

# Git

## 一、Git 基础

### 1、Git 介绍

Git 是目前世界上最先进的分布式**版本控制系统**。

版本控制系统：

设计师在设计的时候做了很多版本

- 设计稿
- 设计稿2
- 设计稿1005
- 设计稿1006
- 设计稿1007
- 设计稿终版
- 设计稿最终确定版本
- 设计稿最终确定版本再改我就不伺候你了

经过了数天去问设计师每个版本都改了啥，设计师此时可能就说不上来了。这个时候如果能有一个软件能记录每次的文件改动，并且还能协调多用户编辑，那岂不是美滋滋？这个软件应用起来应该像这个样子：

版本	文档名	操作用户	日志	修改时间
1	shejigao.txt	zhangsan	修改标题	2019-10-01 10:10:31
2	shejigao.txt	lisi	删除备注信息	2019-10-01 10:11:49
3	shejigao.txt	lisi	增加了许可协议	2019-10-03 11:31:00
4	shejigao.txt	zhangsan	修改版权信息	2019-10-05 09:32:11

### 2、Git 与 Github

#### 2.1、两者区别

Git 是一个分布式版本控制系统，**简单的说其就是一个软件**，用于记录一个或若干文件内容变化，以便将来查阅特定版本修订情况的软件。

Github (<https://www.github.com>) 是一个为用户提供 Git 服务的网站，**简单说就是一个可以放代码的地方**（不过可以放的当然不仅是代码）。Github 除了提供管理 Git 的 web 界面外，

还提供了订阅、关注、讨论组、在线编辑器等丰富的功能。Github 被称之为全球最大的基友网站。

## 2.2、Github 注册

打开 Github 官网: <https://github.com/>, 点击右上角的“Sign up”按钮。

# Create your account

Username \*

用户名

Email address \*

电子邮箱地址



Password \*

密码

Make sure it's at least 15 characters OR at least 8 characters including a number and a lowercase letter.  
[Learn more.](#)

选择免费的账户类型:

## Pick the plan that's right for you

 <b>Free</b> <b>\$0</b> USD The basics of GitHub for every developer <a href="#">Choose Free</a>	 <b>Pro</b> <b>\$7</b> USD Per month Pro tools for developers with advanced requirements <a href="#">Choose Pro</a>
---	---

提示我们需要验证邮箱：



## Please verify your email address

Before you can contribute on GitHub, we need you to verify your email address.

An email containing verification instructions was sent to **itcast@cherish.pw**

打开刚才注册的邮箱，去邮箱中找到邮件点击验证

Didn't get the email? [Resend verification email](#) or [change your email settings](#).

打开邮箱中收到的邮件，点击按钮进行验证：

Almost done, **@bjitcast!** To complete your GitHub sign up, we just need to verify your email address:  
**itcast@cherish.pw.**

