



Git简介和环境搭建使用

朱源波

20220701

timer-2022.07.01

提 纲

01

Git是什么

02

Git环境搭配 (GitHub)

03

使用技巧

timer-2022.07.1

— Git是什么

由Linux之父Linus Torvalds在2005年创建：

- 产生于Linux开发社区
- 设计用于对Linux内核进行版本控制



Git的目标：

- 快速
- 支持非线性开发（拥有成千上万的平行分支）
- 完全分布式
- 能够有效地处理大型工程项目

版本控制例子：

- 天若有情天亦老（V1）
- 天若有情天亦老，人间正道是沧桑（V2）



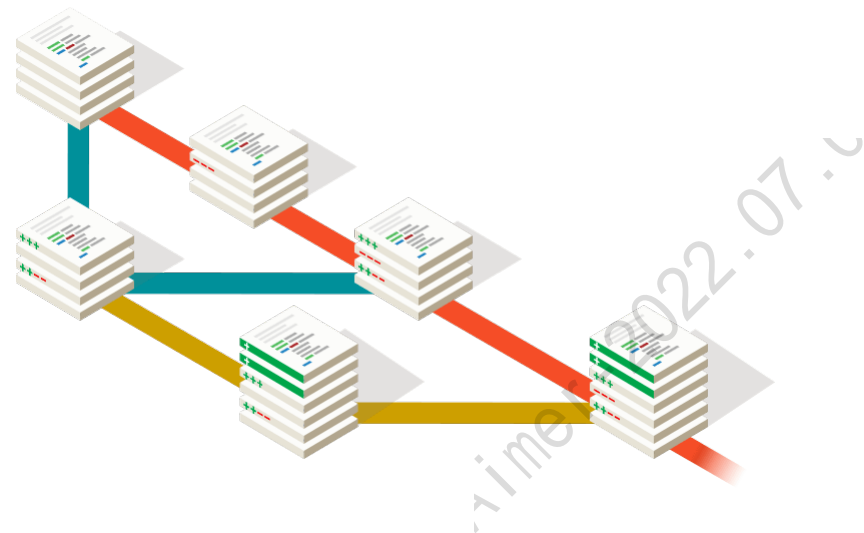
— Git是什么

Git是一个自由和开源的分布式版本控制系统，旨在以速度和效率处理从小型到非常大的项目。Free and Open Source

Git很容易学习，占地面积小，性能快如闪电。它超越了像Subversion、CVS、Perforce和ClearCase这样的SCM工具，具有廉价的本地分支、方便的暂存区域和多种工作流程等特点。



--distributed-is-the-new-centralized



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

— Git是什么

中心化 VCS(Version Control System)

1. 在Subversion、CVS、Perforce等。一个中央服务器仓库（repo）持有代码的 "官方副本"

