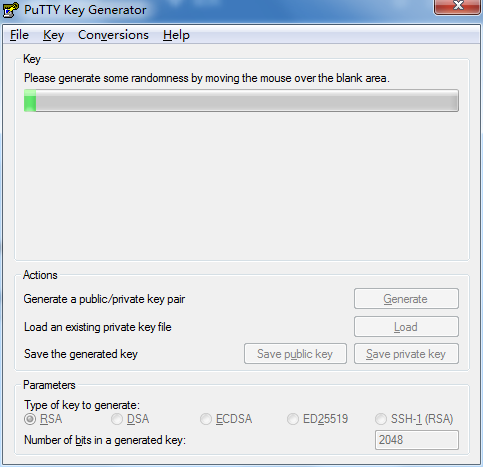
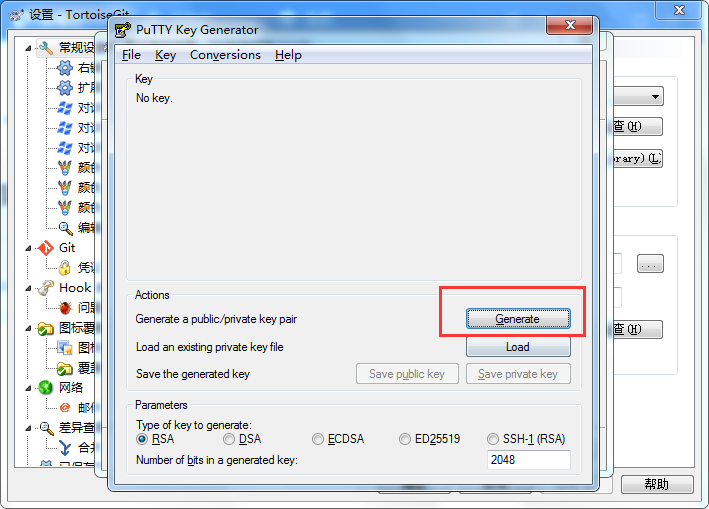
2．TortoiseGit客户端配置：











上述秘钥生成后，登录到GitLab，点击右上角你的用户头像，点击Edit Profile settings，点击SSH Keys，点击Add SSH Key，填写Title栏（Title栏中内容可任意填写，但为了方便使用，可以根据工作命名），复制用户目录下.ssh/id\_rsa.pub文件的内容到Key，点击Add Key。



* 1. **用户管理**

1．配置本地操作用户：

在使用Git前，需要配置个人本地用户名称和电子邮件地址，用户名和邮箱地址是本地git客户端的一个变量，不随git库而改变。每次commit都会用用户名和邮箱记录。

$ git config --global user.name "your name"

$ git config --global user.email [your\_email@youremail.com](mailto:your_email@youremail.com)

--your name为个人用户名

--[your\_email@youremail.com](mailto:your_email@youremail.com)为个人邮箱

2．查看用户名和邮箱地址：

$ git config user.name

$ git config user.email

3．修改用户名和邮箱地址：

$ git config --global user.name "username"

$ git config --global user.email "email"

1. **Git****基本操作**

由于TortoiseGit客户端是不需要命令，只需基础的配置，即可开始进行Git相关的使用，以下是通过Git Bash进行Git的命令操作：

* 1. **Git init – 创建和初始化仓库**

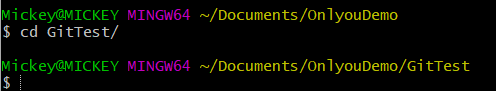
打开Git Bash，手动或在你需要的位置创建文件夹，此文件夹是Git中你存放代码或文件的本地仓库。

创建文件夹并进入文件夹：

$ mkdir “文件夹名称”

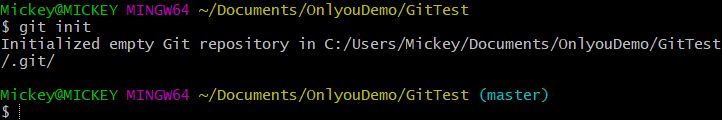
$ cd “文件夹名称”





