

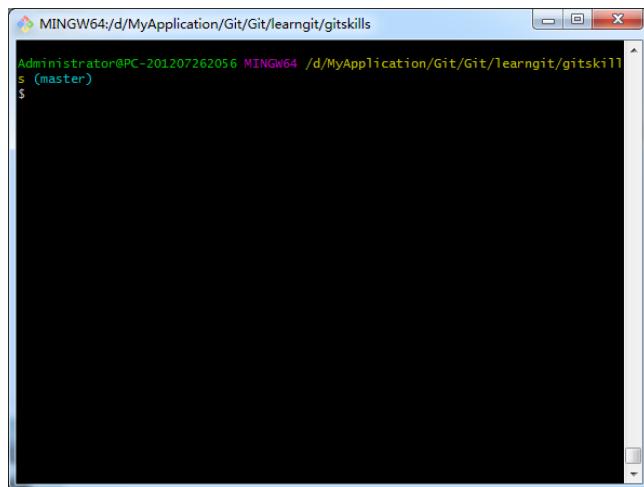
# Git

一款高端大气上档次的分布式版本控制系统

## 安装Git(Win)

在Windows中安装Git：直接从官网上下载安装程序，默认点击下一步即可。

安装完成后点击安装目录中的git bash会弹出一个类似dos的命令框，然后只需在命令框内配置用户名和邮箱即可完成安装！



配置用户名和邮箱：

```
$git config --global user.name "nickName"
```

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

## 创建仓库

第一步：创建一个空目录

注意调整创建空目录的位置，默认位置有可能会在c盘，需要使用cd命令调整到合适的位置创建，或使用绝对路径创建。

```
mkdir learnt  -----建目录learngt
```

第二步：在目录中生成为Git仓库

使用cd命令将当前位置调整到 learnt 目录下 键入 git init 完成仓库创建

\*本地可以创建多个仓库，只需在安装目录先创建一个空文件夹，然后在该文件夹下执行git init即可完成创建

## 添加文件到仓库

要添加文件到Git仓库，首先要将要添加的文件放到生成仓库的目录下（learngt）然后进行以下操作

第一步：使用命令git add 将文件从工作区添加至仓库暂存区

```
$git add test.txt
```

执行上面命令后无任何反应，说明文件添加成功。

第二步：使用git commit将文件有暂存区提交到你所操作的分支上（仓库）

\$ git commit -m "提交说明"

提交说明尽量输入一些有意义的内容，方便以后追踪。git commit会提交当前分支暂存区的所有文件，即可以一次提交多个文件。git add一次只能添加一个文件。

## 版本回退

可以将文件还原到任意一次上传是的状态

1. 显示提交日志(git log)

\$git log:显示所有提交日志的详细信息。（仅显示当前版本之前的日志信息，不显示已退回

的日志信息）

\$git log --pretty=oneline:显示所有提交日志的简要信息。（仅显示当前版本之前的日志信

息，不显示已退回的日志信息）

\$ git redlog :显示所有提交日志的简要信息，包括已退回的版本日志信息。

2. 退回版本(git reset)

1). \$ git reset --hard [HEAD^] :将仓库退回到上一次上传的版本。

HEAD^:表示上一版本。

HEAD^^:表示上上一版本（退回两个版本）。

HEAD~n:表示上n个版本（退回n个版本）。

2). \$ git reset --hard [commit id]:将仓库退回到指定版本

commit id : 表示每次上传对应的16进制编号，是唯一的（一般退回操作是输入前7位，

该方法既可以实现退回过去版本，也可以实现回到退回之前的版本，只要有版本编号，

可以回到任意 版本）

## 工作区与暂存区

1. 工作区（Working Directory）

在我们电脑本地操作的文件夹（learngit）就是工作区。

2. 版本库（Git仓库）

在工作区的一个隐藏文件夹 .git 就是Git的版本库。版本库分为两个部分，一个是暂存区还有

一个是Git版本库的分支（master），在Git版本库中可以创建多个分支。

### 3. 提交文件流程

第一步：使用 git add 把工作区中修改的文件添加至暂存区，可以添加多次

第二步：使用 git commit 将暂存区的所有文件提交至Git分支上

### 4. Git 的管理内容

在Git系统中所管理的并不是文件而是修改，每次提交上传至分支上时，只是将你本次修改的

内容进行了上传而不是将整个文件上传，这样大大提高了文件上传的效率。

## 撤销修改

1. 撤销工作区的修改（仅修改了工作区的源文件到并未添加至暂存区）

\$ git checkout -- test.txt : 将工作区的文件退回到与Git仓库相同的状态（最后一次上传的状态）

2. 撤回暂存区的修改（工作区修改的文件已添加至暂存区（git add）操作但是并未提交至Git

仓库的分支上（git commit））

\$ git reset HEAD test.txt : 将暂存区的内容撤销，重新放回工作区，然后使用（1）撤销工作

区的修改。

3. 撤回版本库的修改（工作区的修改以提交至暂存区并提交到了Git版本库，但并未推送至远程）

使用版本退回，退回到上一个版本即可 \$ git reset --hard [HEAD^]