服务器维护版本库的唯一历史版本

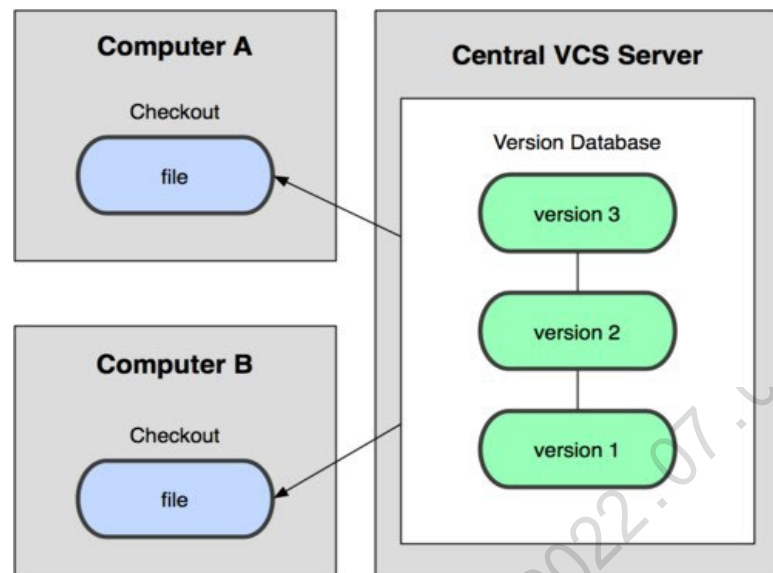
2. 当把官方副本导出到本地

能够在本地修改

本地变动没有版本号

3. 本地完成修改后，需要上传到中心服务器

本地版本才能算加入官方版本库的版本



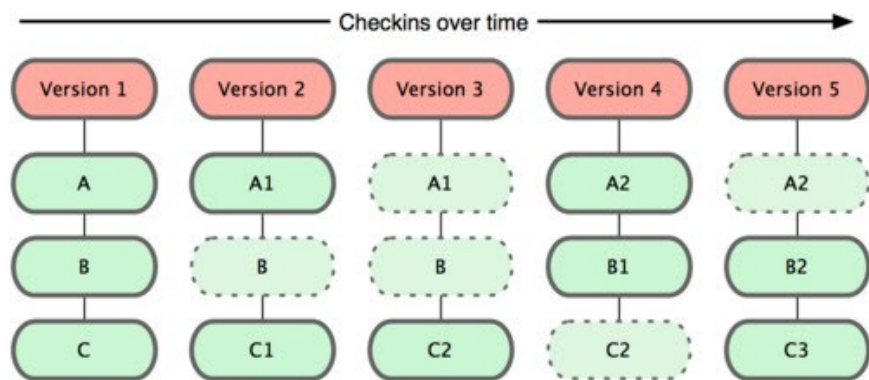
— Git是什么

Git快照(snapshot: 在电脑系统中, 快照是整个系统在某个时间点上的状态)

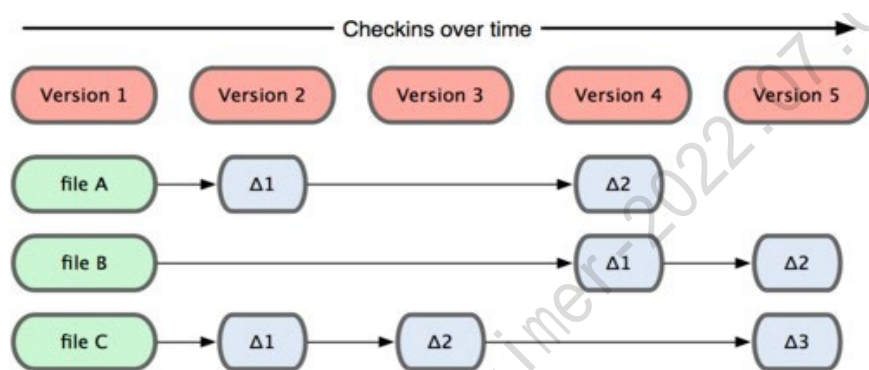
1. Git保留了整个项目状态的 "快照"