初始化文件夹为本地仓库：

$ git init

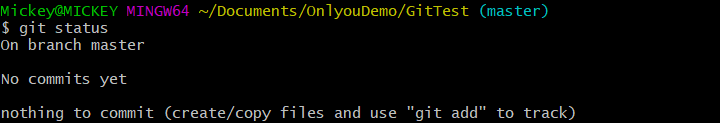


该命令执行完后会在当前目录生成一个 *.git* 目录。

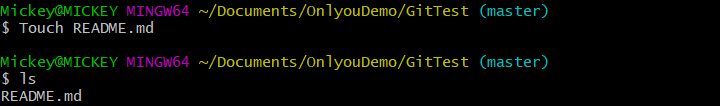
* 1. **Git status - 查看仓库状态**

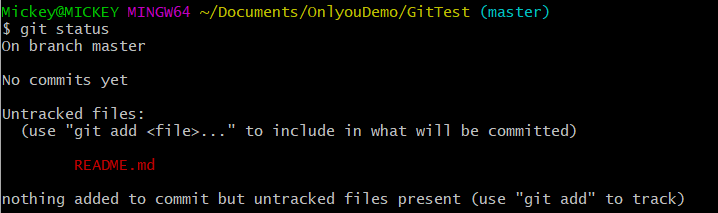
git status 显示Git仓库的状态，十分常用，请务必牢记。

$ git status



Touch README.md，查看status变化



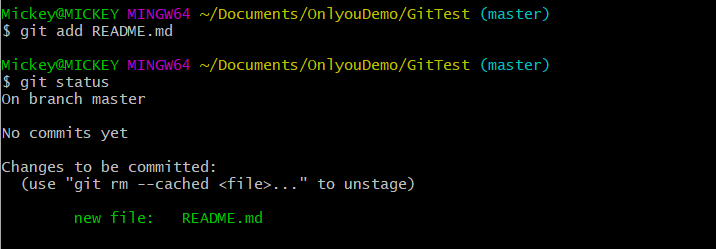


* 1. **Git add - 向暂存区中添加文件**

如果只是用Git仓库的工作树创建了文件，文件并不会被记录入Git仓库的版本管理对象当中。想让文件成为Git仓库的管理对象，就需要用git add 将其加入暂存区（Stage或者Index）中，暂存区是提交之前的一个临时区域。

$ git add . --添加所有文件

$ git add <file> --可反复多次使用，添加指定的多个文件



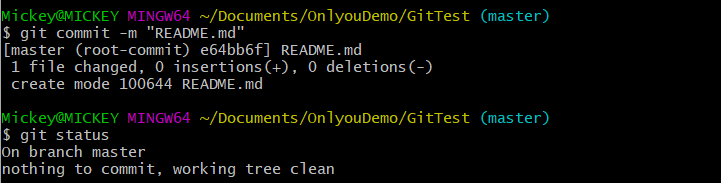
查看状态，显示为Changes to be committed。

* 1. **Git commit - 保存仓库的历史记录**

Git commit将当前暂存区中的文件实际保存到仓库的历史记录中，通过这些记录，就可以在工作树中复原文件。

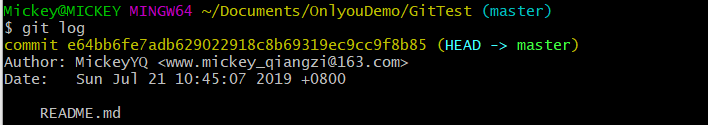
提交并查看当前提交后的状态：

$ git commit -m "注释" --把暂存区内容提交到本地仓库

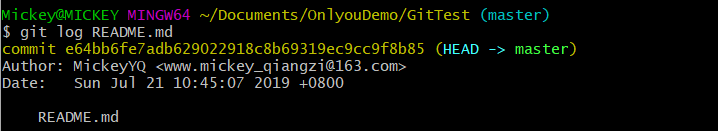


* 1. **Git log - 查看提交日志**

$ git log



$ git log “文件名称”