## 删除文件

\$ rm test.txt : 删除没用的test.txt文件到工作区。

一般文件删除后会有两种操作

1. 文件确实要删除

1). git rm test.txt : 删除没用的test.txt文件到暂存区

2). git commit : 将删除内容提交到版本库分支，完成删除。

2. 文件错删

使用 git checkout -- test.txt 把误删的文件撤回工作区即可

## 创建远程仓库

## 1. 创建SSH Key

在C盘用户目录下，看看有没有 .ssh 目录，如果没有使用 \$ ssh-keygen -t -C "email" 命令添

加。在 .ssh 文件夹内会有 id\_rsa 和 id\_rsa.pub 两个文件。id\_rsa 是私钥，不要泄  
露，

id\_rsa.pub 是公钥

## 2. 登录 GitHub 将 id\_rsa 和 id\_rsa.pub 的公钥内容添加至 SSH Keys

The screenshot shows the GitHub profile settings page. On the left, there's a sidebar with options like Personal settings, Profile, Account, Emails, Notifications, Billing, Security, Blocked users, Repositories, and Organizations. The 'SSH and GPG keys' option is highlighted with a red arrow and labeled '第二步'. In the main content area, there's a section titled 'SSH keys' with a red arrow pointing to it and labeled '第三步'. It lists an SSH key named 'my ssh key' with a fingerprint of '1c:54:f3:97:ec:3c:f1:54:bc:11:28:e9:f9:f9:01:30'. Below this is a section for 'GPG keys' which says 'There are no GPG keys associated with your account.' A red arrow points to the 'Settings' link in the top right corner of the page, labeled '第一步'.

## 3. 添加远程仓库

第一步：

The screenshot shows the GitHub homepage with a green overlay at the top containing the text 'Learn Git and GitHub without any code!'. Below the overlay, there's a message about using the Hello World guide to create a repository, start a branch, write comments, and open a pull request. There are two buttons: 'Read the guide' (green) and 'Start a project' (white). A red arrow points from the 'Read the guide' button towards the 'Repositories' section below.

The screenshot shows the GitHub repositories page. At the top, there are buttons for 'Browse activity' and 'Discover repositories'. Below that, there's a section for 'Discover interesting projects and people to populate your personal news feed.' A red arrow points from the 'Repositories you contribute to' section towards the 'Your repositories' section. In the 'Your repositories' section, there are four repositories listed and a green 'New repository' button. A red arrow points to this 'New repository' button.

第二步：

A repository contains all the files for your project, including the revision history.

Owner: mynamehzw / Repository name: learnkits ✓ 仓库名

Great repository names are short and memorable. Need inspiration? How about [turbo-octo-invention](#).

Description (optional): 仓库说明

Public: Anyone can see this repository. You choose who can commit. 公开仓库（免费）

Private: You choose who can see and commit to this repository. 私有仓库（收费）

Initialize this repository with a README: This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository. 在仓库创建README文件

Add .gitignore: None None | Add a license: None None | ⓘ

Create repository 创建

## 关联远程仓库

### 1. 关联仓库

在本地仓库下运行 & git remote add origin [远程仓库URL]

### 2. 克隆仓库

在本地仓库下运行 & git clone [远程仓库URL]

Quick setup — if you've done this kind of thing before

Set up in Desktop or  HTTPS  SSH [https://github.com/mynamehzw/frist\\_Git.git](https://github.com/mynamehzw/frist_Git.git) 远程仓库URL

We recommend every repository include a [README](#), [LICENSE](#), and [.gitignore](#).

### 3. 推送内容

仓库关联完成后即可使用 \$ git push origin master 进行内容推送，推送只会讲已提交至

分支的内容推送至远程仓库（推送时必须在关联（克隆）的工作区进行）

关联后的第一次推送需加关键字 -u 即 \$git push -u origin master

## 分支管理

### 1. 创建分支

\$ git checkout -b dev //创建并且换至分支 == \$ git branch + \$git checkout dev

或 : \$ git branch dev //创建一个分支

\$ git checkout dev //切换分支

\*git branch : 列出所有分支 (前面标一个\*的为当前分支)