- 整体代码的每个checkin版本中都有一份每个文件的副本
- 有些文件在给定的checkin时发生变化, 有些则没有
- 更多的冗余, 但更快

2. 像Subversion这样的集中式VCS跟踪每个文件的版本数据



Git



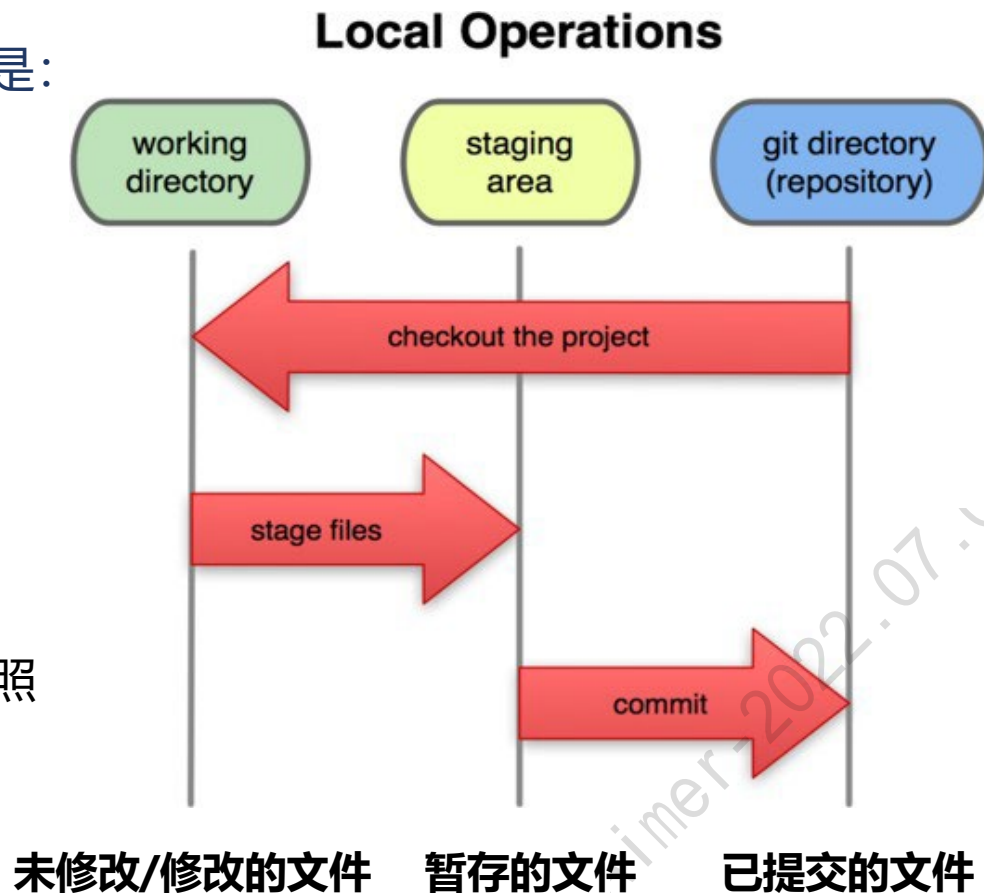
Subversion

— Git是什么

Git本地仓库的三大区域

在Git上的本地仓库副本中，文件可以是：

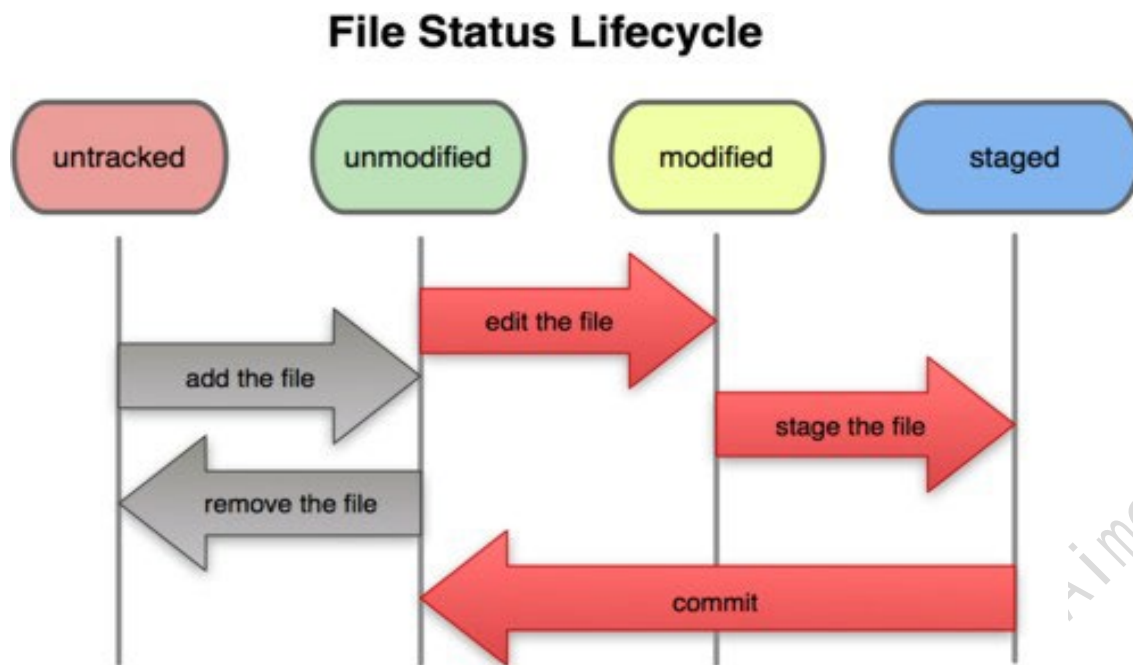
1. 在你的本地仓库
 - 被提交状态 (committed)
2. 已签出并修改，但尚未提交
 - 工作状态中的副本
3. 两者之间的缓存区
 - 阶段性的文件已经准备好被提交
 - 提交就会保存所有暂存状态的快照



