2025/11/17 23:30 1/3 git

GIT 是一个分布式的版本管理工具

## ssh远程登录不输入密码

执行ssh-keygen -t rsa 生成.ssh目录

进入目录.ssh下,将id\_rsa.pub里的内容添加到远程的机子.ssh下的authorized\_keys文件中(文件不存在的话就创建一个)这样在远程登录的时候就不用输入密码

```
ssh-keygen -t rsa -C "youremail@example.com" github上介绍使用的方法 不用加email也可以
```

安装git[]用它的图形界面gitk[]需要tcl, tk的支持, 在gitk里面的File栏有Start git gui[]

# git init 初始化方法

- 1. git init (个人使用)
- 2. git --bare init □□初始化成裸库
- 3. git --bare init sample.git 生成本地sample库目录,并且初始化好

## 添加远程库

本地已建有库,把本地库推送到github上(远程库需要建立对应库),本地库和远程库相互关联.

```
git remote add origin git@github.com:xxx/xxx.git
git push --set-upstream origin master
```

# 克隆远程库

```
git clone git@github.com:xxx/xxx.git
```

git默认拒绝了push操作,需要进行设置,修改.git/config 或者.gitconfig文件添加如下代码:

```
[receive]
  denyCurrentBranch = ignore
```

别名alias 比如用git st 代替 git status

```
git config --global alias.st status
```

2025/11/17 23:30 2/3 git

```
git config --global alias.co checkout
git config --global alias.ci commit
git config --global alias.br branch
git config --global alias.mr merge
```

### 基本使用

提交修改到暂存区[] git add file1 撤消操作[] git checkout -- file1 从暂存区提交版本到库[] git commit file1 撤消操作[] git reset HEAD file1 查看状态[] git status

版本回退(整体版本回退,最好不要使用单个文件的版本回退,因为比较容易忘记文件是在与哪个版本作对比):

```
git reset --hard HEAD<sup>^</sup>
git reset --hard 版本号 版本号可以只输入前几位□git会自动查找,一般可以输7位
--hard 版本指针回退到指定版本, 文件也回退到指定版本
--soft 版本指针回退到指定版本, 文件保持不变,个人感觉没什么意义
```

HEAD 表示最新版本, HEAD 表示上一个版本, HEAD 个表示上上个版本 HEAD ~ 100表示向上100个版本

```
git reflog 会记录版本的每一次变化(提交&回退等),用于很方便的版本回退工作。
git log
git log --oneline 显示的log信息以单行显示
git log -1 显示最近1次 log
git log -2 显示最近2次 log
```

Git鼓励大量使用分支: 查看分支[git branch 创建分支[git branch <name> 切换分支[git checkout <name> 创建+切换分支[git checkout -b <name> 合并某分支到当前分支[git merge <name> 删除分支[git branch -d <name> git push origin :branch-name (如果把本地分支传到远程分支,删除远程分支需要此句,冒号前空格不可少)

合并分支 通常,合并分支时,如果可能□Git会用Fast forward模式,但这种模式下,删除分支后,会丢掉分支信息。 如果要强制禁用Fast forward模式□Git就会在merge时生成一个新的commit□这样,从分支历史上就可以看出分支信息. git merge --no-ff -m "merge message with no-ff" dev

BUG分支 用于暂存现场的环境, 一般是用于工作干到一半,代码不能提交,但是又需要分析另外一个BUG 使用如下命令 git stash 把现场环境保存起来,本地恢复成干净的分支 git stash pop 恢复刚才保存起来的现场

#### git 全局忽略.svn目录

编辑~/.gitignore\_global文件,添加如下行.svn 忽略.svn目录 \*~ 忽略以~结尾的文件执行命令 git config --global core.excludesfile ~/.gitignore global

2025/11/17 23:30 3/3 git

## 搭建Git服务器

准备一台运行Linux的机器

第一步,创建git用户,用来运行git服务[] \$ adduser git

第二步,创建证书登录: 收集所有需要登录的用户的公钥,就是他们自己的id\_rsa.pub文件,把所有公钥导入到/home/git/.ssh/authorized\_keys文件里,一行一个。

第三步,初始化Git仓库:先选定一个目录作为Git仓库,假定是/srv/sample.git□在/srv目录下输入命令□\$git init --bare sample.git Git就会创建一个裸仓库,裸仓库没有工作区,因为服务器上的Git仓库纯粹是为了共享,所以不让用户直接登录到服务器上去改工作区,并且服务器上的Git仓库通常都以.git结尾。然后,把owner改为git□\$chown -R git:git sample.git

第四步,禁用shell登录:出于安全考虑,第二步创建的git用户不允许登录shell[]这可以通过编辑/etc/passwd文件完成。找到类似下面的一行[] git:x:1001:1001:,,,:/home/git:/bin/bash 改为[] git:x:1001:1001:,,,:/home/git:/usr/bin/git-shell 这样[]git用户可以正常通过ssh使用git[]但无法登录shell[] 因为我们为git用户指定的git-shell每次一登录就自动退出。

第五步,克隆远程仓库: 现在,可以通过git clone命令克隆远程仓库了,在各自的电脑上运行□\$ git clone git@server:/srv/sample.git 剩下的推送就简单了。

# git配置文件

.gitconfig

```
[push]
    default = current
[receive]
denyCurrentBranch = ignore
[alias]
    co = checkout
    ci = commit
    br = branch
    st = status
   mr = merge
[user]
    name = zhangguo1286
    email = zhangquo1286@163.com
[core]
    excludesfile = ~/.gitignore global
[http]
    postBuffer = 10737418240
```