Git学习

Time：2024-07-09

目 录

[Part A：Python学习 1](#_Toc140504151)

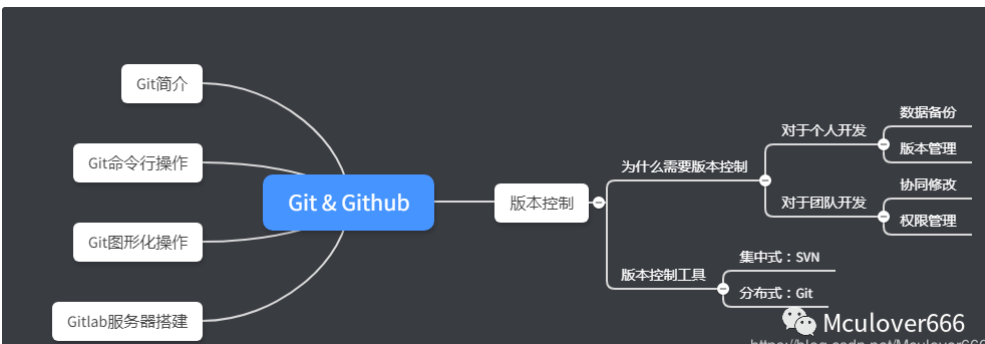
[1.环境的配置与安装 1](#_Toc140504152)

[1.1什么是anaconda 1](#_Toc140504153)

[2.python基础 1](#_Toc140504154)

[2.1数据类型 1](#_Toc140504155)