— Git是什么

Git的基础工作流程

1. 修改工作目录中的文件
2. 阶段性文件 (Stage files) , 将它们的快照添加到Git的暂存区域
3. 提交 (Commit) 将暂存区域的文件, 并将该快照永久地保存在Git的目录中

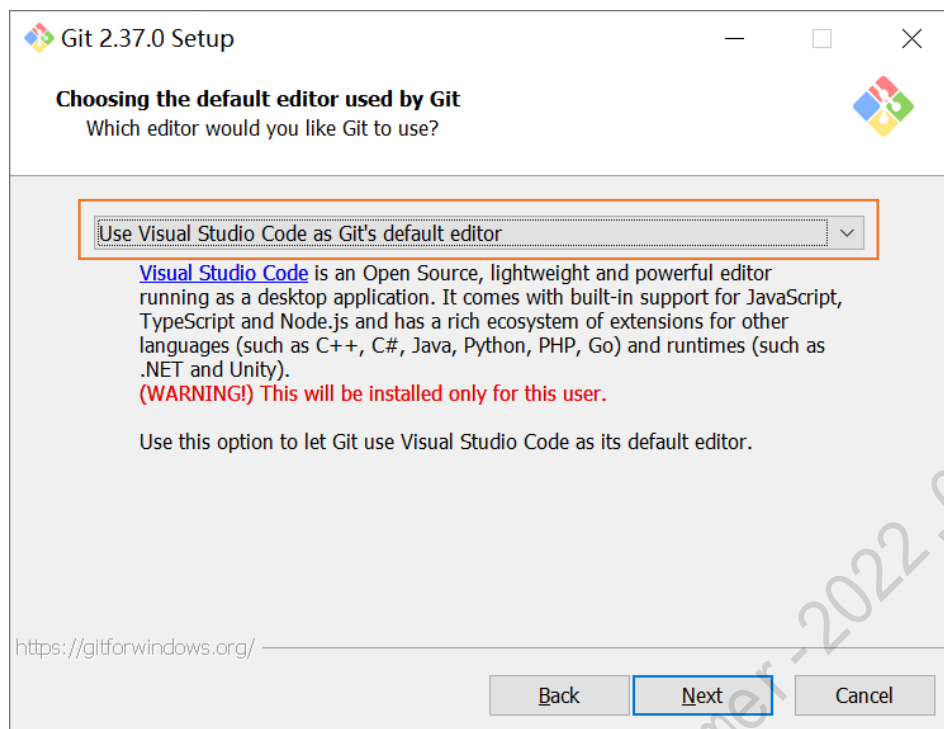
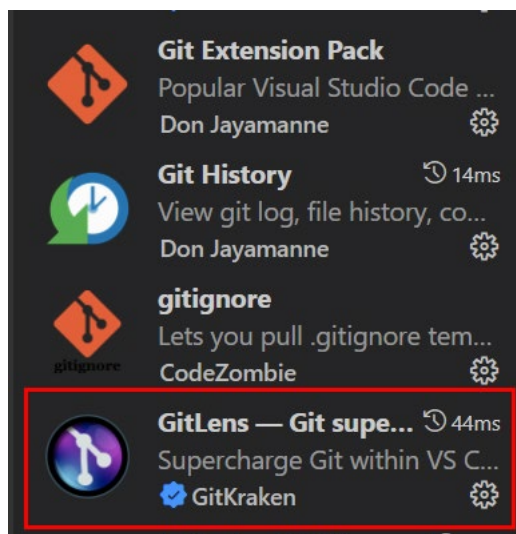


二 Git环境搭建

软件安装环境准备

这里示例用的是VS Code+Git的方案，也可以直接使用其他编辑工具作为Git的代码编辑器（Nano、Vim、Sublime Text等）。

注：需要先安装代码编译器，再安装Git的安装包，然后在Git安装界面可以选择编译器。



Visual Studio Code下载地址: <https://code.visualstudio.com/download>

Git下载地址: <https://git-scm.com/download/win>

二 Git环境搭建

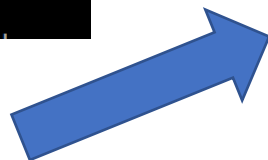
软件安装环境准备

1.创建SSH Key和绑定到Github账户:

