Git–版本管理–多人协作

标签　　空

新增属性

官网地址：https://git-scm.com/

Git 是 linus 花费两周使用 c 语言写的分布式版本控制系统，它用于多人协做项目中进行版本管理。也就是说，当某个项目由多个人同时开发，最终提交后需要将代码合并，而 Git 可以使你避免手动合并。

原理

Git 最主要的功能就是拉取和推送。

拉取

创建一个远程的 Git 仓库，、通过命令将远程仓库及内容拉取到本地。此时会复制出一个本地的仓库（即.git 文件夹）和远程代码。

推送

修改完本地代码，我们将变更的部分提交到本地仓库，然后通过命令将本地仓库推送到远程仓库。

常见概念

- 工作区：就是你的代码。
- 暂存区：add 之后的文件都被放在暂存区，然后通过 commit 提交到本地仓库。
- 本地仓库：一个.git 文件夹，可以是本地创建的，也可以是远程拉取的。
- 远程仓库：一个不在本地的仓库，和本地的作用类似。

常用指令

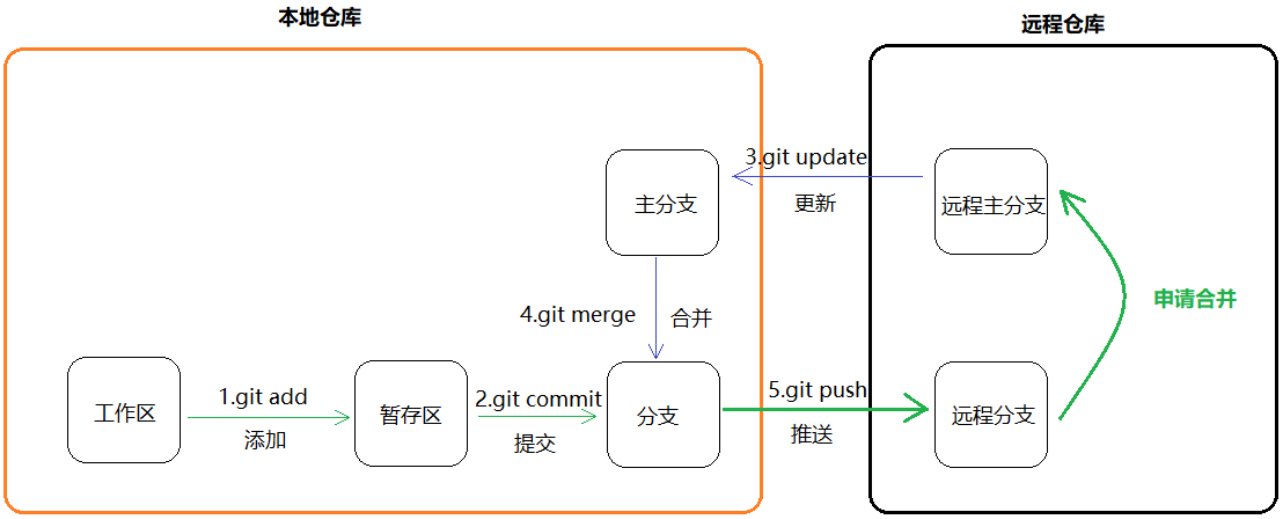
拉取仓库和内容

用途	指令	备注
拉取仓库	git clone 仓库路径	仓库路径根据远程仓库获取，可以是 ssh、http 等
拉取内容	git pull	拉取最新的内容

> ssh 拉取速度 > http 拉取速度，不过 http 简单暴力。

提交及推送

提交代码的操作如下：



提交推送操作

用途	指令	备注
添加代码到暂存区	git add .	. 代表所有更新的文件，可以单独指定具体文件名
提交代码到本地仓库	git commit –m "提交说明"	–m 参数是提交后的说明信息
推送代码到远程仓库	git push	首次推送 git push –u origin master –f

