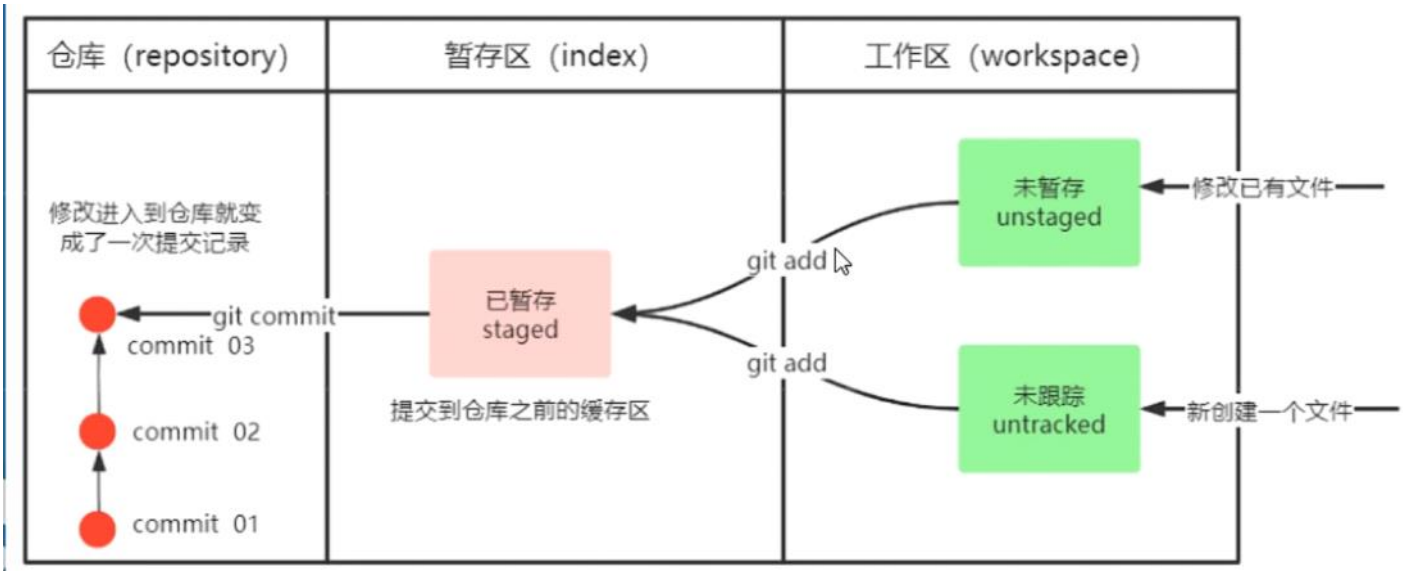
Git

2022年8月28日 12:44



1. 创建工作区 git init
2. 创建文件 touch file.txt
3. 查看状态 git status

```
Fine@LAPTOP-FHD MINGW64 ~/Desktop/新建文件夹 (master)
$ git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        file.txt
        file01.txt

nothing added to commit but untracked files present (use "git add" to track)
```

文件处于【工作区】【未跟踪】状态

4. 添加到【暂存区】

   git add . //添加所有文件到【暂存区】

   git add file01.txt //添加指定文件到【暂存区】

```
MINGW64:/c/Users/Fine/Desktop/新建文件夹

Fine@LAPTOP-FHD MINGW64 ~/Desktop/新建文件夹 (master)
$ git add file01.txt

Fine@LAPTOP-FHD MINGW64 ~/Desktop/新建文件夹 (master)
$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   file01.txt

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        file.txt


Fine@LAPTOP-FHD MINGW64 ~/Desktop/新建文件夹 (master)
$ git add .

Fine@LAPTOP-FHD MINGW64 ~/Desktop/新建文件夹 (master)
$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   file.txt
        new file:   file01.txt
```

5. 【暂存区】提交到【仓库】git commit -m "add file"

```
Fine@LAPTOP-FHD MINGW64 ~/Desktop/新建文件夹 (master)
$ git commit -m "add file01"
bash: $'\302\223\302\223\302\223git': command not found

Fine@LAPTOP-FHD MINGW64 ~/Desktop/新建文件夹 (master)
$ git commit -m "add file"
[master (root-commit) 9501f74] add file
 2 files changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 file.txt
 create mode 100644 file01.txt
```

6. 查看提交记录 git log

```
Fine@LAPTOP-FHD MINGW64 ~/Desktop/新建文件夹 (master)
$ git log
commit 9501f7436ef8a3800ac88307c9f3da6c2816593c (HEAD -> master)
Author: 范宏达 <fanhongda@sinosoft.com.cn>
Date:   Sun Aug 28 15:35:52 2022 +0800
```

