■版本控制的历史

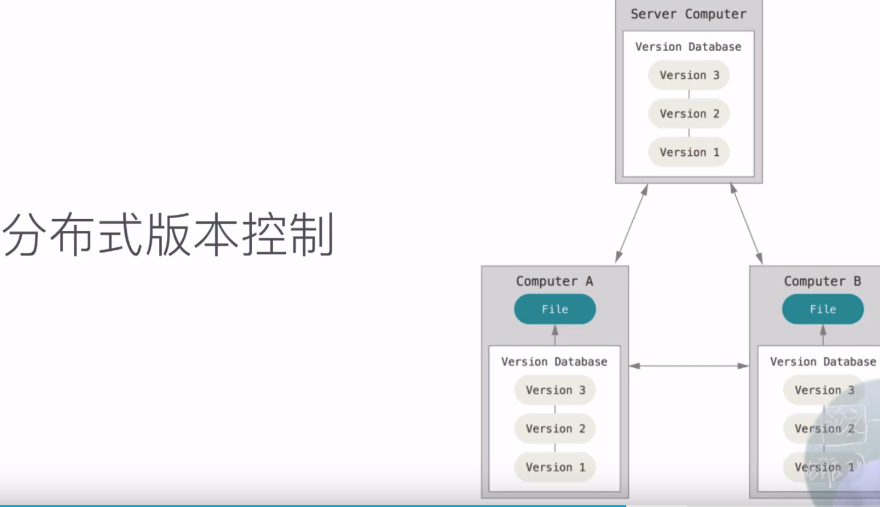
●集中式

以前是集中式，如SVN



●分布式

现在主流是分布式，如GIT



●集中式和分布式的不同点

集中式，每台终端上只存一个版本。

分布式，每台终端上存多个版本。

■安装git

安装的是版本控制软件，git

●

Git官网

<https://git-scm.com/>

<https://git-scm.com/book/en/v2>

安装git

<https://git-scm.com/book/zh/v2/%E8%B5%B7%E6%AD%A5-%E5%AE%89%E8%A3%85-Git>

●安装之后，鼠标右击能看到如下选项，说明安装成功



点击Git Bash Here 或者在命令行里输入

git –version

可查看git版本

■基本操作

●

例子，如对一个项目文件夹gittest进行版本管理，首先要在此文件夹目录下执行Git Bash Here或打开命令行

执行

git init

之后会在此文件夹里生成一个隐藏文件加，名为.git

●对项目文件夹内的文件进行更新后，执行

◎查看更新状态

git status

◎将工作区变更的文件添加到暂存区

将变更的单个文件添加到暂存区

git add 文件名

添加变更的所有文件

git add .

添加变更的所有文件，包括删除的文件

git add -A

◎将缓存区的文件推到版本库，生成一个新版本

git commit -m '版本信息'