> 首次推送时实际上是推送了整库：master 代表推送主分支，–f 表示强制推送

撤销操作

用途	指令	备注
撤销变更	git checkout – – 文件	
撤销暂存区添加	git reset head 文件名	撤销 add 操作，如果不带文件名则表示所有文件
删除暂存区文件	git rm – – cached 文件	直接删除暂存区文件，不是撤销

查看当前状态

用途	指令	备注
查看提交状态	git status	
查看最新修改	git diff 文件名	

详细内容

Git 安装

在使用 git 前，需要在主机上安装 git。

Linux 安装

PowerShell

```
# 检查Git是否安装
git
```

```
# 安装Git
apt-get install git
```

Windows安装

从Git官网下载安装程序，双击即可安装。windows需要设置仓库名字、邮箱，因为每个机器需要介绍自己：

```PowerShell
#--global参数本机表示所有Git仓库
$ git config --global user.name "Your Name"
$ git config --global user.email "email@example.com"
```

仓库目录不要用中文、MS-Word不会被跟踪、自带记事本不要用，推荐Notepad++设置UTF-8 without BOM。

Git 仓库

远程仓库：创建空仓库→关联ssh用户→添加本地主机密钥。

本地仓库

本地仓库实际上就是一个.git目录，该目录被Git管理，内部文件内容的变动都会被跟踪，并可以在之后还原。在某个目录下，使用如下命令即可构建：

```PowerShell
git init
```

不过我们也可以通过克隆远程仓库而产生一个本地仓库：

```PowerShell
git clone git用户名@服务IP地址:/仓库目录/仓库名.git
```

远程仓库（中央仓库）

- 方式一：直接在github或gitee上创建一个空的远程仓库。
- 方式二：在自己的远程主机上创建一个中央仓库。

方法二详细（Linux上搭建）：

1.首先我们需要在远程主机创建一个纯净的版本仓库：

```Git
# 初始化一个没有暂存区的版本库demo.git
git  init --bare demo.git
```

2.在远程主机创建一个用于git连接的用户给本地使用：

```PowerShell
# 创建一个远程主机账户
adduser gituser
passwd gituser
# 设置密码为123456
```

由于该账户仅用于连接git，因此禁止该账户shell连接：

> vim /etc/passwd

```PowerShell
gituser:......:/home/git:/bin/bash
# 改为
gituser:......:/home/git:/usr/bin/git-shell
```

3.将git仓库demo.git与指定的用户gituser关联：

```PowerShell
chown -R gituser:123456 demo.git
```

此时git的连接地址就变成了：git用户名@服务IP地址:/仓库目录/仓库名.git

到此，一个空的中央仓库搭建完毕！！！

本地仓库与远程仓库互联

远程仓库创建完毕，我们需要与远程仓库进行连接才能获取、推送内容。通常情况下我们使用ssh进行连接。

创建本地主机的ssh密钥：

```PowerShell
#在 ~/.ssh目录下执行命令，一直回车，id_rsa.pub内容就是公钥
$ ssh-keygen -t rsa -C "你的邮箱"
```

将公钥id_rsa.pub内容复制到中央仓库的git用户的.ssh中：即gituser/home/.ssh/authorized_keys文件

人少：直接添加到/home/git/.ssh/authorized_keys文件；人多：用Gitosis。

> 如果允许多台本地主机连接此中央仓库，则将所有主机的公钥加入即可。

然后在本地任意目录使用克隆或关联，即可操作远程仓库了：

```PowerShell
# 方法一：直接克隆远程
git clone git用户名@服务IP地址:/仓库目录/仓库名.git
# 方法二：本地与远程关联
git remote add origin git用户名@服务IP地址:/仓库目录/仓库名.git
```

不过，如果远程仓库刚刚创建，我们需要在本地仓库进行一次推送，首次推送命令如下：

```PowerShell
git push -u origin master -f
```

版本控制

每次提交都会形成一个版本，可以通过命令来进行版本切换。

版本切换

用途	指令	备注
查看版本列表	git log	添加--pretty=oneline参数，查看简要信息
回退一个版本	git reset --hard HEAD^	窗口关闭前可用版本号回跳，HEAD~n表示上n代
跳到指定版本	git reset --hard 1094a	版本号为commit的值，可以输任意长度，但注意类似的版本号
查看git命令历史	git reflog	关电脑时想撤销回退，可以使用此命令查看命令历史
版本比较	git diff HEAD -- 文件名	查看工作区和版本库里最先版本区别