其他参数git log --pretty=oneline --abbrev-commit --all --graph
显示所有分支 git log --all
将提交信息显示为一行 git log --pretty=oneline
使输出的commit ID更简短git log --abbrev-commit
以图的形式显示 git log --graph
显示可引用的历史版本记录 git reflog （可以看到删除的提交记录）

7. 版本回退 git reset --hard 9501f74
   回退之后也可以根据commitID还原回来 记不住commitID 可以使用git reflog
8. 忽略文件
   创建.gitignore文件 touch .gitignore
   编辑内容 *.a 保存（以.a为后缀的忽略）
   再次git status .a后缀的文件已经忽略
9. git命令别名
10. 查看帮助文档 git help

PS：vim编辑器
vi 文件名 编辑文件
ESC退出编辑
:wq保存并退回到目录
:q退出
:q! 退出不保存

分支

1. 查看本地分支 git branch
2. 创建分支 git branch 分支名
3. HEAD->当前分支
4. 切换分支 git checkout dev01
   执行git add 和git commit 之后 切换分支后 其他分支是看不到的
5. 创建分支并切换到新的分支 git checkout -b dev02
6. 合并分支
   切换到要被合并的主分支master git checkout master
   合并分支 git merge dev02
7. 删除分支
   删除分支时做检查 git branch -d dev02
   删除分支时不做检查 git branch -D dev01
   如果要删除的分支没有merge到master 可能会出现删除不成功的情况。
   可以用git branch -D dev01
8. 解决冲突
   不同分支 都对同一文件做出修改

```
Fine@LAPTOP-FHD MINGW64 ~/Desktop/新建文件夹 (master)
$ git merge dev
Auto-merging file.txt
CONFLICT (content): Merge conflict in file.txt
Automatic merge failed; fix conflicts and then commit the result.

Fine@LAPTOP-FHD MINGW64 ~/Desktop/新建文件夹 (master|MERGING)
```

直接手动修改 然后继续 git add . 再git commit -m '提交'