执行此命令生成一个新的版本。此版本包括之前添加到暂存区中的所有变更的文件。

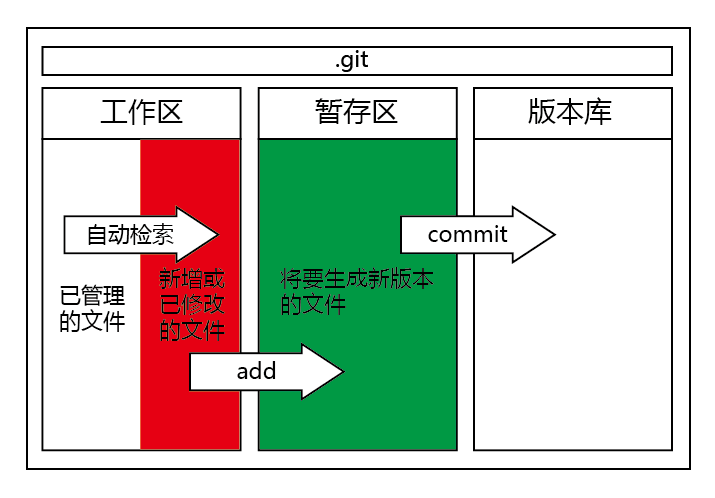
◎再次查看是否想要更新的内容都已生成新版本

git status

●查看版本记录

git log

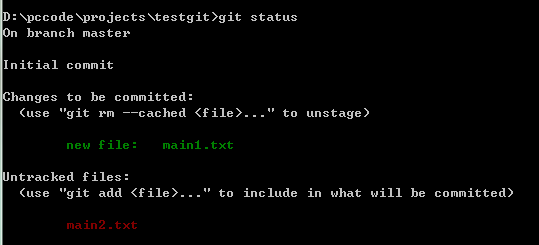
●小结



●使用git status

红色的是工作区新变更的文件。

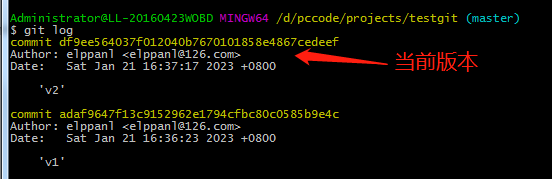
绿色的是暂存区的文件。



█回滚

●回滚到之前某个版本

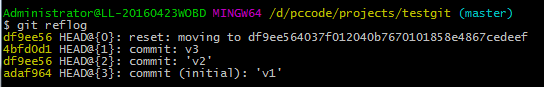
使用 git log查询版本历史，最上面的是当前的版本



使用 git reset –hard 版本号来回滚到想要的版本

git reset --hard adaf9647f13c9152962e1794cfbc80c0585b9e4c

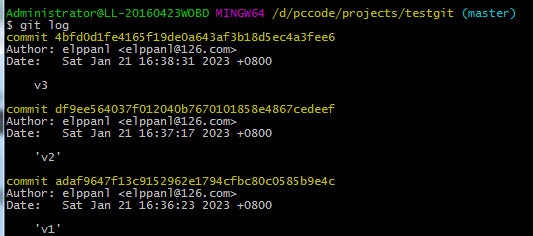
●回滚后，想要再回到之前的最新那个版本，用git log是查不到的。需要用 git reflog