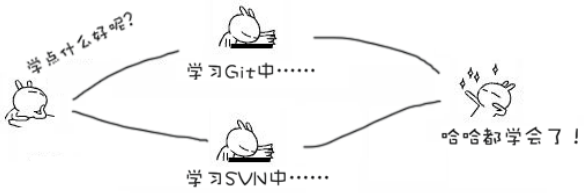
撤销修改

修改提交流程：工作区→暂存区→本地仓库→远程仓库

用途	指令	备注
撤销工作区修改	git checkout -- 文件名	
撤销暂存区修改	git reset HEAD 文件名	
删除仓库文件	git rm 文件名	如果删除目录，需要在目录前添加-r参数
撤销删除	git restore 文件名	
撤销删除	git checkout -- 文件名	恢复删除的文件到工作区

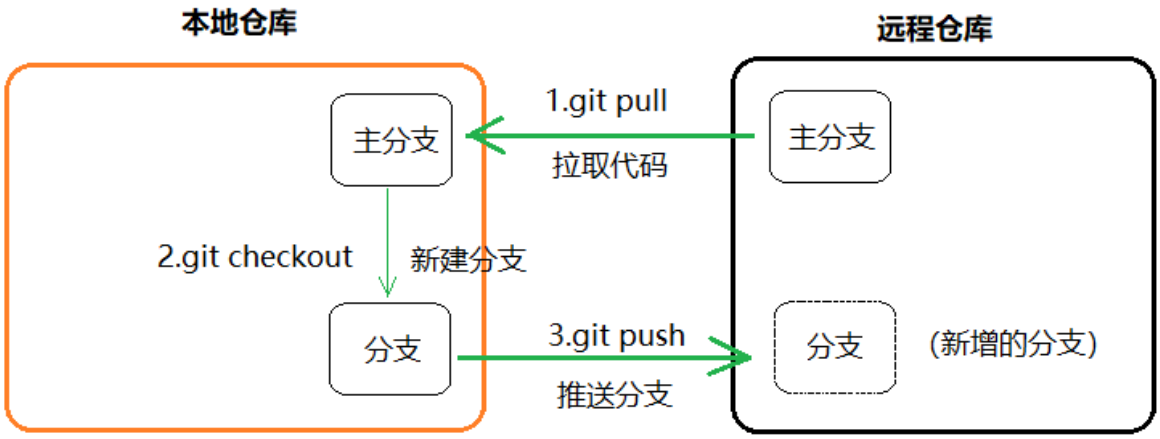
分支管理

Git中有分支的概念，分支之间不会互相产生影响，通常开发者在各自分支中提交代码，最终全部合并到主干。这样可以避免开发者之间的代码影响。



详情参考：https://www.liaoxuefeng.com/wiki/896043488029600/900003767775424

新建分支的流程如下：



常用命令

用途	指令	备注
查看分支列表	git branch	* 标记的是当前工作分支
创建分支并切换	git checkout -b 分支名	加上-b参数表示将工作分支切换到此分支
创建分支	git switch -c 分支名	
切换分支	git switch 分支名	同：git checkout 分支名
合并分支	git merge 分支名	将目标分支合并到当前工作分支，并删除目标分支信息
合并分支	git merge --no-ff -m "说明" 分支名	保留分支信息，并添加说明
删除分支	git branch -d 分支名	-d未合并不能删
暂存当前分支	git stash	临时切换分支前，不提交当前工作分支，仅进行暂存
查看暂存分支列表	git stash list	
恢复暂存	git stash pop	恢复后，删除暂存
恢复暂存	git stash apply	恢复后，保留暂存

