

Git 入门技术分享(吕小杰)

看完还不会用的 GIT 操作,你就掐死我!!!

阅读 5319, 1 月 13 日 发布, 来源: blog.yubangweb.com

无论你是前端还是后台, 无论是运维还是移动端研发, GIT 是逃避不了的东西, 当然你说你要用 SVN, 那不在这次的讨论范围之内。不多说, 请看下文 GIT 图解分析, 10 分钟学会 git 操作, 当然下面的教程是为实战为主, 会跟你在别的网站看到的不一樣。

1. GIT 是啥玩意呀?





首先每一个项目，我们都把他变成一个 git 仓库。

- 一个 git 仓库包含无数分支，默认分支为 master
- 每个分支都包含无数个版本库
- 每个版本库都包含无数个文件

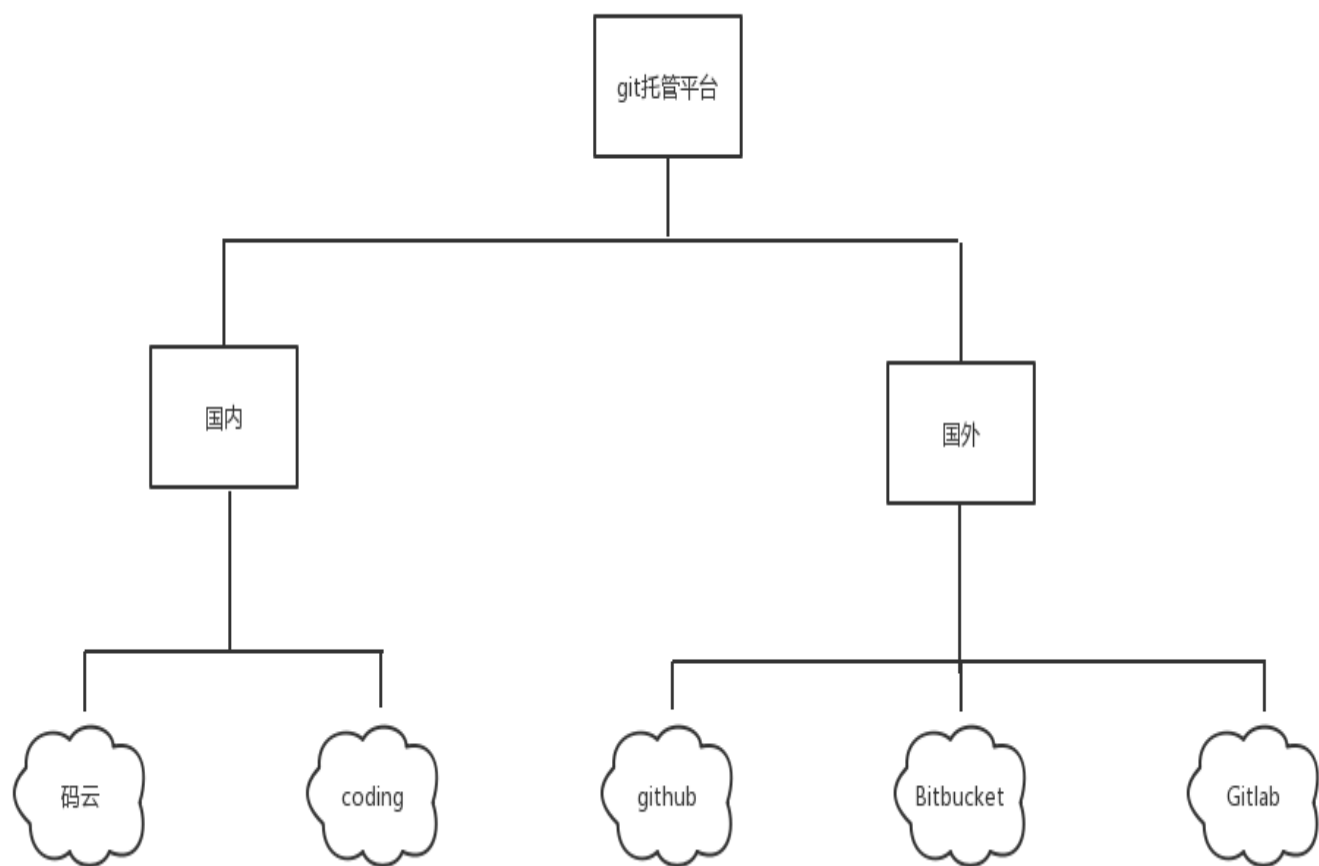
注：具体包含关系看上图哈，看这图仅仅让你知道 git 的样子

我们为什么要用 GIT 呢？

- 我们可以每次修改一些文件之后，冻结住当前所有文件，然后定义成一个版本，让自己有一颗后悔药吃，可以随时拿到某个版本的文件内容。
- 记录下每个人修改了什么，可以秋后算账（后半句开玩笑啦）