`ssh-keygen -t rsa -b 4096 -C "your_email@example.com"`

```
MINGW64 /d/Github
ssh-keygen -t rsa -b 4096 -C "your_email@example.com"
Generating public/private rsa key pair.
Enter file in which to save the key (/c:/Users/Aimer/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
```

id_rsa 私钥
id_rsa.pub 公钥



Public profile
Account
Appearance
Accessibility
Notifications

Access
Billing and plans
Emails
Password and authentication
SSH and GPG keys
Organizations
Moderation

SSH keys / Add new

Title

XXXX

Key

Begins with 'ssh-rsa', 'ecdsa-sha2-nistp256', 'ecdsa-sha2-nistp384', 'ecdsa-sha2-nistp521', 'ssh-ed25519', 'sk-ecdsa-sha2-nistp256@openssh.com', or 'sk-ssh-ed25519@openssh.com'

Add SSH key

2.git配置用户名和邮箱:

`git config --global user.name "username"`

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

3.git添加公钥与测试:

测试命令 `ssh -T git@github.com`

ssh -T报错后: `ssh-agent bash`

`ssh-add -k ~/.ssh/XXX_rsa`

```
$ ssh -T git@github.com
git@github.com: Permission denied (publickey).
```

```
Aimer@DESKTOP-9LSFK0H MINGW64 /d/Github
$ ssh -T git@github.com
Hi xthc! You've successfully authenticated,
cess.
```

二 Git环境搭建

GitHub常使用的三大概念:

Repository : 简称 “Repo” ,仓库, 库。库是GitHub的最基本元素, 可想象成本地的项目文件夹; 一个库包含所有的项目文件(包括帮助文档), 并保存每个文件的修改历史; 库可以有多个合作开发者, 也可以作为公共库或私有库的形式开发。

Commit : 提交信息。或者称为修改信息, 是个人提交的对文件的修改记录。

Branch: 分支。分支是一个库的并行版本, 包含在库内, 允许独立的开发而不影响现有主分支(primary or master)的运行; 当在分支的修改需要发布时, 就可以将分支合并(merge)至主分支(master branch), 这样利于多人的分布式开发。

二 Git环境搭建

初始化仓库:

The screenshot illustrates the process of initializing a local Git repository from a GitHub repository. It shows the 'Clone' dialog box with the repository URL `https://github.com/huggingface/transformers` highlighted (1) and the 'Download ZIP' option selected (2). Below the dialog, the local file explorer shows the repository structure, with the `.git` folder highlighted (3).

Clone Dialog:

- Repository: `https://github.com/huggingface/transformers` (1)
- Options: HTTPS, SSH, GitHub CLI
- Buttons: Open with GitHub Desktop, Download ZIP (2)

Local File Explorer:

名称	类型
.circleci	文件夹
.github	文件夹
.git	文件夹 (3)
docker	文件夹
docs	文件夹
examples	文件夹
model_cards	文件夹
notebooks	文件夹
scripts	文件夹

二 Git环境搭建

初始化仓库:

初始化命令: `git init`, 使其成为Git的仓库 (repo)