标签管理

版本ID很长，我们可以为版本打标签，使用标签了指定某个版本。

用途	指令	备注
为当前版本添加标签	git tag v1.0	标签为 v1.0
为指定版本添加标签	git tag v0.9 1094adb	
查看当前分支标签列表	git tag	
根据标签查看版本	git show v0.9	查看标签 v0.9 对应的版本
推送指定标签到远程	git push origin v1.0	
推送所有标签到远程	git push origin --tags	
删除本地标签	git tag -d v0.1	
删除远程标签	git push origin :refs/tags/v0.9	先删除本地标签

其它

SSH 秘钥

> GitHub 支持 ssh 协议，它可以通过公钥来判断推送的文件是由你推送的。
> 密钥：一台主机创建一个即可，一次创建一直可用，与仓库无关。

本地 Git 仓库和 GitHub 仓库之间的传输通过 SSH 加密,因此需要设置密钥：

```PowerShell
#切换到ssh
$ cd ~/.ssh
#生成密钥
$ ssh-keygen -t rsa -C "你的邮箱"
```

> 注意：不要起名（id_rsa 除外），不要设置密码，因为设置名字后，你的名字与 ssh 内置名字不一致，导致连接错误。
> 在生成密钥的时候，请在 “ ~/.ssh/ ”目录下操作。或者生成后把文件移动到“ ~/.ssh/ ”目录下。

如果非要起个性名字：

```PowerShell
#家目录 .ssh 里面添加 config 文件（不要后缀）
#配置路径和私钥的路径
Host github.com
  HostName github.com
  IdentityFile ~/.ssh/密钥文件名
#可创建多个gethub、gitee等
```

查看加入的密钥列表：

```PowerShell
$ ssh-add -l
```

2.登陆 GitHub，打开 https://github.com/settings/keys 页面添加密钥文件 xxx.pub 内容（多台机器，添加多个）。

3.验证是否可用：

```PowerShell
$ ssh -T git@github.com
#有时候会提示你验证成功但githup不支持shell，是因为你的key是http的
```

多人协作

指令	用途
git remote	查看远程库信息，加-v 显示详细（显示推送地址）
git push origin 分支名	推送到分支，master 为主分支

你的小伙伴克隆一个远程库上的 master，此时你已经创建了一个 dev 分支，他想在 dev 上开发，需要创建远程 dev 到本地，他就可以随意在 dev 上推送了

```PowerShell
#创建远程dev到本地
$ git checkout -b dev origin/dev
```

如果你在他提交前已经对dev做了一次提交，他再提交，就会产生冲突，此时他需要将dev上的最新提交拉到本地，然后在本地合并，（如果冲突，手动修改）再提交。

> 需要设置本地dev与远程dev分支的链接，否则无法拉取。

```PowerShell
#关联远程dev和本地dev分支
$ git branch --set-upstream-to=origin/dev dev
#拉取origin/dev最新提交
$ git pull
```

Rebase：

必须得`git pull`后合并再推送，此时提交的历史就会出现一个分叉...靠哇！后推送的小伙子再看提交历史不就辣住胯了？

如何让这个分叉变成直线：推送之后，使用命令`git rebase`即可。

自定义 Git

指令	用途
$ git config --global color.ui true	让Git显示颜色

文件排除

在Git工作区的根目录下创建一个特殊的`.gitignore`文件，然后填入忽略的文件名（可以下载一个标准文件，再添加自己的即可）。

内容示例：https://github.com/github/gitignore 提供一些自动生成的范例。

```PowerShell
# Windows:如winddows自动生成的文件
Thumbs.db
ehthumbs.db
Desktop.ini
#加入自己定义的排除文件
a.txt
b.md
```

windows注意：资源管理器里新建会提醒你输入文件名，可以在编辑器**另存为**解决。

如果想添加已经被忽略的文件：

```PowerShell
#强行加入
$ git add -f 文件名
#查看忽略位置，然后修改即可，查看指令如下：
$ git check-ignore -v 文件名
```

配置别名

指令单词太长、太难记辣住胯？起个别名直接用别名就行了：

```PowerShell
#--global表示全局设置，所有仓库可用，默认当前仓库
$ git config --global alias.新词 旧词
```