2.创建一个 git 项目

- 寻找一家第三方 git 托管平台商（知道就不看下面的图片，不知道就看看下面的图片哈）




注：国内的码云，coding，国外的 github，Bitbucket,或者 gitlab，哪家自己看名字比较喜欢就选哪家啦。都不喜欢，那么可以自己搭建 git 平台。

- 找到一家服务商之后，注册账号，然后在网页点击新建项目（下图是我用 github 来做示范）

Create a new repository

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

Owner

 yubang ▾

Repository name

/ test ✓

Great repository names are short and memorable. Need inspiration? How about **fictional-octo-barnacle**.

Description (optional)

为了写文章而建立的项目



Public

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



Private

You choose who can see and commit to this repository.



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.

Add .gitignore: None ▾

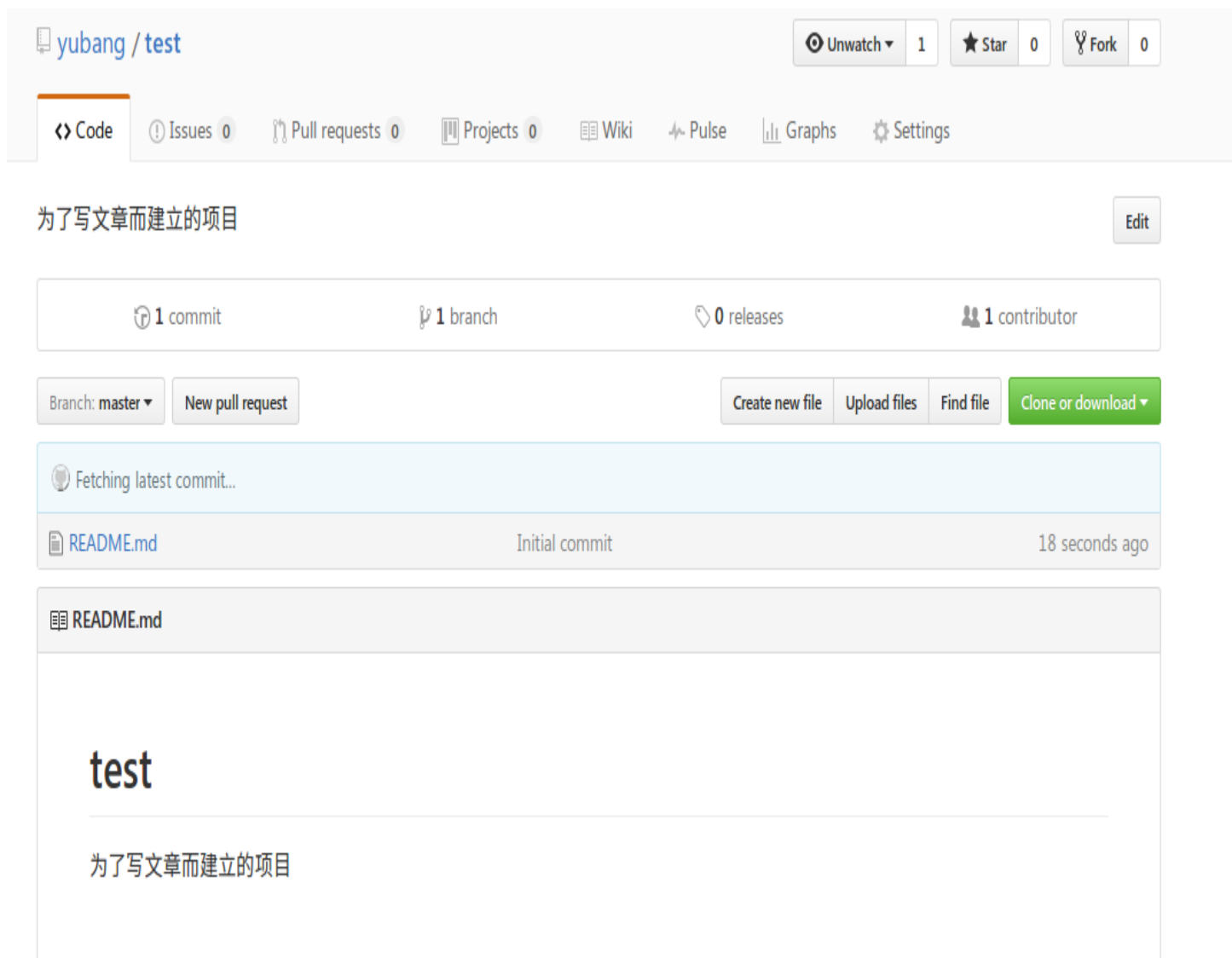
Add a license: None ▾



Create repository

注：test 改成自己的项目名，那段中文改成自己的项目描述，其它的不用理。（都说是实战教程，只教你速成）

- 创建完成之后，会跳转到项目主页（不要卡掉这个页面先哈）



3. 在自己电脑安装一个 git 客户端

- window 用户：在官网下载一个安装包，然后运行，不停点 next 就好，啥都默认即可。
- linux 用户：拿 centos 做例子，执行命令 `yum install -y git` 即可
- mac 用户：抱歉，没有钱买苹果机来测试

4. 把刚才新建的项目同步到本地

- window 用户打开 git 终端
- linux 和苹果用户打开普通终端即可

然后在刚才哪个项目主页，找到 git 地址，注意这玩意有两个地址。一个是 https，而另一个是 ssh。我们本着最简单的原则，我们选择 copy https 这个地址。然后听我命令，在命令行执行 `git clone 地址` 假如提示你输入账