MINGW64:/d/Github/transformers-main

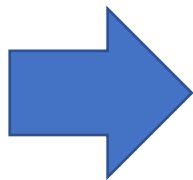
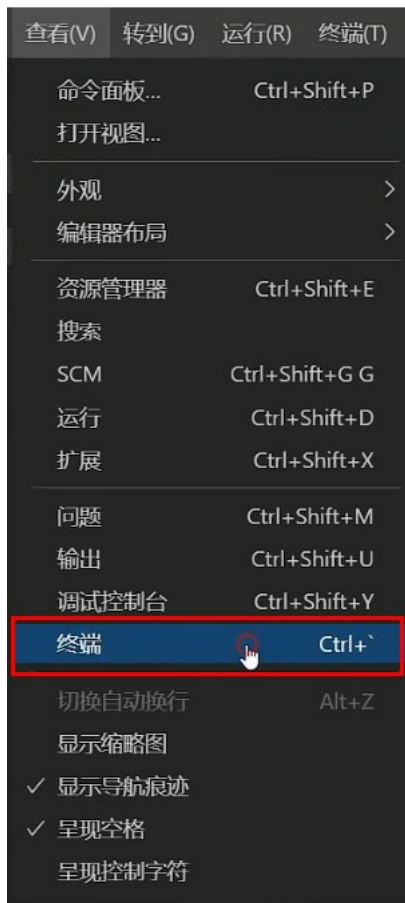
```
Aimer@DESKTOP-9LSFK0H MINGW64 /d/Github/transformers-main
$ git init
Initialized empty Git repository in D:/Github/transformers-main/.git/
Aimer@DESKTOP-9LSFK0H MINGW64 /d/Github/transformers-main (master)
$
```



Github > transformers-main >		
名称	^	类型
.circleci		文件夹
.git		文件夹
.github		文件夹
docker		文件夹
docs		文件夹
examples		文件夹
model_cards		文件夹
notebooks		文件夹
scripts		文件夹

二 Git环境搭建

第一次仓库提交:

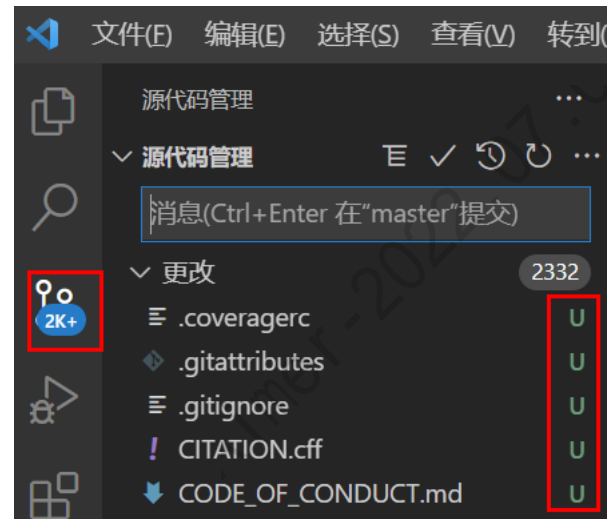
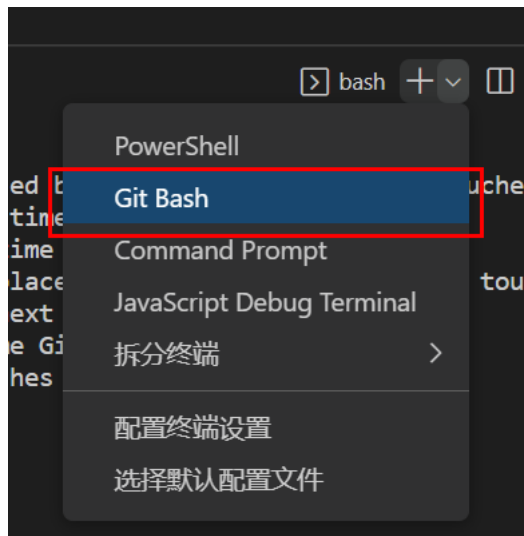


```
问题 输出 调试控制台 终端 GITLENS JUPYTER

Windows PowerShell
版权所有 (C) Microsoft Corporation。保留所有权利。

尝试新的跨平台 PowerShell https://aka.ms/pscore6

PS D:\Github\transformers-main> git -v
git version 2.37.0.windows.1
PS D:\Github\transformers-main> |
```