Verify email address

点击按钮，进行验证

至此，Github 帐号注册完毕，我们将在后面会使用到本次注册的帐号。

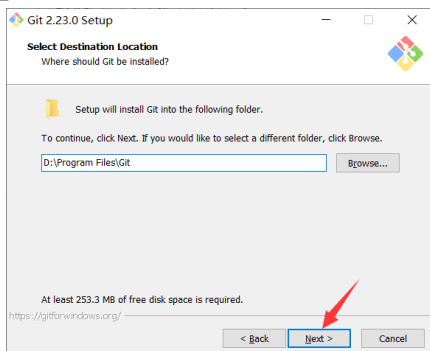
### 3、Git 安装

① 下载得到安装包，并运行

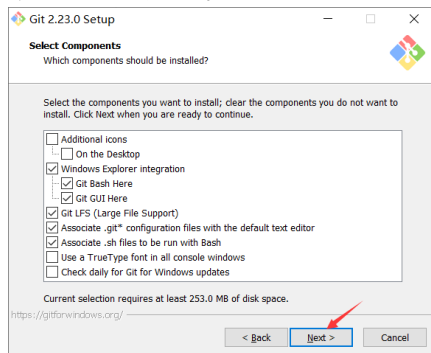
名称

Git-2.23.0-64-bit.exe

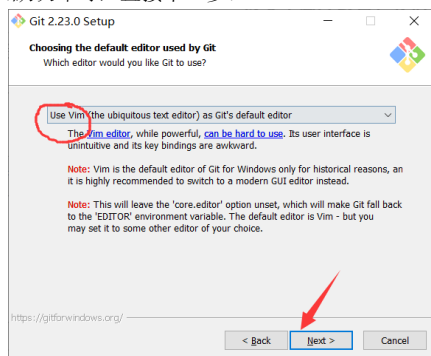
② 选择软件的安装位置



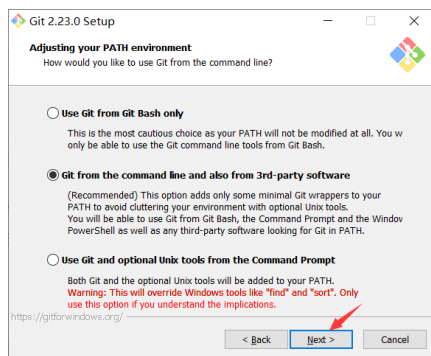
③选择需安装的组件（默认即可，直接下一步）



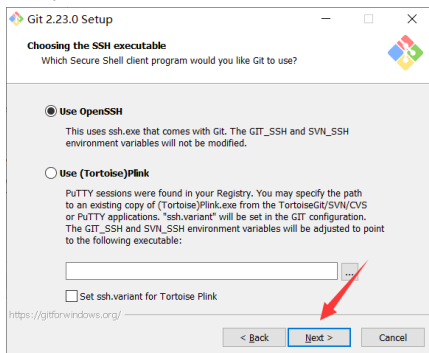
④选择使用的编辑器（默认即可，直接下一步）



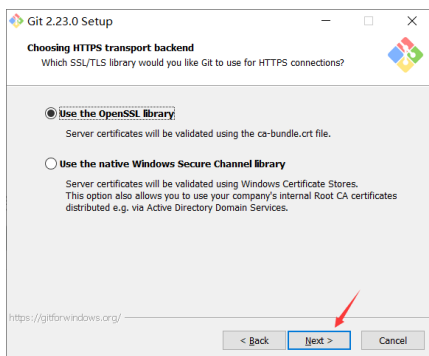
⑤环境变量调节



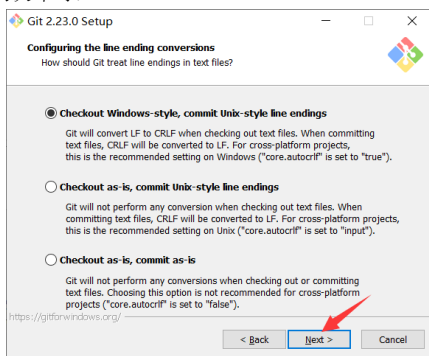
⑥使用 OpenSSH，直接下一步即可



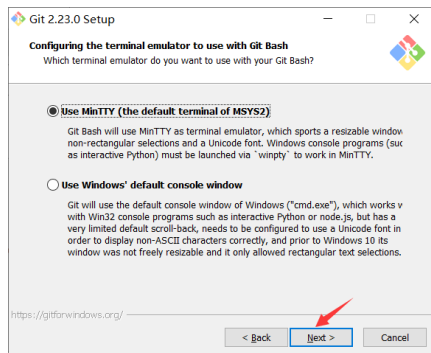
⑦使用 OpenSSL 库



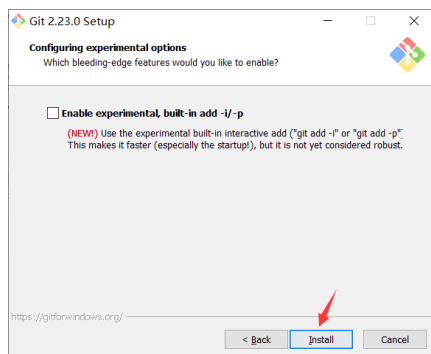
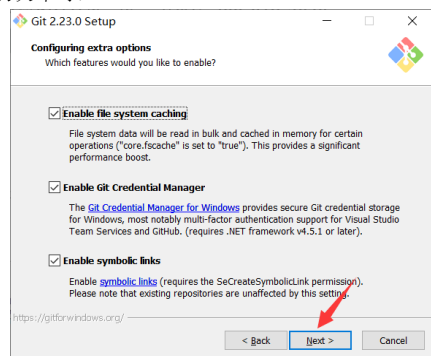
⑧配置命令行会话（默认即可）



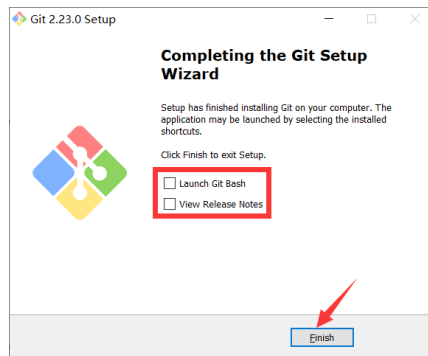
⑨配置终端（默认即可）



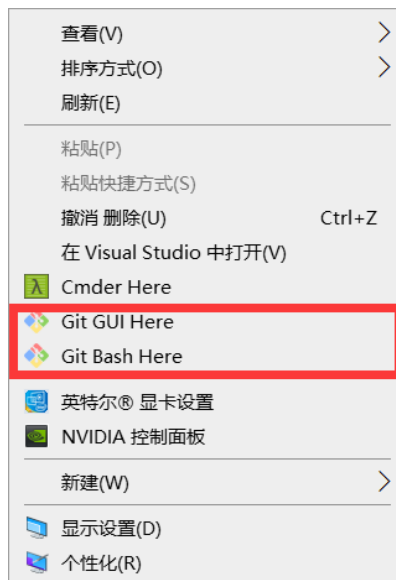
⑩配置额外的选项（默认即可）



安装完成



在桌面空白处右键鼠标，若出现“Git GUI Here”、“Git Bash Here”则安装成功：



## 二、Git 的使用

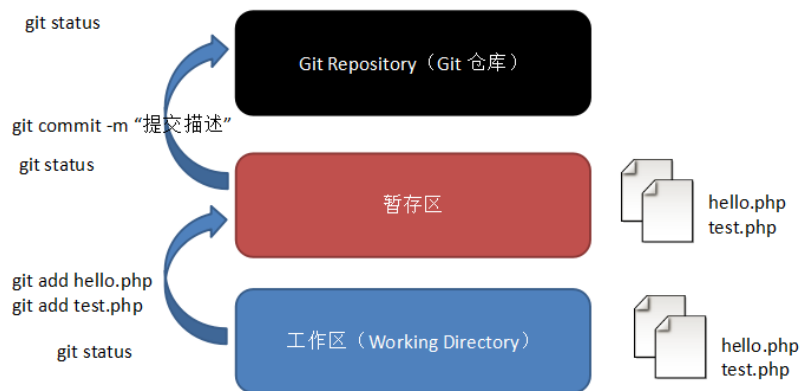
### 1、本地仓库

#### 1.1、工作流程

Git 本地操作的三个区域：



工作流程：





## 1.2、本地仓库操作

什么是仓库呢？仓库又名版本库，英文名 **repository**，我们可以简单理解成是一个目录，用于存放代码的，这个目录里面的所有文件都可以被 **Git** 管理起来，每个文件的修改、删除等操作 **Git** 都能跟踪到。