号密码就输入账号密码，也有可能直接什么都不用。成功之后你就会发现本地多了一个文件夹，名字就是项目

```
✕ _ □ 终端 文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
[root@d62934a3a791 /]# git clone https://github.com/yubang/test.git
Cloning into 'test'...
remote: Counting objects: 3, done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), done.
[root@d62934a3a791 /]#
```

名。

5.git 实战的第一步,生成一个版本

- 首先，先随意添加几个文件
- 然后，命令行先切换到工程目录里面（这个很重要）
- 接着，执行 `git add` 文件或文件夹相对路径（可以执行多条语句）
- 上面的语句是告诉 `git` 你修改了那些文件，并把修改暂时存了起来。
- 这个时候，距离生成一个版本只差一个命令，就是 `git commit -m "备注"`，当然如果你是跟着本教程走下来的，恭喜你，遇到了一个崩溃的错误。

```
[root@d62934a3a791 /]# cd test/
[root@d62934a3a791 test]# ls
1.txt 2.txt 3.txt README.md
[root@d62934a3a791 test]# git add 1.txt
[root@d62934a3a791 test]# git add 3.txt
[root@d62934a3a791 test]# git commit -m "try add file"

*** Please tell me who you are.

Run

  git config --global user.email "you@example.com"
  git config --global user.name "Your Name"

to set your account's default identity.
Omit --global to set the identity only in this repository.

fatal: unable to auto-detect email address (got 'root@d62934a3a791.(none)')
```