想要回到之前最新版本v3，输入v3行首那个版本号即可

git reset --hard 4bfd0d1

再用git log查看，最上面已经是v3版本了。如下图



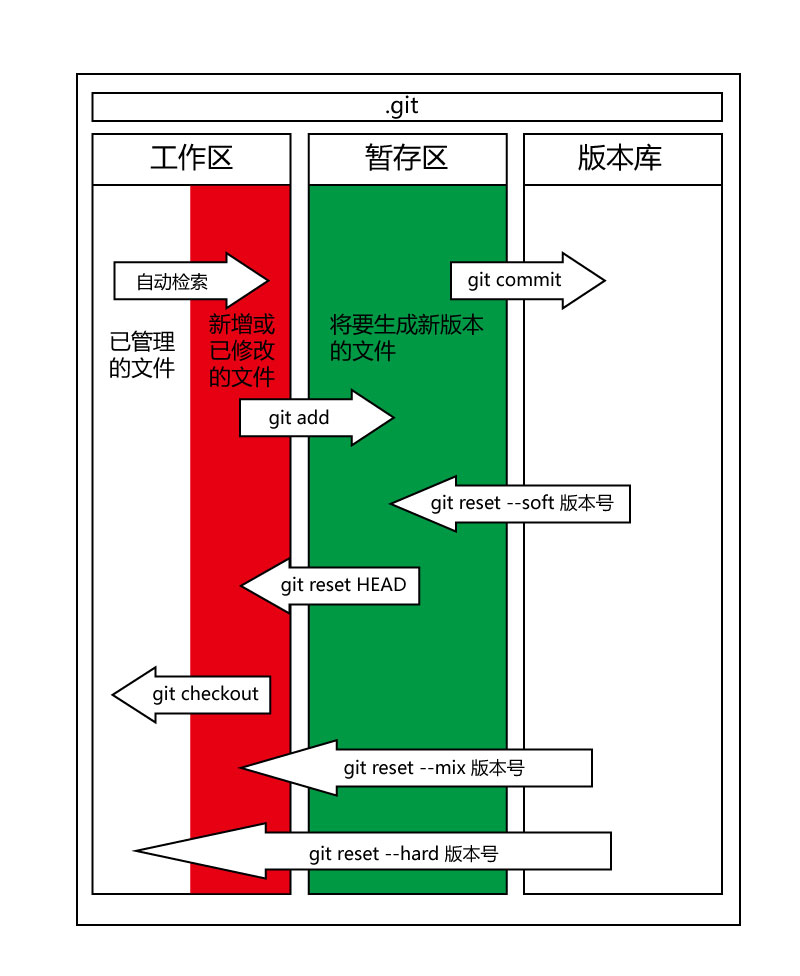
●将暂存区的文件回滚到工作区（绿色变红色）

git reset HEAD 文件名

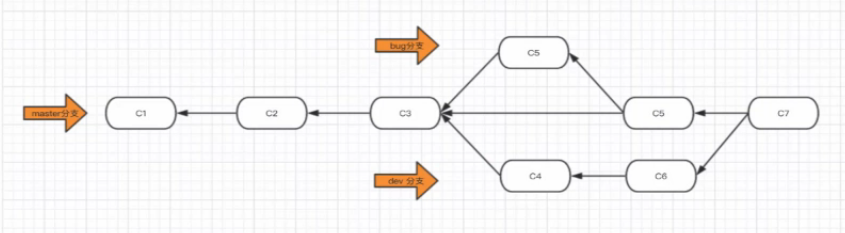
●将工作去新变更的文件，恢复到变更前的状态

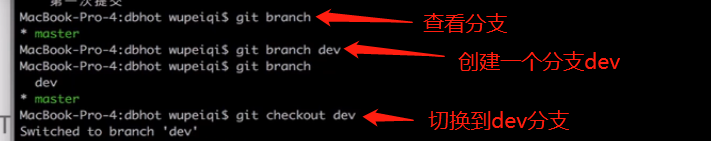
git checkout 文件名

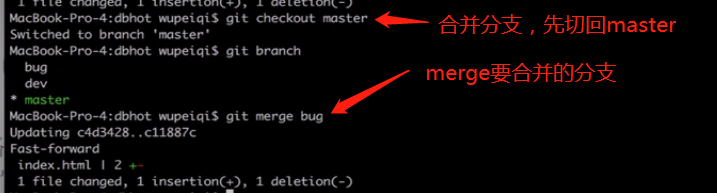
█阶断总结

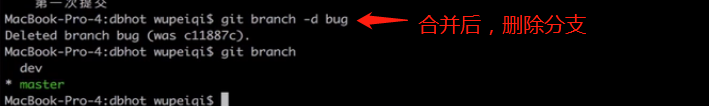


█分支

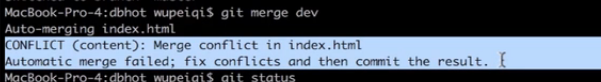








产生冲突





●git 工作流规范

项目开始在master，简单的基本框架放到版本1。

然后就建立dev开发分支。开发都在dev上。等测试通过上线后，在合并到master分支，形成一个大版本。Master分支上永远都是一个一个的大版本。而开发永远都在dev分支上进行。

█github

一般用github，码云等。大公司用gitlab自己建立远端仓库。

●搭建远程仓库

先注册一个github账号。然后创建一个空仓库。

然后进入到本地开发好的项目文件夹，打开命令行输入 git remote add origin 远程仓库地址（给远程仓库起个别名，origin可自定义。方便以后使用）

然后输入 git push origin 分支名 （向远程仓库推送代码）

第一次push我用的以下代码

git push –set-upstream origin master

●从远程仓库克隆代码

git clone 远程仓库地址

然后开发

切换到dev分支 git checkout dev

刚拉下来代码，让dev分支同步最新的master分支代码 git merge master

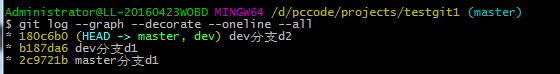
提交和拉取代码

git pull origin 分支名

git push origin 分支名

█查询详细的log信息

git log --graph --decorate --oneline --all



（HEAD -> master, dev） 表示在master分支，将dev分支merge过来

--graph将生成提交树,并 --pretty=oneline为每个提交创建一行可视化

■git配置文件

用户设置，设置或修改用户名和邮箱

<https://blog.csdn.net/u010274449/article/details/124179992>

●配置文件的位置

查看系统config

git config --system --list

查看当前用户（global）配置

git config --global --list

查看当前仓库配置信息

git config --local --list

◎

/etc/gitconfig：包含了适用于系统所有用户和所有项目的值。注是git的安装目录(Win：D:\Git\mingw64\etc\gitconfig) --system 系统级