①在安装好后首次使用需要先进行全局配置

桌面空白处右键，点击“Git Bash Here”以打开 Git 命令行窗口

```
$ git config --global user.name "用户名"
$ git config --global user.email "邮箱地址"
```

```
admin@t2 MINGW64 ~/Desktop
$ git config --global user.name "bjitcast"

admin@t2 MINGW64 ~/Desktop
$ git config --global user.name
bjitcast

admin@t2 MINGW64 ~/Desktop
$ git config --global user.email "itcast@cherish.pw"

admin@t2 MINGW64 ~/Desktop
$ git config --global user.email
itcast@cherish.pw
```

②创建仓库

当我们需要让 **Git** 去管理某个新项目/已存在项目的时候，就需要创建仓库了。注意，创建仓库时使用的目录不一定要是空目录，选择一个非空目录也是可以的，**但是不建议在现有项目上来学习 Git，否则造成的一切后果概不负责！**

注意：为了避免在学习或使用过程中出现各种奇葩问题，请不要使用包含中文的目录名（父目录亦是如此）。

a. 创建空目录



```
admin@t2 MINGW64 ~/Desktop
$ mkdir pro_git

admin@t2 MINGW64 ~/Desktop
$
```

b. 在命令行中进入项目目录 **pro\_git**

```
admin@t2 MINGW64 ~/Desktop
$ cd pro_git

admin@t2 MINGW64 ~/Desktop/pro_git
$ |
```



c. Git 仓库初始化（让 Git 知道，它需要来管理这个目录）

指令：git init

```
pro_git
名称
.git

admin@e2: MINGW64 ~/Desktop/pro_git
$ git init
Initialized empty Git repository in C:/Users/admin/Desktop/pro_git/.git/
admin@e2: MINGW64 ~/Desktop/pro_git (master)
$
```

表现：执行之后会在项目目录下创建“.git”的隐藏目录，这个目录是 Git 所创建的，不能删除，也不能随意更改其中的内容。

### ③Git 常用指令操作

查看当前状态：git status

【非必要】

添加到缓存区：git add 文件名

说明：git add 指令，可以添加一个文件，也可以同时添加多个文件。

语法 1：git add 文件名

语法 2：git add 文件名 1 文件名 2 文件名 3 ...

语法 3：git add . 【添加当前目录到缓存区中】

提交至版本库：git commit -m “注释内容”

在后续对于文件（可以操作 1 个或多个）操作之后，重复使用 git add 与 git commit 指令即可。

```
admin@e2: MINGW64 ~/Desktop/pro_git (master)
$ git status
On branch master

No commits yet

nothing to commit (create/copy files and use "git add" to track)

admin@e2: MINGW64 ~/Desktop/pro_git (master)
$ git add readme.txt
admin@e2: MINGW64 ~/Desktop/pro_git (master)
$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
    new file:   readme.txt

admin@e2: MINGW64 ~/Desktop/pro_git (master)
$ git commit -m "新建文件readme.txt"
[master (root-commit) 45a0b46] 新建文件readme.txt
1 file changed, 1 insertion(+)
create mode 100644 readme.txt
```

### 1.3、时光穿梭机——版本回退

版本回退分为两步骤进行操作：

步骤：

- ①查看版本，确定需要回到时刻点

指令：

```
git log
```

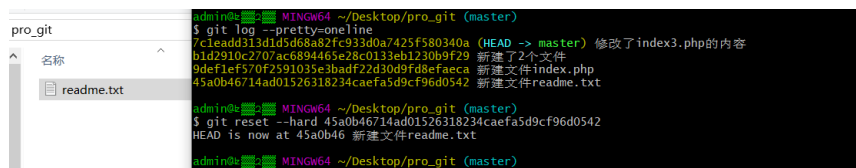
```
git log --pretty=oneline
```

- ②回退操作

指令：

```
git reset --hard 提交编号
```

案例：想坐时光机回到创建好第一个文件 `readme.txt` 的时候。



```
admin0@MINGW64 ~/Desktop/pro_git (master)
$ git log --pretty=oneline
7cleadd313d1d5d68a82fc933d0a7425f580340a (HEAD -> master) 修改了index3.php的内容
b1d2910c2707ac6894465e28c0133eb1230b9f29 新建了2个文件
9def1ef570f2591035e3badf22d30d9fd8efaeca 新建文件index.php
45a0b46714ad01526318234caefa5d9cf96d0542 新建文件readme.txt

admin0@MINGW64 ~/Desktop/pro_git (master)
$ git reset --hard 45a0b46714ad01526318234caefa5d9cf96d0542
HEAD is now at 45a0b46 新建文件readme.txt

admin0@MINGW64 ~/Desktop/pro_git (master)
```

注意：回到过去之后，要想再回到之前最新的版本的时候，则需要使用指令去查看历史操作，以得到最新的 `commit id`。

指令：`git reflog`



```
45a0b46 (HEAD -> master) HEAD@{2}: reset: moving to 45a0b46714ad01526318234caefa5d9cf96d0542
7cleadd HEAD@{3}: commit: 修改了index3.php的内容
b1d2910 HEAD@{4}: commit: 新建了2个文件
9def1ef HEAD@{5}: commit: 新建文件index.php
45a0b46 (HEAD -> master) HEAD@{6}: commit (initial): 新建文件readme.txt

admin0@MINGW64 ~/Desktop/pro_git (master)
$ git reset --hard 7cleadd
HEAD is now at 7cleadd 修改了index3.php的内容

admin0@MINGW64 ~/Desktop/pro_git (master)
$
```

小结：

- 要想回到过去，必须先得到 `commit id`，然后通过 `git reset --hard` 进行回退；
- 要想回到未来，需要使用 `git reflog` 进行历史操作查看，得到最新的 `commit id`；
- 在写回退指令的时候 `commit id` 可以不用写全，`git` 自动识别，但是也不能写太少，至少需要写前 4 位字符；

## 2、远程仓库

线上仓库的操作学习以 Github 为例。


### 2.1、线上仓库创建

打开创建仓库页面: <https://github.com/new>

圈出的部分为必填项，其余根据实际需要选择性补充：

#### Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere?  
[Import a repository.](#)

Owner:  bjitcast / Repository name \*:

Great repository names are short and memorable. Need inspiration? How about [probable-palm-tree?](#)

Description (optional):

☒ Public  
Anyone can see this repository. You choose who can commit.