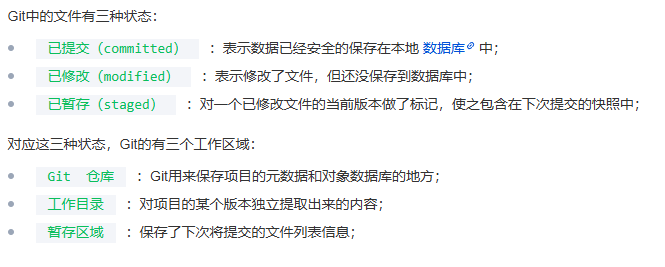
多人协作开发

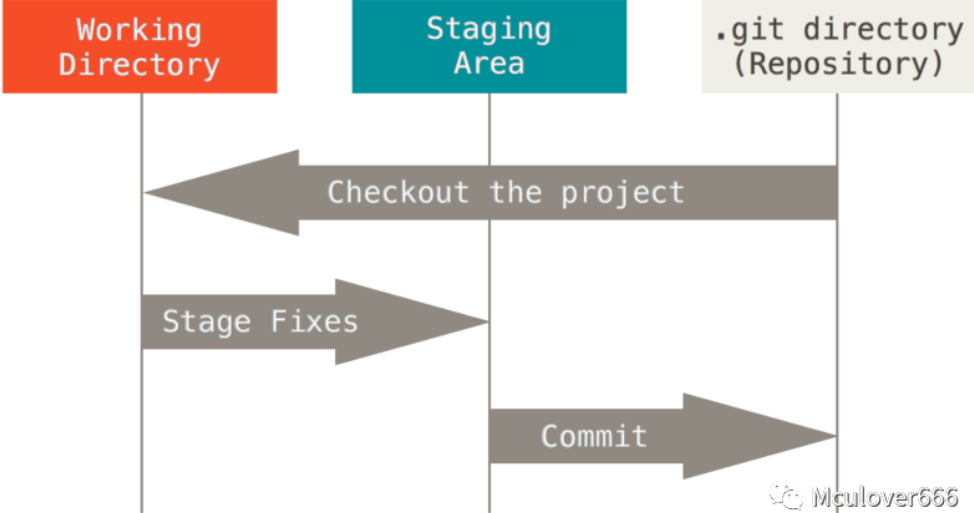


基础介绍

# 基础

## 1.基础介绍





三个工作地：

Working ：自己平时写代码的地方

Staging ：使用命令（commit）暂存在本地仓库

Git仓库：云端仓库

基本流程

本地工作目录修改-》暂存-》提交更新

# 二、基础命令

## 1.获取Git仓库

①初始化本地仓库

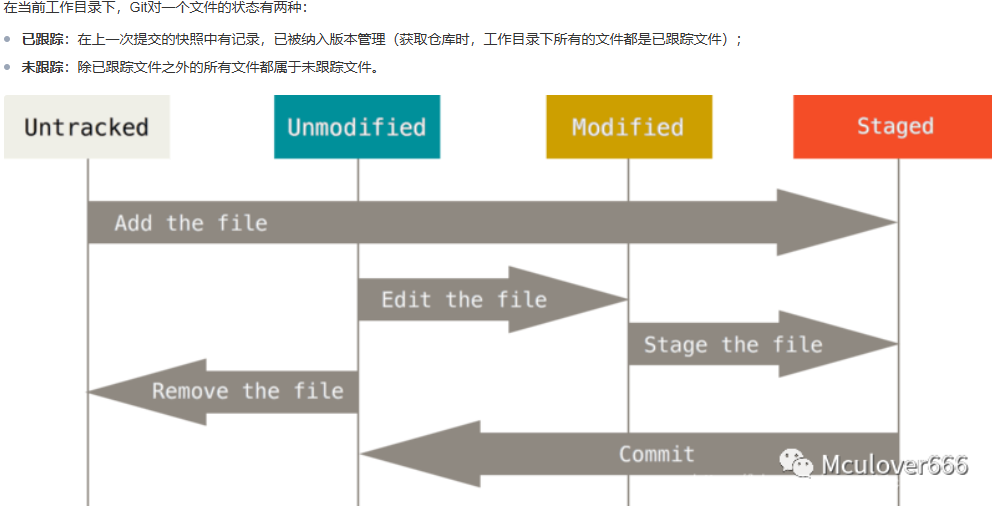
Git init

②克隆（拉取）现有仓库到本地

Git clone URL

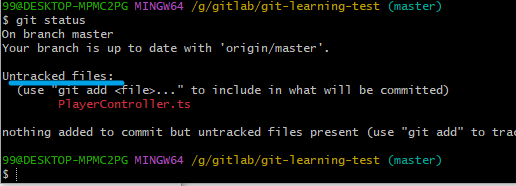
## 2.提交更新★

当前工作目录下Git的文件状态