这个错误出现是我们没有配置 `git` 信息，我们只需要配置一次即可。（执行下面的命令，中文自己替换哈）

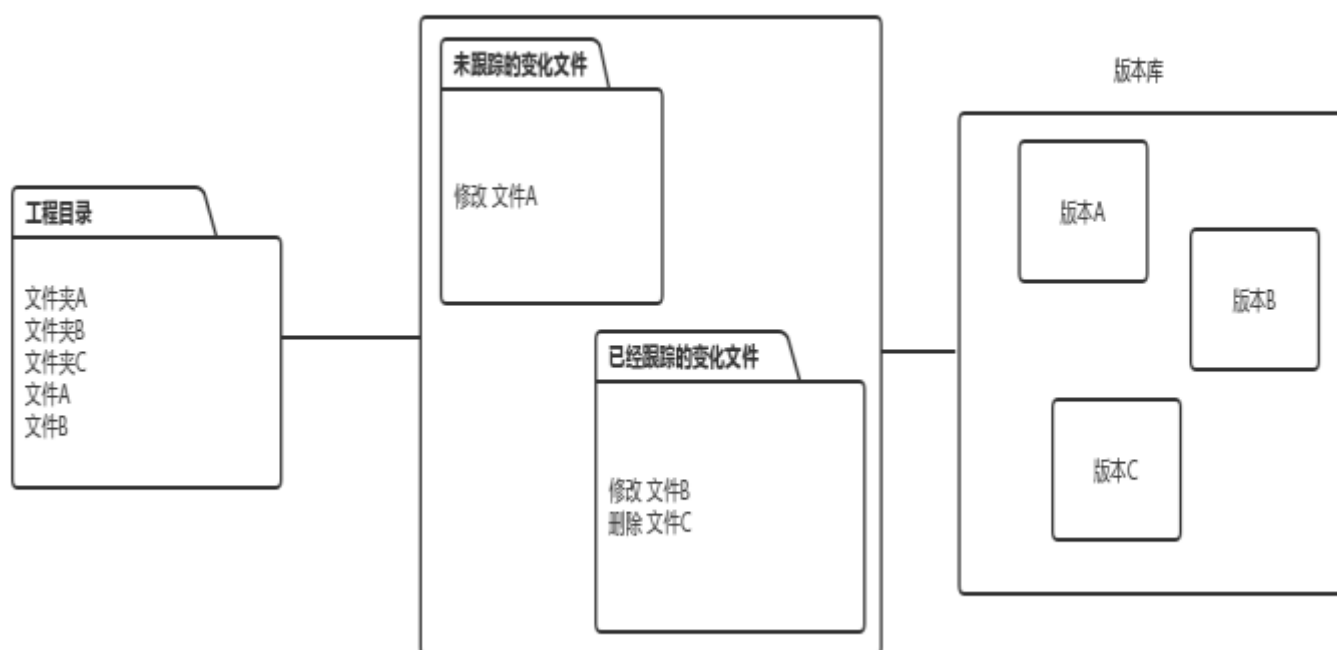
```
git config --global user.email "你的邮箱"
```

```
git config --global user.name "你的名字"
```

然后继续恢复执行 `git commit -m "备注"`，显示成功。

```
[root@d62934a3a791 test]# git commit -m "try add file"
[master 9f8504d] try add file
2 files changed, 2 insertions(+)
create mode 100644 1.txt
create mode 100644 3.txt
[root@d62934a3a791 test]#
```

分析时刻：（请注意对照下图来看）首先我们了解几个概念，我们当前的目录叫做工作区，然后有一个叫做缓存区的东西，接着还有一个叫做版本库的玩意儿。其实我们添加，修改或者删除了工程里面的文件，git 都会发现，并且标志为未跟踪的文件，而 git add 路径这条命令就是告诉 git 把文件列入跟踪文件列表，而 git commit -m "备注"就是把跟踪文件列表的操作清空，而清空前当然是记录成一个版本啦。查看缓冲区的情况只需要输入 git status。



注：每次生成版本只需要，add，然后 commit 即可。

6.同步本地的版本库到托管平台

- 简单的按下 git push origin master 即可
- 当然这个时候我们需要输入账号和密码

```
Counting objects: 5, done.
Delta compression using up to 2 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (4/4), 304 bytes | 0 bytes/s, done.
Total 4 (delta 0), reused 0 (delta 0)
To https://github.com/yubang/test.git
c6f64b8..9f8504d master -> master
```


1. 同步服务器的版本库到本地
2. 简单的按下 `git pull origin master` 即可
3. 当然这个时候我们需要输入账号和密码

注意，如果服务器的版本库的某个文件修改了，而本地又刚好修改了，必须把本地这个文件的变更变成一个版本，然后再执行上面 `pull` 命令。（一般同步的时候，代码合并会成功的，当然也有失败的情况，下面会讲解合并失败的处理方案）

相信你看了上面的教程之后，就可以耍耍 *git* 这凶残的工具了。前方高能预警，*git* 高级用法来了。（如果你要求可以应付项目，下面的忽略即可）

- 我不要输密码，我不要输密码，我不要输密码（总要做的事情说三遍） 方法一：本地记住密码，只需要执行下面的这条命令

```
git config --global credential.helper store
```