☐ Private  
You choose who can see and commit to this repository.

Skip this step if you're importing an existing repository.

☐ Initialize this repository with a README  
This will let you immediately clone the repository to your computer.

Add .gitignore:  Add a license:  ⓘ

注意：仓库名要求在当前帐号下唯一。

### 2.2、两种常规使用方式

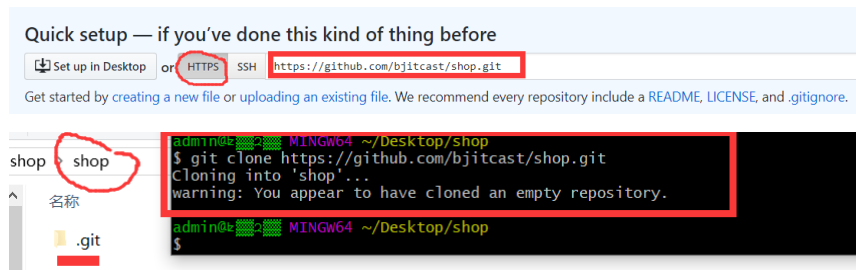
#### 2.2.1、基于 http/https 协议

a. 创建空目录，名称就称为 shop

```
shop
admin@e2222 MINGW64 ~/Desktop
$ mkdir shop
admin@e2222 MINGW64 ~/Desktop
$ cd ./shop
```

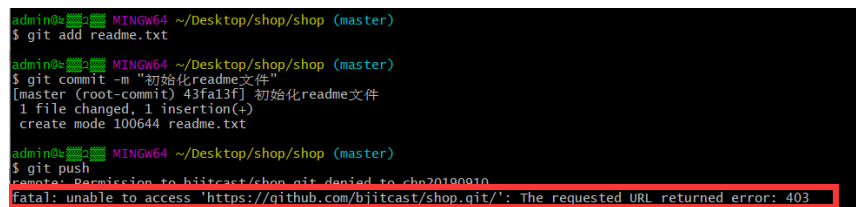
b. 使用 clone 指令克隆线上仓库到本地

语法: git clone 线上仓库地址



c. 在仓库上做对应的操作（提交暂存区、提交本地仓库、提交线上仓库、拉取线上仓库）

提交到线上仓库的指令: git push



在首次往线上仓库 shop 提交内容的时候出现了 403 的致命错误，原因是不是任何人都可以往线上仓库提交内容，必须需鉴权。

需要修改 “.git/config” 文件内容：

```
# 将
[remote "origin"]
  url = https://github.com/用户名/仓库名.git
修改为：
[remote "origin"]
  url = https://用户名:密码@github.com/用户名/仓库名.git
```

例如：

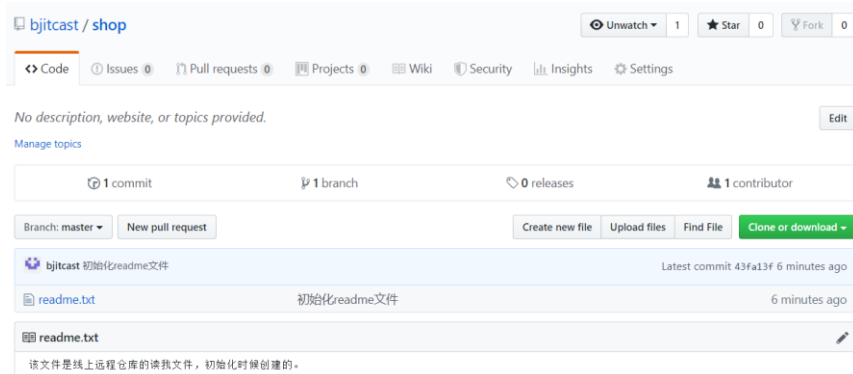


在设置好用户名密码之后再次尝试 push 指令：

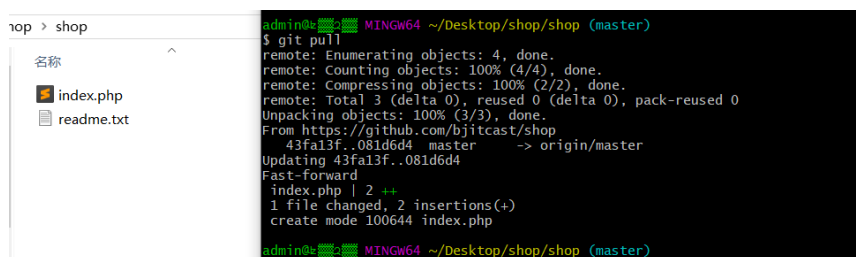
```
admin@e MINGW64 ~/Desktop/shop/shop (master)
$ git push
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 306 bytes | 306.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To https://github.com/bjiticast/shop.git
* [new branch] master -> master
```

如果看到类似上述效果（没有 fatal 错误）则表示提交成功。

【验证】此时可以观察浏览器，刷新线上仓库的地址：



拉取线上仓库：git pull



提醒：

在每天工作的第一件事就是先 git pull 拉取线上最新的版本；每天下班前要做的是 git push，将本地代码提交到线上仓库。

### 2.2.2、基于 ssh 协议（推荐）

该方式与前面 https 方式相比，只是影响 github 对于用户的身份鉴权方式，对于 git 的具体操作（如提交本地、添加注释、提交远程等操作）没有任何影响。

