

git 基础

git概念相关链接:

30分钟新手git教程 <https://www.cnblogs.com/mjbin/p/5820942.html>

Git使用方法 <https://blog.csdn.net/xukai0110/article/details/80637902>

B站Git教程视频 <https://www.bilibili.com/video/BV1hf4y1W7yI>

为什么要用git

- 1.现实编码中涉及好多修改,你记不住。-git可以跟踪历史版本
- 2.可以更好进行团队协作开发-提交历史,版本修改,代码冲突问题.

版本控制工具svn与git区别:

svn集成式:SVN在没有联网的时候是拒绝干活的

git分布式:在本地工作完全不需要考虑远程库的存在,也就是有没有联网都可以正常工作.当有网络的时候,再把本地提交推送一下就完成了同步

github作用:国外远程仓库 github官网 <https://github.com>

码云:国内远程仓库 码云官网 <http://git.oschina.net/>

操作git可视化工具有哪些

可视化-SourceTree,vs code集成好了。

命令行工具(推荐)-

- 1.git bash
- 2.cmd:按下键盘上windows窗户,输入cmd 或者直接在文件url上输入cmd
- 3.powershell:shift+鼠标右键,找powershell

基本dos命令操作:

mkdir 创建文件夹

cd -进入某个目录 cd ../ 上一个目录 cd ./当前目录

cls-清屏

ctrl+c强制退出 wq退出

键盘方向键可以找之前的命令

创建仓库并拉取

码云(国内),github(国外),gitlab(国外)

因为Git是分布式版本控制系统,所以,每个机器都必须自报家门:你的名字和Email地址。如果不想每次提交git都输入用户名和密码,如何操作?

方法1(全局设置用户):

- 1.全局保存用户名和邮箱:

git config --global user.name "Your Name"

git config --global user.email "email@example.com"

方法2(ssh公钥):

- 1.生成ssh公钥: ssh-keygen -t rsa -C '邮箱地址' ,生成地址一般默认是在 C:\Users\dell.ssh\id_rsa.pub

- 2.查看公钥: cat ~/.ssh/id_rsa.pub ,查看后复制到自己的Git帐号的SSH设置中(!!!注意查看命令,要在git bash里才有效)

4.查看所有配置项，看看有没有user.name,user.email这两个配置

git config --list

5.最后克隆个仓库试下(!!!克隆的时候,会让你输入git仓库的登录账号与密码,输入这1次就可以):

1.克隆所有: git clone 仓库地址

或者克隆指定分支: git clone -b dev 仓库地址 (dev是分支名称)

git基本操作流程

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

\$ git clone <url> #克隆远程版本库
\$ git init #初始化本地版本库

修改和提交

\$ git status #查看状态
\$ git diff #查看变更内容
\$ git add . #跟踪所有改动过的文件
\$ git add <file> #跟踪指定的文件
\$ git mv <old> <new> #文件改名
\$ git rm <file> #删除文件
\$ git rm --cached <file> #停止跟踪文件但不删除
\$ git commit -m "commit message" #提交所有更新过的文件
\$ git commit --amend #修改最后一次提交

查看提交历史

\$ git log #查看提交历史
\$ git log -p <file> #查看指定文件的提交历史
\$ git blame <file> #以列表方式查看指定文件的提交历史

撤销

\$ git reset --hard HEAD #撤销工作目录中所有未提交文件的修改内容
\$ git checkout HEAD <file> #撤销指定的未提交文件的修改内容
\$ git revert <commit> #撤销指定的提交

分支与标签

\$ git branch #显示所有本地分支
\$ git checkout <branch/tag> #切换到指定分支或标签
\$ git branch <new-branch> #创建新分支
\$ git branch -d <branch> #删除本地分支
\$ git tag #列出所有本地标签
\$ git tag <tagname> #基于最新提交创建标签
\$ git tag -d <tagname> #删除标签

合并与衍合

\$ git merge <branch> #合并指定分支到当前分支
\$ git rebase <branch> #衍合指定分支到当前分支

远程操作

\$ git remote -v #查看远程版本库信息
\$ git remote show <remote> #查看指定远程版本库信息
\$ git remote add <remote> <url> #添加远程版本库
\$ git fetch <remote> #从远程库获取代码
\$ git pull <remote> <branch> #下载代码及快速合并
\$ git push <remote> <branch> #上传代码及快速合并
\$ git push <remote> :<branch/tag-name> #删除远程分支或标签
\$ git push --tags #上传所有标签

Git Cheat Sheet <CN> (Version 0.1)

2012/10/26 -- by @riku < riku@gitcafe.com / http://riku.wowubuntu.com >

哎哟喂学长

本地仓库推到远程仓库(!!!注意仓库里不能有空文件夹,不能有多个.git文件):

1.注意推之前先拉取

git init 初始化下本地仓库

git remote add origin 仓库地址 关联远程库

git pull origin master 拉取分支名

git add 文件名 提交到暂存区 【git add -A 或git add . 提交本地全部文件】

git status 查看提交状态

git commit -m '提交说明' 提交说明

git push origin master 提交到远程库

//备注用

git push --set-upstream origin master (省略形式为: git push -u origin master) 或者先关联远程库 git remote add origin 仓库地址 然后每次提交用 git push origin master 提交到版本库

团队合作开发:

分支:每个人不同的模块,创建自己的分支

1.先拉取远程仓库分支: 查看本地分支与远程分支是否一致

2.然后把自己分支推送到远程仓库

3.合并分支:每个人加入到对方的分支中



分支操作基本流程:

新建分支 → 分支上开发(写代码) → 提交 → 合并到主分支

2.1 创建分支合并步骤:

1. `git branch kuige` 创建分支名
- `git branch` 查看当前分支
- `git checkout kuige` 切换到创建的分支上
- `git merge master` 把kuige合并到master上(别在原来分支)
- `git push origin kuige` 推到远程子分支
- `git push origin master` 推到远程主分支 注意!!! `git pull origin master` 推之前先拉取分支

2.2 分支其它命令

1. 查看所有分支
`git branch -a`
2. 查看本地分支
`git branch`
3. 创建并切换分支 (加-b表示创建并切换)
`git checkout -b 分支名`
4. 切换回分支
`git checkout 分支名`
5. 从主分支里创建新分支
`git checkout master -b 新分支名`
6. 合并某分支到当前分支
`git merge 分支名`
7. 删除分支-不要在当前分支下, 否则删除不了
`git branch -d 分支名`
`git branch -D 分支名` (强行删除分支)
8. 删除远程分支
`git push origin --delete kuige`

合并冲突问题

2.3 本地合并冲突-分支与分支之间冲突(修改了同一个地方,git不知道以谁为主? 一般以最后提交的为主)

1.问题描述(冲突的地方head代表当前分支, login是另一个分支):

《 《 《 《 《 Head

aa

bb

》 》 》 》 》 login

2.解决办法-把多余的删掉, 留下最新的

3.然后 git add. , git commit ,git merge 最后合并另一个分支到当前分支

2.4远程合并冲突-多个分支向同一个远端分支推送代码时

跟本地冲突解决一样,修改的时候, 小心点, 沟通下。

2.5 回滚问题,冲突问题:

git log 查看版本号 git reflog 查看所有历史信息, 包括删除

git reset --hard 版本号(不用复制全,可以是前7位字母) 回滚到某个版本号
或者下面

git reset --hard HEAD 回滚到最新版本

git reset --hard HEAD~1 回滚到相比于最新的其次1个版本

git push -f origin kuige1 强制删除远程分支(删除24小时内的)

git diff 提交历史号 可以查看冲突地方

自己手动解决冲突问题,一般以线上的版本为主

git add 这些在提交一遍

git的三个区概念:

工作区:你自己电脑里能看到的目录,工作的文件

暂存区:工作区需要提交的文件修改通通放到暂存区, 然后, 一次性提交暂存区的所有修改到远程稳定版本区

优点:

作为过渡层

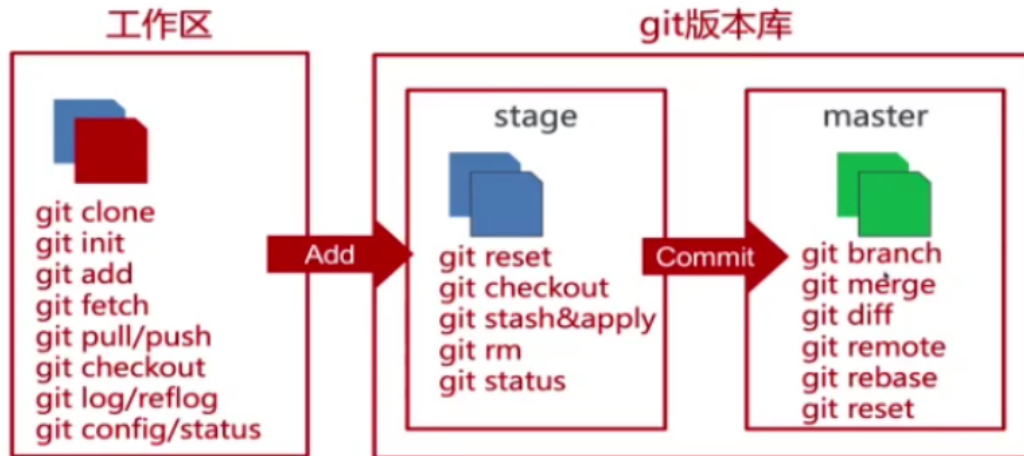
避免误操作

保护工作区和版本区

分支处理

版本区 (库):稳定版本区, 远程仓库

Git基础



常见问题:

1.出现那段红色的报错，这个问题主要是因为远程库与本地库不一致造成的，那么我们把远程库同步到本地库就可以了。使用指令

`git pull --rebase origin master`

```
D:\新建文件夹\实训一\03>git push origin master
To https://gitee.com/lz_bababiba/four-groups-day01.git
! [rejected]        master -> master (fetch first)
error: failed to push some refs to 'https://gitee.com/lz_bababiba/four-groups-day01.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
```

2.冲突问题-出现CONFLICT时

```
$ git pull
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 3 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), done.
From github.com:yanqun/git2019
   9ec53ba..c0dd23b  master    -> origin/master
Auto-merging a.txt
CONFLICT (content): Merge conflict in a.txt
Automatic merge failed; fix conflicts and then commit the result.
```

MINGW64:/c:/Users/YANOUN/Desktop/mvait2/mvait2

```
<<<<<< HEAD
ia second person 2222
=====
ia111111
>>>>>> c0dd23b3cc5b403575b35a86c3c791f9841ec6f2
hello git ...first ...
这是第二个用户..
nit...
hello second
hello
4444world
local a
~
```

冲突问题解决: <https://www.cnblogs.com/bobo1/p/12639095.html>