### 2.1 查看

查看状态：git status



-s列简短信息

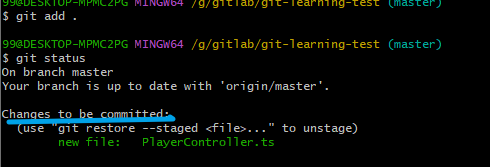




### 2.2 跟踪

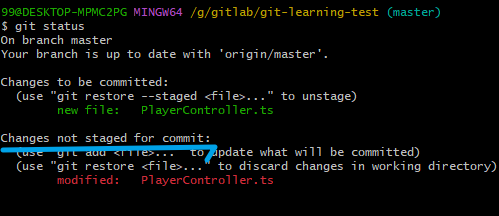
Git add <file/dir>

-A 表示跟踪所有文件

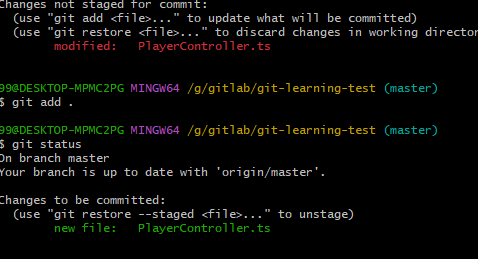


### 2.3 查看暂存已修改

Git add后文件又被修改



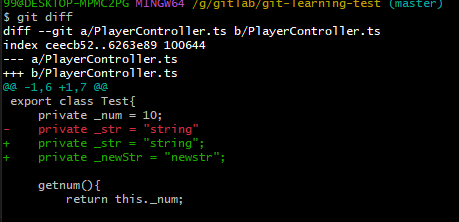
文件修改后没有保存暂存区，使用git add将修改保存到暂存区



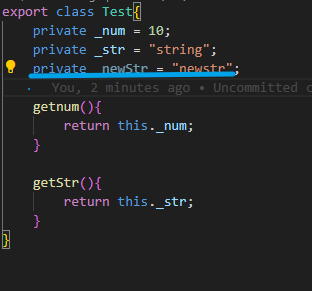
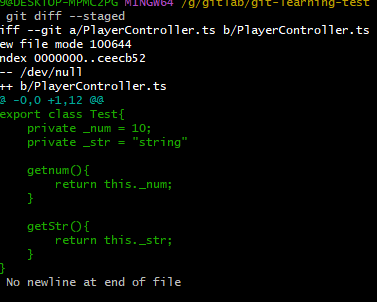
### 2.4 查看未暂存和已暂存的修改