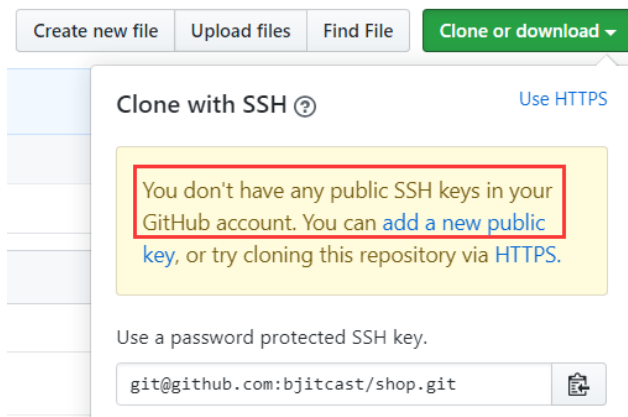
生成公私钥对指令（需先自行安装 OpenSSH）：`ssh-keygen -t rsa -C "注册邮箱"`

步骤：

- ①生成客户端公私钥文件
- ②将公钥上传到 Github

实际操作：

①打开提示



②创建公私钥对文件

```
admin@z: MINGW64 ~/Desktop/shop/shop (master)
$ ssh-keygen -t rsa -C "itcast@cherish.pw"
Generating public/private rsa key pair.
Enter file in which to save the key (/c/Users/admin/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /c/Users/admin/.ssh/id_rsa.
Your public key has been saved in /c/Users/admin/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:1/wyDY81+l1REKo5SZiPwuaw2Gk2zvelusAX/qJEkqU itcast@cherish.pw
The key's randomart image is:
+---[RSA 3072]-----+
|
| o
| o
| . o . . .
| + . + = . . .
| E o = S B . ooo|
| * B o . . .0o|
| . @ + . . o +|
| * +o. o . o|
| +oo+= o|
+---[SHA256]-----+
```

执行指令之后连续回车即可

③上传公钥文件内容 (id\_rsa.pub)

## SSH keys / Add new

Title

我的笔记本电脑 标题任意输入即可

Key

```
ssh-rsa
AAAAAB3NzaC1yc2EAAAADAQABAAQGDwRrCSJ47yd2oqlF1Way7QiDnRqxC+k+YwAlgkDAWbwx5bGN3loJ8XMLug1
PoHLMZIY9kF4SB7rke6JU6LcyzoPJXisDIFIKN8Cm9IA4R700W/v+9Q1pL3gMxvicehGc685B47u9I9xF4YNF/MQrv0zGjd
Rjih+DsOuOEQD+22cflbAeh1nG1TVUOeTqnyArYdamleatttOkOmDeNaRZkiTY8fj/iikbDdpDcrtb4EN+pv0ytjB8lErCC
dLXnHEGD2+EMYP/gGNQEIvGnB5XBvRqbnvk+p7aHAsDGal8GoUQ2l3hb5zPgFgqkcr7r/qCDRNVX2ZvrIsQ4ZYzdl7
aak4X8uLBaOu5Zre5K6dG3uPGtPvNKR3Vt59jG04x5g0Ofv6BeMNRiku0RQsLC7DYN/IEEO8mrAt9kAzGGeVvBSFgYd/
NSBPDprVSz7d2oa5cMKpeFK01jg4++OllEoa+GS5qrAOOQF3R04x6Zttr7GdEjlas/62OYORSH6kM=
itcast@cherish.pw
```

Add SSH key

填写完毕之后保存即可。

④执行后续 git 操作，操作与先前一样

a. clone 线上仓库到本地 (git clone)

```
admin@MINGW64 ~/Desktop/shop
$ cd ../withSSH/

admin@MINGW64 ~/Desktop/shop/withSSH
$ git clone git@github.com:bjitcast/shop.git
Cloning into 'shop' ...
remote: Enumerating objects: 6, done.
remote: Counting objects: 100% (6/6), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 6 (delta 0), reused 3 (delta 0), pack-reused 0
Receiving objects: 100% (6/6), done.
```

b. 修改文件后添加缓存区、提交本地仓库、提交线上仓库

```
admin@MINGW64 ~/Desktop/shop/withSSH/shop (master)
$ git add index2.php

admin@MINGW64 ~/Desktop/shop/withSSH/shop (master)
$ git commit -m "新建index2.php"
[master f99b50b] 新建index2.php
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 index2.php

admin@MINGW64 ~/Desktop/shop/withSSH/shop (master)
$ git push
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 312 bytes | 312.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To github.com:bjitcast/shop.git
081d6d4..f99b50b master -> master
```

在 push 的时候并没有提示要求我们输入帐号密码，因为公私钥已经实现了用户身份鉴权。

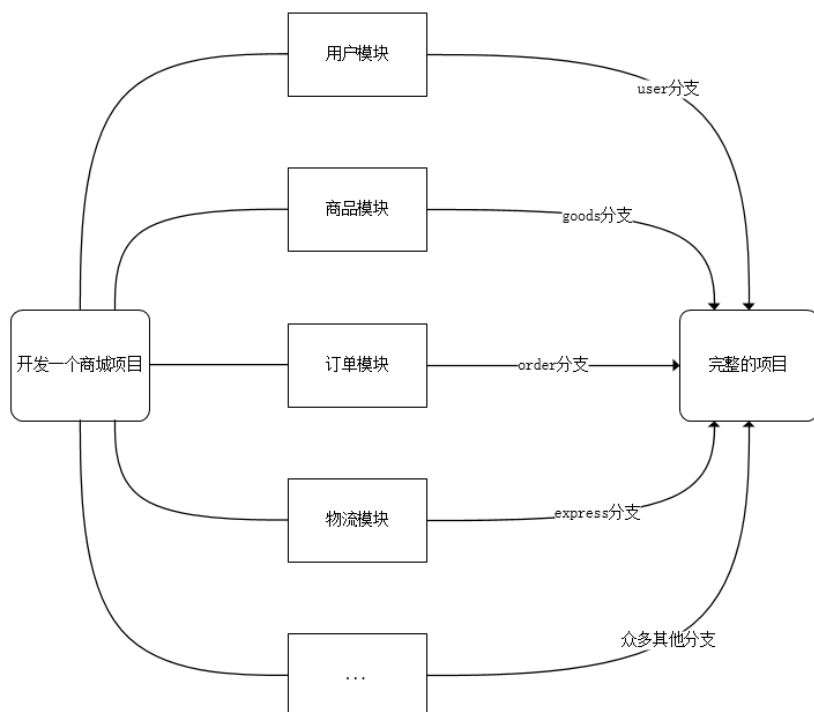


线上仓库的效果：

bjtcast 新建index2.php		Latest commit f99b50b 1 minute ago
index.php	创建index.php	31 minutes ago
index2.php	新建index2.php 通过SSH方式进行提交的	1 minute ago
readme.txt	初始化readme文件	39 minutes ago