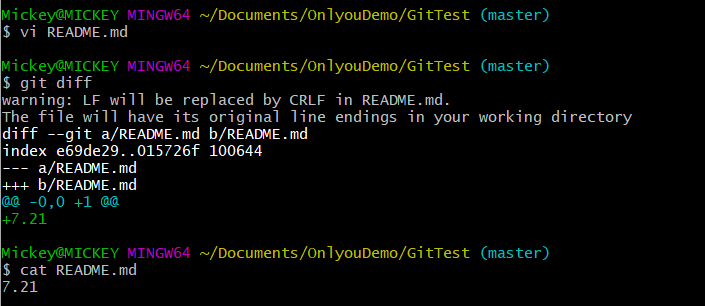


$ git log –oneline --查看历史记录的简洁的版本

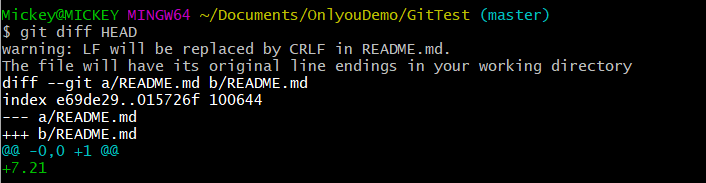
$ git log --oneline –graph --开启了拓扑图

* 1. **Git diff - 查看更改前后的差别**

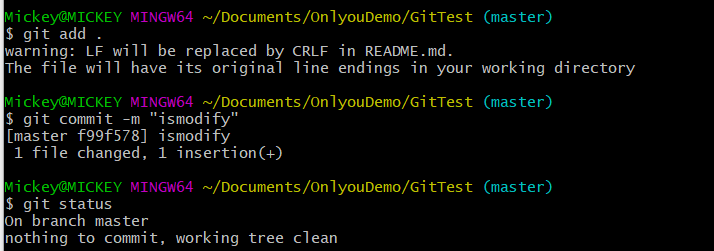
首先在刚才的文件中添加内容并查看差别：



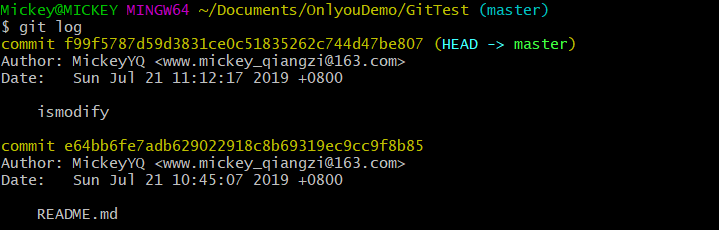
$ git diff HEAD



提交修改以上文件查看差别：



查看日志：



* 尚未缓存的改动：git diff
* 查看已缓存的改动： git diff –cached
* 查看已缓存的与未缓存的所有改动：git diff HEAD
* 显示摘要而非整个 diff：git diff --stat

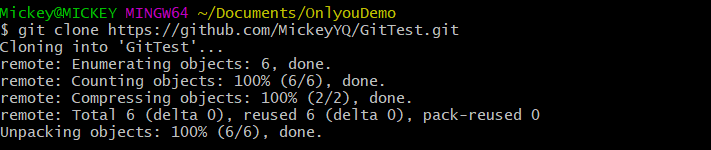
注：在Git Bash中大多数的linux命令都能使用

1. **Git 从远程仓库获取**

git clone支持多种协议，除了HTTP(s)以外，还支持SSH、Git、本地文件协议等,如下图：



$ git clone [url] –-[url] 为你想要复制的项目



1. **Git分支管理**
   1. **master分支与develop分支的区别**

**Master分支：**

Master为时常保持可发布状态的分支，不允许开发者直接对master分支的代码进行修改和提交。

其他分支的开发工作进展到可以发布的程度后，将会与master分支进行合并，而且这一合并只在发布成品是进行，发布时会附加包含版本编号的Git标签（Tag）。

**Develop分支：**

Develop分支是开发过程中的代码中心分支，与master分支一样，这个分支也不允许开发者直接进行修改和提交。

开发者要以develop分支为起点新建feature分支，在feature分支中进行新功能的开发或者代码的修改。也就是说，develop分支维持着开发过程中的最新源代码，以便程序员创建feature分支进行自己的工作。

* 1. **分支操作**

**查看分支命令：**

$ git branch

当你执行 git init 的时候，缺省情况下 Git 就会为你创建 master 分支，没有参数时，git branch 会列出本地的分支。

**创建分支命令：**

$ git branch (branchname) –-branchname 为分支名称

**切换分支命令:**

$ git checkout (branchname)

**创建新分支并切换：**

$ git checkout -b (branchname)

**合并分支命令:**

$ git merge (branchname)

--将分支上的内容合并，首先先切回master分支，然后再执行该命令

**删除分支命令：**

$ git branch -d (branchname)

1. **Git 常用命令速查表**