◎

~/.gitconfig：只适用于当前登录用户的配置。(Win：C:\Users\Administrator.gitconfig) --global 全局

◎

位于git项目目录中的.git/config：适用于特定git项目的配置。–local当前项目

◎例如找user和邮箱信息，先在项目中找，如果没有就到global找。一级一级向上找。如果都没有，就会提示输入用户名和邮箱。

■设置免密登录

Github从2021年8月起不支持密码登录了。这样在pull 或push时，提示输入用户名和密码后，会报错。

●解决方法是在github官网生成一个token,然后输入到密码的位置

◎生成token

github.com官网右上角用户头像 – settings – developer settings – personal access tokens – tokens(classic) – generate new token(classic)

◎本地配置

打开配置文件： 项目\.git\config

[remote "origin"]

    url = https://elppanl:ghp\_lC1rHo6K0vxZH4BFiOL7mCITkrkwXl2QKQ46@github.com/elppanl/python-notes.git

    fetch = +refs/heads/\*:refs/remotes/origin/\*

原url是：

url = https://github.com/elppanl/python-notes.git

将用户名加冒号加token加@号，插入到原地址开头

url = https://elppanl:ghp\_lC1rHo6K0vxZH4BFiOL7mCITkrkwXl2QKQ46@github.com/elppanl/python-notes.git

这样在pull或者push就不用提示输入密码了，直接成功

不过这种方法，如果两个项目用的两个github账号，一个生成token后，成功。然后另一个生成token后，成功。前一个就不能用了。

●通过SSH公钥

◎本地命令行输入

ssh-keygen

之后会提示自定义一个 passphrase 类似于密码的东西，自己记住了

◎然后到

C:\Users\Administrator\.ssh

里面会有两个文件

id\_rsa.pub 为公钥

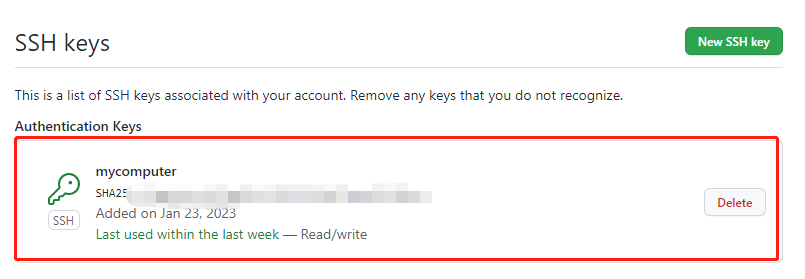
id\_rsa 是私钥

打开id\_rsa.pub文件，复制里面的内容。

◎然后到github.com – settings – SSH and GPG keys

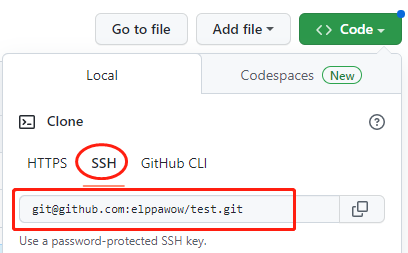
点击 New SSH key 将复制好的公钥粘贴到这里

成功后会生成



◎最后回到本地项目配置文件，打开配置文件： 项目\.git\config

将url地址换成如下



■解决github不稳定问题

<https://www.cnblogs.com/zyfenblog/p/16624378.html>

<https://tool.chinaz.com/dns?type=1&host=github.global.ssl.fastly.net&ip>=

<https://blog.csdn.net/weixin_47666981/article/details/121302531>

●

报错：

[OpenSSL](https://so.csdn.net/so/search?q=OpenSSL&spm=1001.2101.3001.7020) SSL\_read: Connection was reset, errno 10054

解决：

在配置文件（git项目目录中的.git/config）中加入如下两句

[http]

    sslverify = false

[https]

    sslverify = false