## 2.3、分支管理

什么是分支？



在版本回退的章节里，每次提交后都会有记录，Git 把它们串成时间线，形成类似于时间轴的东西，这个时间轴就是一个分支，我们称之为 **master 分支**。

在开发的时候往往是团队协作，多人进行开发，因此光有一个分支是无法满足多人同时开发的需求的，并且在分支上工作并不影响其他分支的正常使用，会更加安全，Git 鼓励开发者使用分支去完成一些开发任务。

分支相关指令：

查看分支：git branch

创建分支：git branch 分支名

切换分支：git checkout 分支名

批注 [DY1]: 对于新分支，可以使用“git checkout -b 分支名”指令来切换分支，-b 选项表示创建并切换，相当于两个操作指令。

>  
< 删除分支: `git branch -d 分支名`  
< 合并分支: `git merge 被合并的分支名`  
>

查看分支:

```
admin@k2 MINGW64 ~/Desktop/shop/withSSH/shop (master)
$ git branch
* master
```

注意: 当前分支前面有个标记“\*”。

创建分支:

```
admin@k2 MINGW64 ~/Desktop/shop/withSSH/shop (master)
$ git branch dev

admin@k2 MINGW64 ~/Desktop/shop/withSSH/shop (master)
$ git branch
dev
* master
```

切换分支:

```
admin@k2 MINGW64 ~/Desktop/shop/withSSH/shop (master)
$ git checkout dev
Switched to branch 'dev'

admin@k2 MINGW64 ~/Desktop/shop/withSSH/shop (dev)
$ git branch
* dev
master
```

合并分支:

现在先在 dev 分支下的 readme 文件中新增一行并提交本地

readme.txt - 记事本  
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)  
该文件是线上远程仓库的读我文件, 初始化时创建的。  
这是在dev分支下新加的内容。

```
admin@k2 MINGW64 ~/Desktop/shop/withSSH/shop (dev)
$ git add readme.txt

admin@k2 MINGW64 ~/Desktop/shop/withSSH/shop (dev)
$ git commit -m "在readme文件中新增一行"
[dev 377f0c7] 在readme文件中新增一行
1 file changed, 3 insertions(+), 1 deletion(-)

admin@k2 MINGW64 ~/Desktop/shop/withSSH/shop (dev)
$
```

切换到 master 分支下观察 readme 文件

readme.txt - 记事本  
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)  
该文件是线上远程仓库的读我文件, 初始化时创建的。  
刚才在dev分支中新增的文字已经消失

```
admin@k2 MINGW64 ~/Desktop/shop/withSSH/shop (dev)
$ git checkout master
Switched to branch 'master'
Your branch is up to date with 'origin/master'.

admin@k2 MINGW64 ~/Desktop/shop/withSSH/shop (master)
$
```

将 dev 分支的内容与 master 分支合并:

readme.txt - 记事本  
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)  
该文件是线上远程仓库的读我文件, 初始化时创建的。  
这是在dev分支下新加的内容。 合并之后的效果

```
admin@k2 MINGW64 ~/Desktop/shop/withSSH/shop (master)
$ git merge dev
Updating f99b50b..377f0c7
Fast-forward
 readme.txt | 4 +++
 1 file changed, 3 insertions(+), 1 deletion(-)

admin@k2 MINGW64 ~/Desktop/shop/withSSH/shop (master)
$
```

删除分支：

```
admin@b2 MINGW64 ~/Desktop/shop/withSSH/shop (master)
$ git branch -d dev
Deleted branch dev (was 377f0c7).

admin@b2 MINGW64 ~/Desktop/shop/withSSH/shop (master)
$ git branch
* master
```

注意：在删除分支的时候，一定要先退出要删除的分支，然后才能删除。

合并所有分支之后，需要将 master 分支提交线上远程仓库中：

bjttcast 在readme文件中新增一行

1 contributor

3 lines (2 sloc) 116 Bytes

1 该文件是线上远程仓库的读我文件，初始化时候创建的。

2 这是在dev分支下新加的内容。

```
admin@b2 MINGW64 ~/Desktop/shop/withSSH/shop (master)
$ git push
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 8 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 397 bytes | 397.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To github.com:bjttcast/shop.git
f99b50b..377f0c7 master -> master
admin@b2 MINGW64 ~/Desktop/shop/withSSH/shop (master)
$
```

## 2.4、冲突的产生与解决

案例：模拟产生冲突。

①同事在下班之后修改了线上仓库的代码

<> Edit file

Preview changes

1 该文件是线上远程仓库的读我文件，初始化时候创建的。

2

3 这是在dev分支下新加的内容。

4

5

6 这是我同时小A在我下班之后做的修改。

注意：此时我本地仓库的内容与线上不一致的。

7 lines (3 sloc) 172 Bytes

1 该文件是线上远程仓库的读我文件，初始化时候创建的。

2

3 这是在dev分支下新加的内容。

4

5

6 这是我同时小A在我下班之后做的修改。

readme.txt - 记事本

文件(E) 编辑(E) 格式(O) 查看(V) 帮助(H)