方法二：还记得上面我们使用了最简单的 `https` 地址么，只需要换成 `ssh` 地址，但是本地需要生成一个证书（执行下面的命令，要按三次回车）

```
ssh-keygen -t rsa -C "你的邮箱"
```

然后打开文件（`linux` 在 `~/.ssh/idrsa.pub`，`window` 在 `C:\Users\用户名\.ssh\idrsa.pub`），然后在第三方托管平台添加公钥，内容就是这个文件的内容。然后就像平时那么操作即可。

- 我误删了一个文件，我要恢复它

```
git 版本号 checkout -- 文件路径
```

- 我想尝试文件时光机 下面的命令会彻底把本地状态变成某个版本的状态，回退后建立新版本，再调用 `push` 命令的时候需要加上 `-f` 参数

```
git reset --hard 版本号
```

- 如何玩分支的

```
git branch #查看本地分支
```

```
git branch -r #查看远端分支
```

```
git branch -a #查看所有分支，包括本地和远程的
```

```
git branch 分支名 #新建一个分支
```

```
git checkout -b 分支名 #切换到一个分支（注意，本地文件也会变成分支的当前版本的文件）
```

```
git branch -d 分支名 #删除本地分支
```


- 冲突合并

一般出现冲突，文件都会出现一堆神奇的字符，我们这个时候只需要二选一，把不需要的字符都删掉，然后执行 add, commit 命令即可解决冲突。

<<<<<<< HEAD

你好

=====

您好

>>>>>>> 未知字符串

注：该教程仅仅为了让第一次接触 GIT 的同学快速入门而已，并不面向有经验的同学。

最后附上 Git 常用命令速查表

Git 常用命令速查表		master : 默认开发分支	Head : 默认开发分支
		origin : 默认远程版本库	Head^ : Head 的父提交
=====			
创建版本库		分支与标签	
\$ git clone <url>	#克隆远程版本库	\$ git branch	#显示所有本地分支
\$ git init	#初始化本地版本库	\$ git checkout <branch/tag>	#切换到指定分支或标签
修改和提交		\$ git branch <new-branch>	#创建新分支
\$ git status	#查看状态	\$ git branch -d <branch>	#删除本地分支
\$ git diff	#查看变更内容	\$ git tag	#列出所有本地标签
\$ git add .	#跟踪所有改动过的文件	\$ git tag <tagname>	#基于最新提交创建标签
\$ git add <file>	#跟踪指定的文件	\$ git tag -d <tagname>	#删除标签
\$ git mv <old> <new>	#文件改名	合并与衍合	
\$ git rm <file>	#删除文件	\$ git merge <branch>	#合并指定分支到当前分支
\$ git rm --cached <file>	#停止跟踪文件但不删除	\$ git rebase <branch>	#衍合指定分支到当前分支
\$ git commit -m "commit message"	#提交所有更新过的文件	远程操作	
\$ git commit --amend	#修改最后一次提交	\$ git remote -v	#查看远程版本库信息
查看提交历史		\$ git remote show <remote>	#查看指定远程版本库信息
\$ git log	#查看提交历史	\$ git remote add <remote> <url>	#添加远程版本库
\$ git log -p <file>	#查看指定文件的提交历史	\$ git fetch <remote>	#从远程库获取代码
\$ git blame <file>	#以列表方式查看指定文件的提交历史	\$ git pull <remote> <branch>	#下载代码及快速合并
撤销		\$ git push <remote> <branch>	#上传代码及快速合并
\$ git reset --hard HEAD	#撤销工作目录中所有未提交文件的修改内容	\$ git push <remote> :<branch/tag-name>	#删除远程分支或标签
\$ git checkout HEAD <file>	#撤销指定的未提交文件的修改内容	\$ git push --tags	#上传所有标签
\$ git revert <commit>	#撤销指定的提交		

Atom-IDE, 来自 Github 和 Facebook

C#, Flow, Java, JavaScript, PHP, and TypeScript...[查看全文](#)>

cheri · 12 小时前

工具

ssh keys **管理工具**

多账户 github, 切换管理 sshkeys [查看全文](#)>

天生的黑 · 15 小时前

工