Git diff可以查看具体内容，通过文件补丁方式显示“- +”

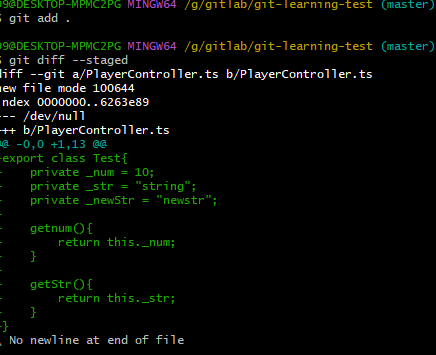
但是如果文件没有被追踪，git diff不比较，添加文件之后记得git add追踪文件



--staged 查看已经暂存的修改命令

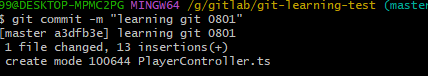
 

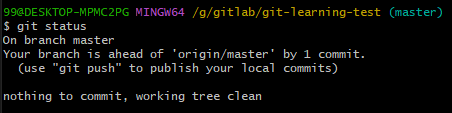
新添加的一行没有暂存，使用git add .添加到暂存区



### 2.5 提交更新

Git commit 将暂存内容提交到本地仓库

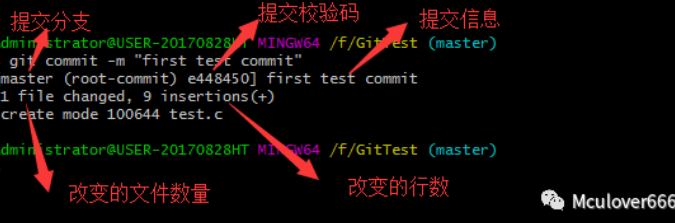




提交后，本地仓库比git仓库领先一个版本。

每次提交都是对该项目的一个快照，在以后的任何时候都可以回退到该次状态。

提交信息解读：

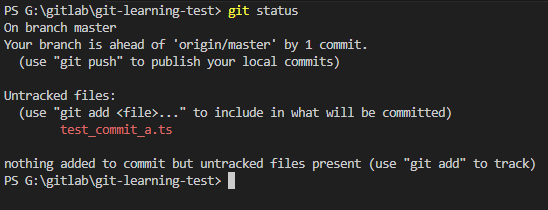


跳过暂存直接更新，git commit -a -m “msg”

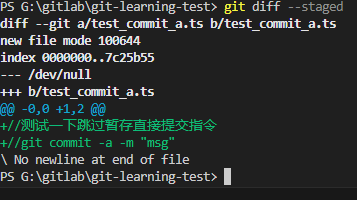
（这里跳过指的是跳过暂存步骤，直接将工作区修改的文件提交到本地仓库，未追踪文件不能提交）