该文件是线上远程仓库的读我文件，初始化时候创建的。

这是在dev分支下新加的内容。

②第二天上班的时候，我没有做 git pull 操作，而是直接修改了本地的对应文件的内容

📄 readme.txt - 记事本

文件(E) 编辑(E) 格式(O) 查看(V) 帮助(H)

该文件是线上远程仓库的读我文件，初始化时候创建的。

这是在dev分支下新加的内容。

这些文字是我次日上班时候写的。

③需要在下班的时候将代码修改提交到线上仓库（git push）

```
admin@e MINGW64 ~/Desktop/shop/withSSH/shop (master)
$ git add readme.txt

admin@e MINGW64 ~/Desktop/shop/withSSH/shop (master)
$ git commit -m "修改readme文件"
[master 01b9bfc] 修改readme文件
1 file changed, 3 insertions(+), 1 deletion(-)

admin@e MINGW64 ~/Desktop/shop/withSSH/shop (master)
$ git push
To github.com:bjitcast/shop.git
! [rejected]        master -> master (fetch first)
error: failed to push some refs to 'git@github.com:bjitcast/shop.git'
hint: Updates were rejected because the remote contains work that you do
hint: not have locally. This is usually caused by another repository pushing
hint: to the same ref. You may want to first integrate the remote changes
hint: (e.g., 'git pull ...') before pushing again.
hint: See the 'note about fast-forwards' in 'git push --help' for details.
```

提示我们要在再次 push 之前先 git pull 操作。

#### 【解决冲突】

④先 git pull

```
admin@e MINGW64 ~/Desktop/shop/withSSH/shop (master)
$ git pull
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 3 (delta 2), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), done.
From github.com:bjitcast/shop
377f0c7..a709ab6 master -> origin/master
Auto-merging readme.txt
CONFLICT (content): Merge conflict in readme.txt 合并冲突到readme.txt中
Automatic merge failed; fix conflicts and then commit the result.
```

此时 git 已经将线上与本地仓库的冲突合并到了对应的文件中。

⑤打开冲突文件，解决冲突

解决方法：需要和同事（谁先提交的）进行商量，看代码如何保留，将改好的文件再次提交即可。

该文件是线上远程仓库的读我文件，初始化时候创建的。

这是在dev分支下新加的内容。

这些文字是我次日上班时候写的。

将里面的内容进行调整，保留需要的，不需要的删除即可。

这是我同时小A在我下班之后做的修改。

|

#### ⑥重新提交

```
admin@zabbix MINGW64 ~/Desktop/shop/withSSH/shop (master|MERGING)
$ git add readme.txt

admin@zabbix MINGW64 ~/Desktop/shop/withSSH/shop (master|MERGING)
$ git commit -m "解决了冲突"
[master a2095bf] 解决了冲突

admin@zabbix MINGW64 ~/Desktop/shop/withSSH/shop (master)
$ git push
Enumerating objects: 10, done.
Counting objects: 100% (10/10), done.
Delta compression using up to 8 threads
Compressing objects: 100% (6/6), done.
Writing objects: 100% (6/6), 611 bytes | 305.00 KiB/s, done.
Total 6 (delta 4), reused 0 (delta 0)
remote: Resolving deltas: 100% (4/4), completed with 2 local objects.
To github.com:bjitcast/shop.git
 a709ab6..a2095bf master -> master
```

线上效果：

11 lines (4 sloc) | 221 Bytes

```
1 该文件是线上远程仓库的读我文件，初始化时候创建的。
2
3 这是在dev分支下新加的内容。
4
5
6 这些文字是我次日上班时候写的。
7
8
9 这是我同时小A在我下班之后做的修改。
```