## 2. 合并分支

1). 切回master分支 : git checkout master

2). 合并dev分支 :

git merge dev //Fast forward模式合并

git merge --no-ff -m "合并备注信息" dev //非Fast forward模式

合并

3). 删除分支: git branch -d dev //用于删除已合并的分支

强行删除分支 : git branch -D dev //用于删除尚未合并的分支

4). 查看分支历史

查看所有提交(commit)信息 :

git log

查看所有提交(commit)的详细信息 :

git log --graph --pretty=oneline --abbrev-commit

--graph 使用 ASCII 图形布局显示提交树

--date=short 保留日期格式更好且更短

--pretty=... :定义输出的格式

--pretty=oneline : 单行显示

%h 是提交 hash 的缩写

%d 是提交的装饰 ( 如分支头或标签 )

%ad 是创作日期

%s 是注释

%an 是作者姓名

\* 只要主线和分支未修改同一个文件的相同文本都可以成功合并分支 ( Git会将各自的修改合并 )

\* Fast forward 模式下删除分支会丢失分支信息 , 禁用Fast forward模式 , 合并时会生成

commit信息 , 即使分支删除也可以从分支历史上看出分支信息。

## 3. 合并冲突

如果主线和分支同时修改了同一个文件的同一段内容 , 此时合并会冲突。

使用 \$git status 命令查看冲突文件, 打开冲突文件会发现内容多出一些特殊符号

( <<<<< ,

===== , >>>>> ) 此时两个分支已经完成合并 , 只不过内容有冲突 ( 可以直接

删除创建

的分支了），符号标记出不同分支的冲突内容，只需将冲突内容重新修改回最终内容即可。

最终内容可以与两个分支内容都不同，此操作只是简单的修改操作，合并已经完成修改操作

与分支已经没有关系了。

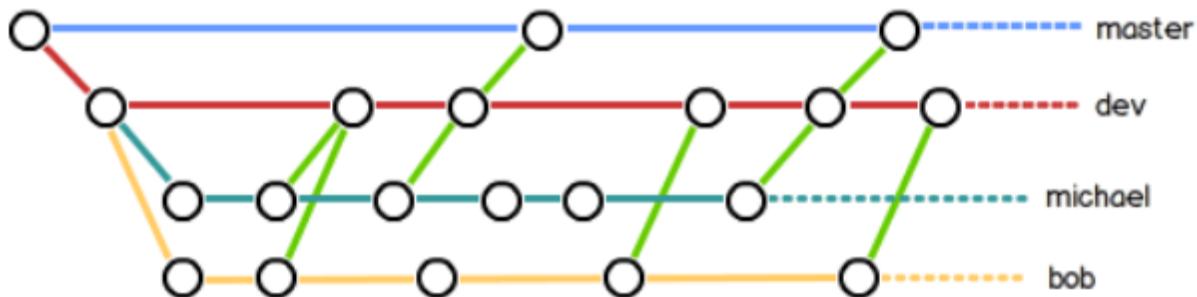
#### 4. 分支使用

master分支应该非常稳定，仅用来发布新版本，平时不可以在上面操作，操作都在分支

dev上进行，当项目结束需要版本发布时，再将dev分支合并到master身上，使用master分

支发布。团队开发是，成员可以从dev分支上分出子分支，可随时进行合并。

团队合作分支图：



#### 5. 隐藏分支

隐藏当前工作现场

git stash      此时使用git status 查看工作区不会显示已隐藏的工作内容

查看隐藏工作现场

git stash list

恢复隐藏工作现场

git stash apply      //恢复工作现场

git stash drop      //删除隐藏记录(stash)

git stash pop      //回复工作现场并删除隐藏记录

\* 修复bug时只需在出现bug的分支上新建一条分支，解决后将其合并即可，尽量不要在原分支

上修复

## 多人协作

1). 获取远程仓库的详细信息 (git remote -v) :

```
Administrator@PC-201207262056 MINGW64 /GitTest/gitskills (master)
$ git remote -v
origin  https://github.com/mynamehzw/gitskills.git (fetch)
origin  https://github.com/mynamehzw/gitskills.git (push)
```

\* **fetch**:表示你具有抓取权限；**push**:表示你具有推送权限

### 2). 创建远程仓库分分支到本地

一般克隆的远程仓库在本地只会显示 master 一条分支，如果需要在其他分支上工作，就必须

必须创建远程仓库的分支到本地： \$ git checkout -b dev origin/dev

\* **origin**:远程仓库的默认名称

### 3). 抓取分支

在进行抓取分支之前需要先指定当前分支与远程分支的链接。

指定当前分支与远程分支的链接：

```
$ git branch --set-upstream dev origin/dev
```

抓取分支：

```
$ git pull
```

\* 一般如果推送失败，可能是因为当前分支不是最新版本，只需将远程最新版抓取到本地，合并

后在进行推送即可。

## 标签管理

1).