解决了一个问题，每次修改都需要先git add将修改提交，然后再git commit提交到仓库，就很烦

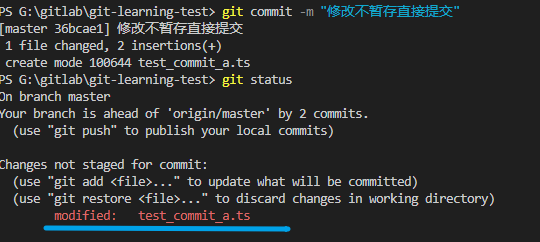
#### （1）提交到本地仓库基本流程



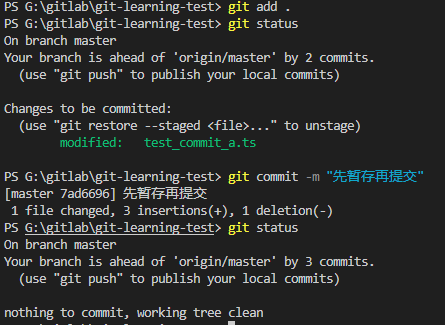
文件未追踪，git add添加追踪到暂存区



加一行后不暂存直接commit，显示本地修改没有提交

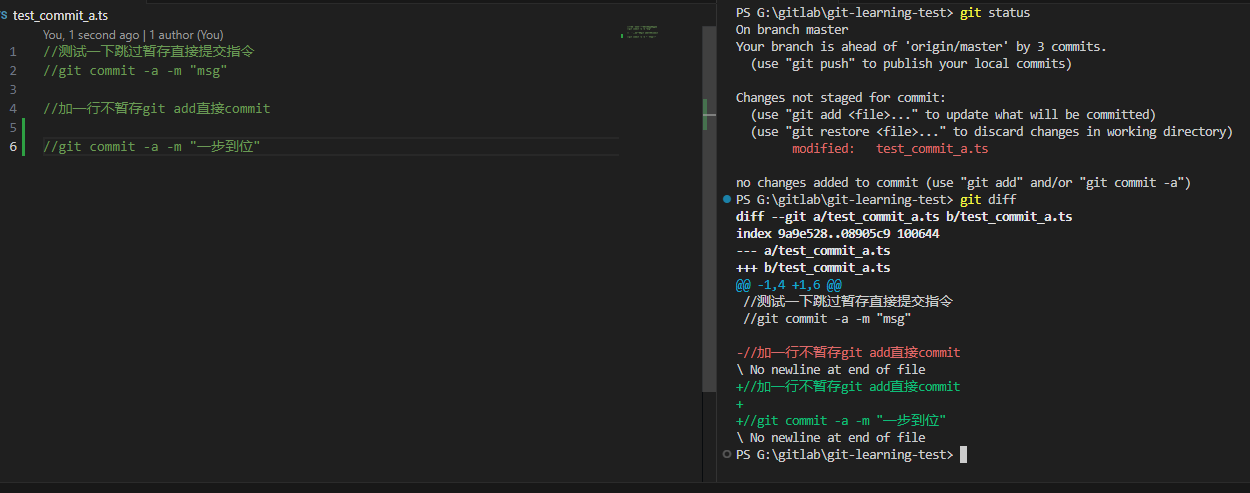


需要先git add暂存再commit

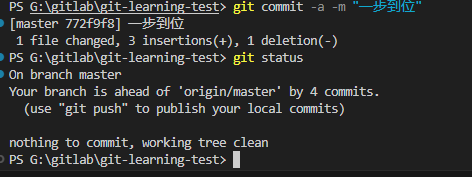


#### （2）一步到位

加一行

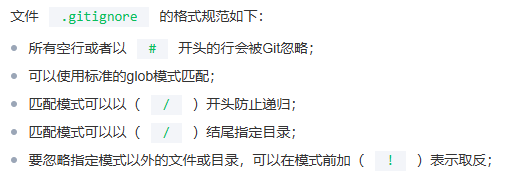


使用git commit -a一步到位



### 2.6 忽略文件

工作目录下创建一个.gitignore文件列出忽略文件



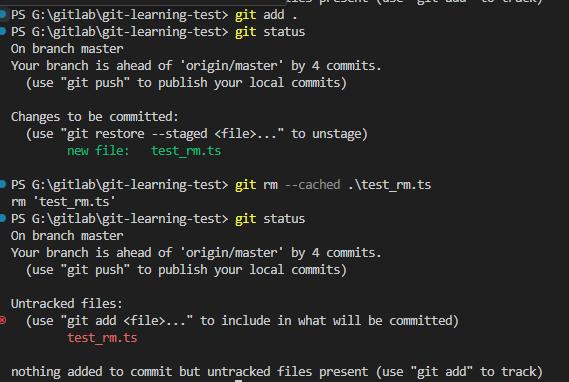
### 2.7 移除文件

从暂存区删除且从工作目录删除

Git rm -f filename

从暂存区删除，但保留工作区源文件

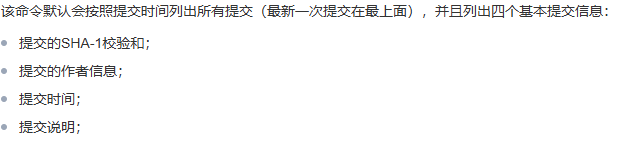
Git rm –cached filename



## 3.本地库操作

### 3.1 提交历史

Git log





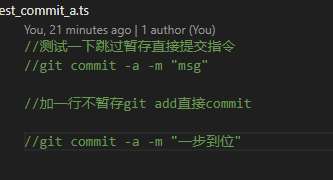
#### （1）输出控制

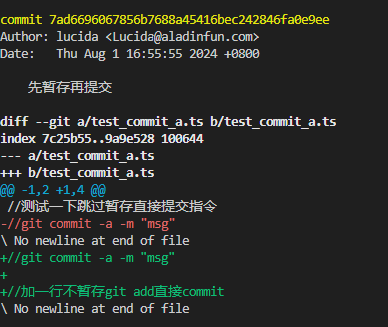


### 3.2 版本前进和回退

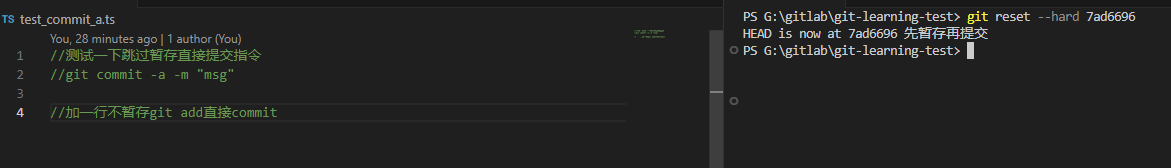
Git reset –hard “索引”