（idea中可直接手动选择合并，更方便）

9. 快进模式

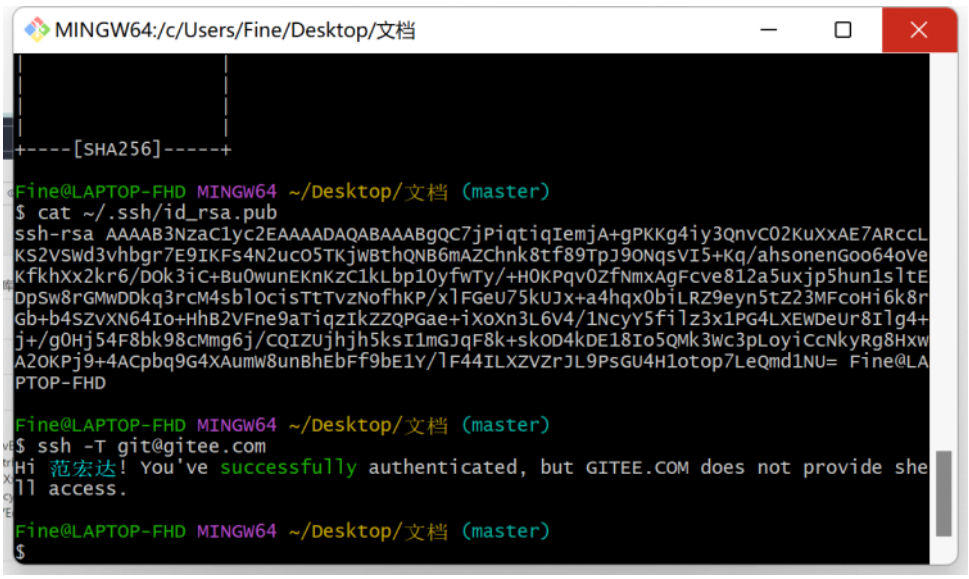
远程仓库

1. 配置SSH公钥

   生成非对称密钥$ ssh-keygen -t rsa

   获取公钥 $ cat ~/.ssh/id_rsa.pub

   查看是否成功$ ssh -T git@gitee.com

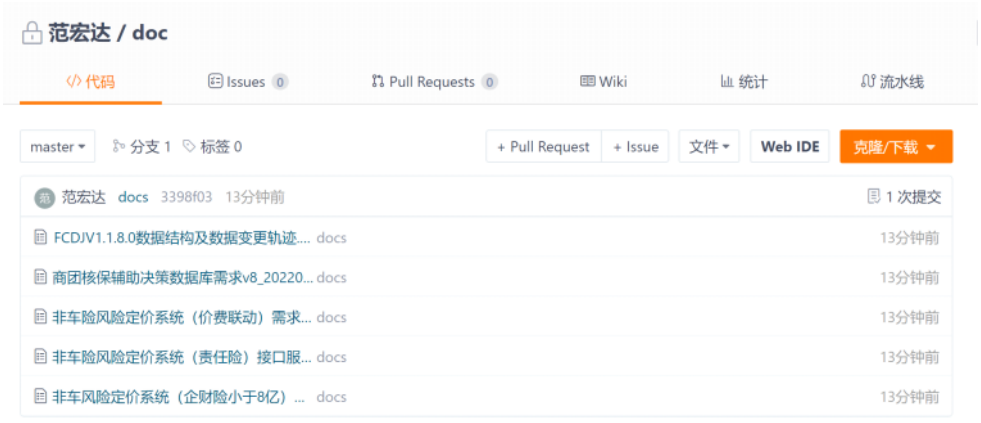


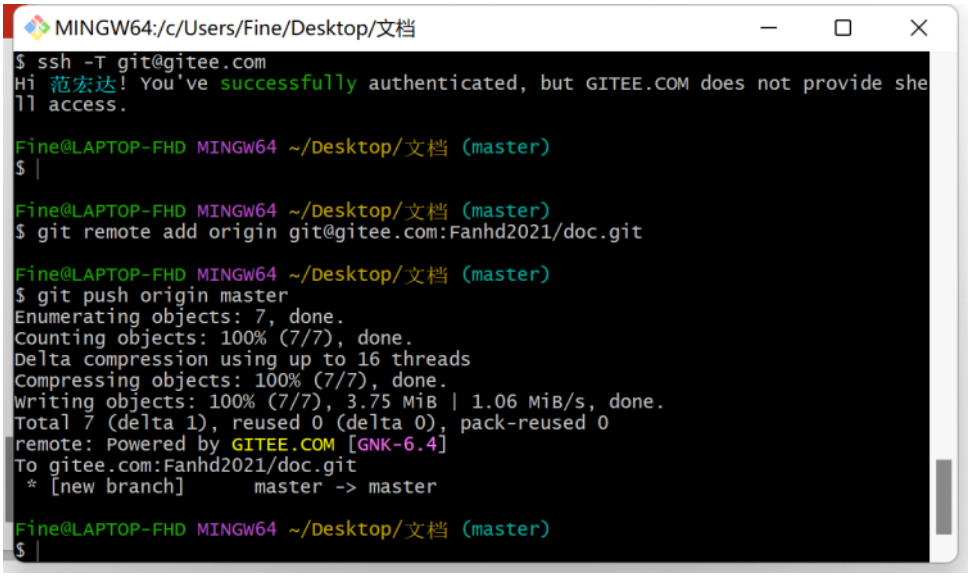
2. 告诉本地仓库对应的远程仓库是哪一个

   $ git remote add origin git@gitee.com:Fanhd2021/doc.git

3. 本地代码推送到远程仓库

   $ git push origin master（将上面命名的origin推到master上）





推送到远程仓库的其他参数

git push [-f] [-set-upstream][远端名称 [本地分支名]:[远端分支名]]

$ git push --set-upstream origin master:master

（ --set-upstream保存绑定关系，下次在master分支push就可以了）

git push origin master （这里origin是本地起好的要推送的远端名。master是本地分支，由于和远程一样所有只写一个就可以了。-f 表示强制覆盖，例如修改了同一个文件产生冲突，使用-f后会直接把远程分支与本地一致。）

4. 查看本地分支与远程分支的对应关系$ git branch -vv
   $ git branch -vv
5. 克隆$ git clone git@gitee.com:Fanhd2021/doc.git [指定的文件夹名不写默认生成一个]
6. 抓取git fetch [remote name] [branch name]
   抓取指令就是将仓库里的更新都抓取到本地，不会进行合并
   如果不指定远端名称和分支名，则抓取所有分支
7. 拉取git pull[remote name] [branch name]
   拉取指令就是将远程仓库的修改拉到本地并自动进行合并，等同于fetch+merge
   如果不指定远端名称和分支名，则抓取所有并更新当前分支。

注意：

1. 切换分支先提交代码，可以先不push，但是一定要先commit。这样切回分支还能保持原来的样子。commit也是可以取消的。
2. 代码常commit就不会丢了，可以先不push。
3. 推送代码时一定要先pull再push
4. idea搁置问题