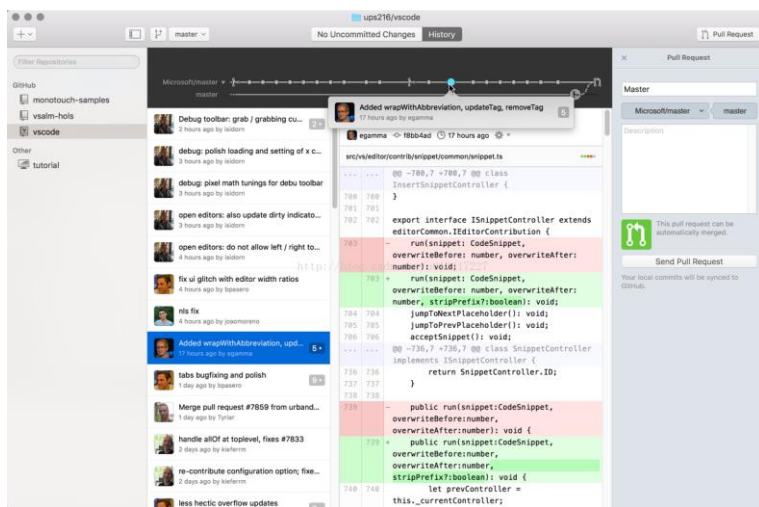
此时线上显示是最新的，已经发生改变。

新手上路小技巧：上班第一件事先 `git pull`，可以在一定程度上避免冲突的产生。

## 三、Git 实用技能

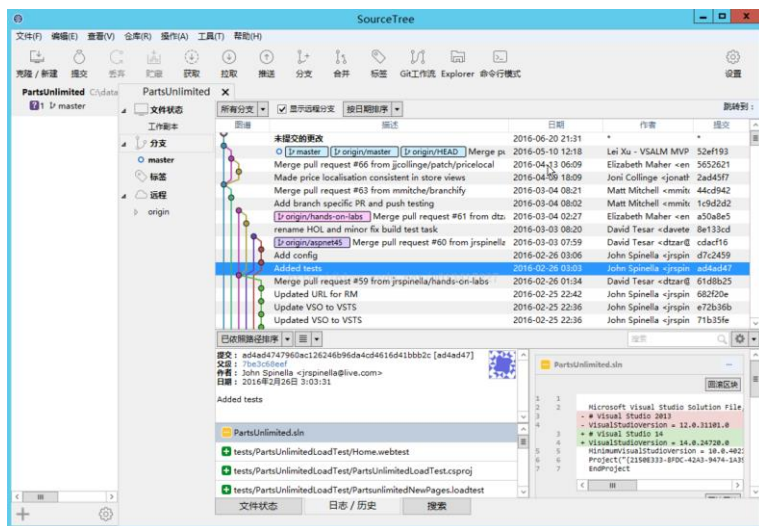
### 1、图形管理工具

#### ①Github for Desktop



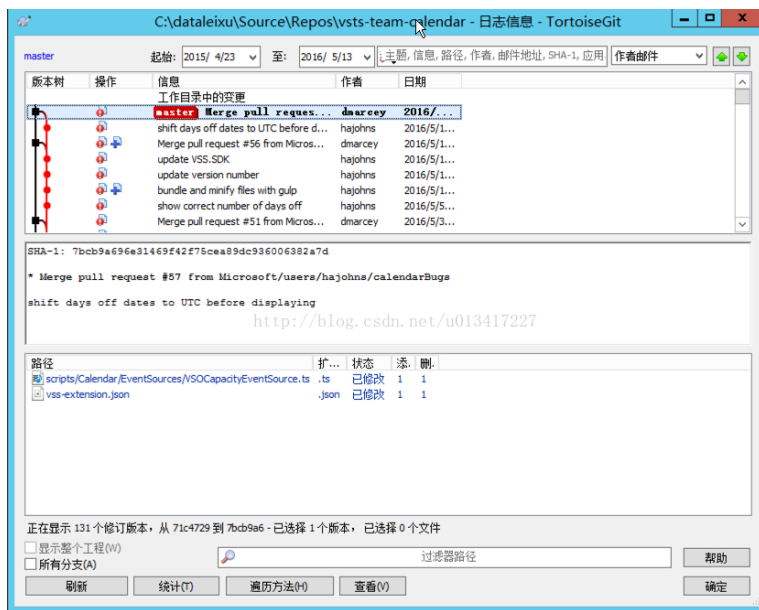
Github 出品的软件，功能完善，使用方便。对于经常使用 GitHub 的开发人员来说是非常便捷的工具。界面干净，用起来非常顺手，顶部的分支时间线非常绚丽。

#### ②Source tree



老牌的 Git GUI 管理工具了，也号称是最好用的 Git GUI 工具。功能丰富，基本操作和高级操作都非常流畅，适合初学者上手。

### ③TortoiseGit



对于熟悉 SVN 的开发人员来说，这个小乌龟图标应该是非常友善了。TortoiseGit 简称 git，中文名海龟 Git。它与其前辈 TortoiseSVN 都是非常优秀的开源版本控制客户端软件。

## 2、忽略文件

场景：在项目目录下有很多万年不变的文件目录，例如 css、js、images 等，或者还有一些目录即便有改动，我们也不想让其提交到远程仓库的文档，此时我们可以使用“忽略文件”机制来实现需求。

忽略文件需要新建一个名为.gitignore 的文件，该文件用于声明忽略文件或不忽略文件的规则，规则对当前目录及其子目录生效。

注意：该文件因为没有文件名，没办法直接在 windows 目录下直接创建，可以通过命令行 Git Bash 来 touch 创建。

常见规则写法有如下几种：

- 1) /mtk/ 过滤整个文件夹
- 2) \*.zip 过滤所有.zip 文件
- 3) /mtk/do.c 过滤某个具体文件
- 4) !index.php 不过滤具体某个文件

在文件中，以#开头的都是注释。

案例：

①先在本地仓库中新建一个 js 目录以及目录中 js 文件

p > withSSH > shop > js

名称

login.js

②依次提交本地与线上

```
admin@MINGW64 ~/Desktop/shop/withSSH/shop (master)
$ git add .

admin@MINGW64 ~/Desktop/shop/withSSH/shop (master)
$ git commit -m "新增JavaScript目录"
[master 4084c6f] 新增JavaScript目录
1 file changed, 18 insertions(+)
create mode 100644 js/login.js

admin@MINGW64 ~/Desktop/shop/withSSH/shop (master)
$ git push
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 8 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (4/4), 657 bytes | 328.00 KiB/s, done.
Total 4 (delta 0), reused 0 (delta 0)
To github.com:bjitcast/shop.git
a2095bf..4084c6f master -> master
```

③新增.gitignore 文件

js

.gitignore

index.php

a2095bf..4084c6f master -> master

```
admin@MINGW64 ~/Desktop/shop/withSSH/shop (master)
$ touch .gitignore
```

④编写文件中的规则（根据需要编写）

.gitignore - 记事本

文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

#忽略掉/js目录

/js/

⑤再次提交本地与线上



```
admin@MINGW64 ~/Desktop/shop/withSSH/shop (master)
$ touch .gitignore

admin@MINGW64 ~/Desktop/shop/withSSH/shop (master)
$ git add .

admin@MINGW64 ~/Desktop/shop/withSSH/shop (master)
$ git commit -m "新增index.js"
[master b10385a] 新增index.js
1 file changed, 2 insertions(+)
create mode 100644 .gitignore

admin@MINGW64 ~/Desktop/shop/withSSH/shop (master)
$ git push
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 303 bytes | 151.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To github.com:bjitcast/shop.git
  4084c6f..b10385a master -> master
```

观察线上仓库 js 目录中是否有新增 index.js 文件:

[Code](#) [Issues 0](#) [Pull requests 0](#) [Projects 0](#) [Wiki](#)

Branch: master [shop / js /](#)

 bjitcast 新增JavaScript目录

..

并没有新增index.js文件

 login.js

新增JavaScript目录