当前内容



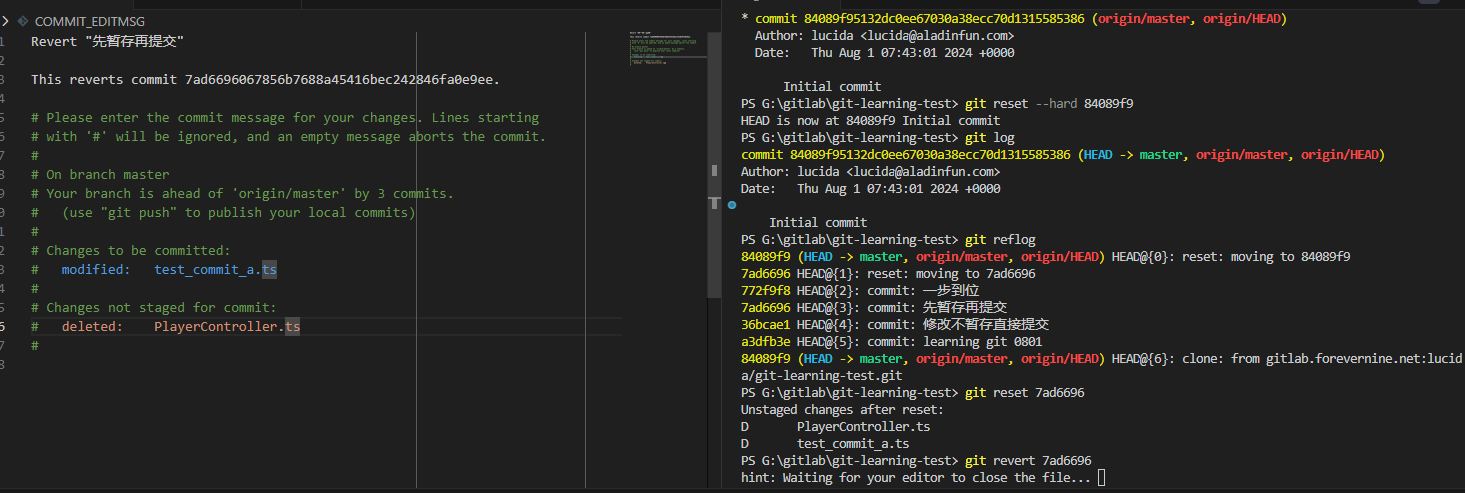


回退到版本



注意只能在git log显示上操作

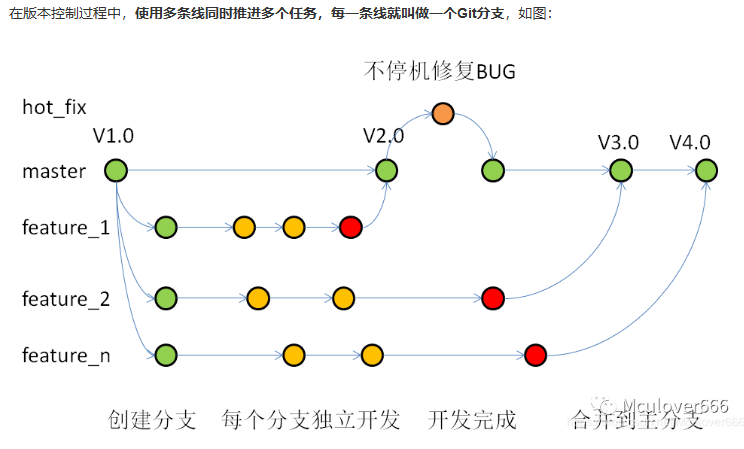
如果reset错了，使用指令恢复，但是错误操作会丢失这段时间的更改



[使用git checkout和git reset回退到历史版本 - 南木阁 - SegmentFault 思否](https://segmentfault.com/a/1190000020863861)

## 4.本地库管理

### 4.1 git分支



### 4.2 操作

每一个远端git都是落后于自身git的，需要自己push保持同步

#### （1）基础操作

创建：git branch name

查看：git branch -v

切换：git checkout branch\_name

#### （2）合并分支★

要将一个分支所做的更新合并：

①切换到要合并的分支上；

②执行命令。

Git merge 待合并分支

一般来讲使用合并前，先看一下本地仓库和Git仓库是否同步，先将本地的修改push到git仓库，然后再将远端仓库拉下来进行合并。如果发现冲突，需要进行手动修改。

建议使用可视化工具， shell命令熟悉即可

#### （3）删除分支

删除本地分支：

Git branch -D name

删除远程分支：

Git push origin –delete name

#### （4）创建分支

创建本地分支：

Git branch -b name //根据当前创建并切换

推送本地分支到远程

Git push origin name

#### （5）本地分支与远程分支关联

[git设置本地与远程分支关联 - coder小白 - 博客园 (cnblogs.com)](https://www.cnblogs.com/coderxiaobai/p/14780195.html)

设置本地与远程分支关联：

git branch --set-upstream-to=origin/branch branch（前面branch为远程分支名，后面branch为本地分支名）

#### （6）远程拉取

[Git创建本地分支并关联远程分支（一） - Kero小柯 - 博客园 (cnblogs.com)](https://www.cnblogs.com/kesimin/p/9936266.html)

## 5.远程库操作

代码托管中心

局域网环境，公有服务器上使用gitlab搭建一个代码托管中心；

外网环境，第三方平台

### 5.1 拉取

Git pull <远程库地址><远程分支>

取个别名

别名：git remote add <别名><远程库地址>



Clone拉取：Git clone URL

将远程仓库的所有内容克隆到本地（避免了拉取操作）

默认创建了origin作为远程仓库地址的别名（避免了取别名的操作）

初始化了本地库（避免了初始化操作）

### 5.2 推送

git push <远程库地址/别名> <要推送信息的分支(本地)>

### 5.3 协作

邀请添加

注意多人协作时的合并冲突

### 5.4 PR操作

Fork分支

本地修改

Push修改到远程分支

提交合并请求，pull request

审核者审核后合并代码

# 三、VSCode + Git

# 四、参考

Git开发分支对齐master主分支：<https://blog.csdn.net/xyl_20171026/article/details/112847997>

**Git系列：**[**Git实用教程（九）| 使用VS Code进行Git可视化操作-腾讯云开发者社区-腾讯云 (tencent.com)**](https://cloud.tencent.com/developer/article/1662720)