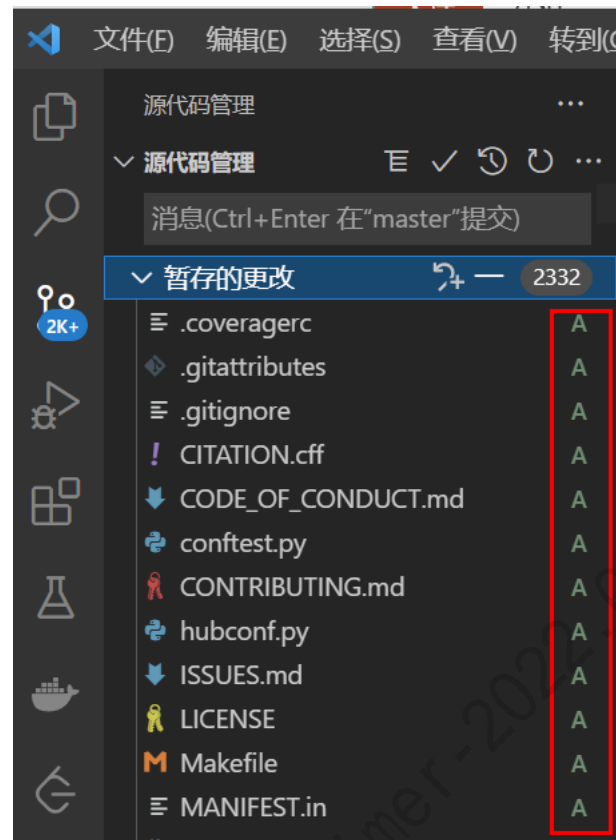
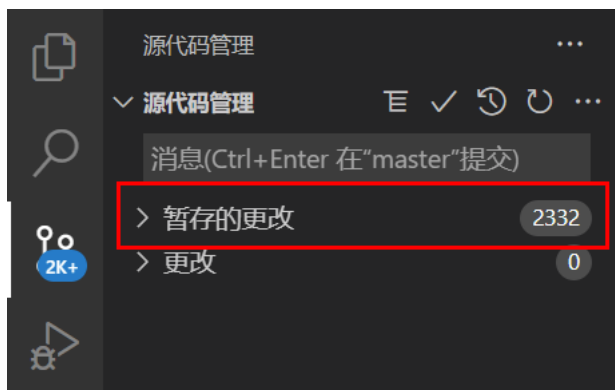


二 Git环境搭建

第一次仓库提交:

提交到暂存区命令: `git add -A`

```
Aimer@DESKTOP-9LSFK0H MINGW64 /d/Github/transformers-main (master)
$ git add -A
warning: in the working copy of '.circleci/config.yml', LF will be replaced by CRLF
warning: in the working copy of '.coveragerc', LF will be replaced by CRLF
warning: in the working copy of '.gitattributes', LF will be replaced by CRLF
```



二 Git环境搭建

第一次仓库提交:

提交到仓库命令: `git commit -m "提交信息"`

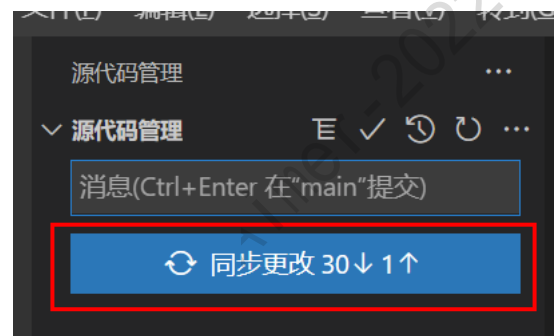
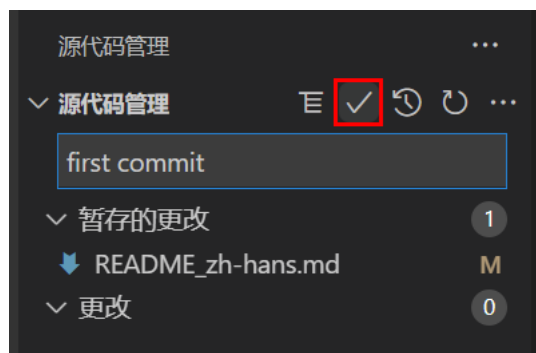
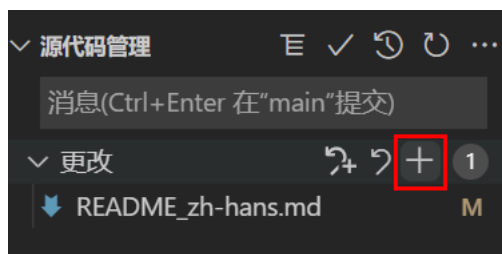
```
问题 输出 调试控制台 终端 GITLENS JUPYTER

create mode 100644 utils/test_module/custom_configuration.py
create mode 100644 utils/test_module/custom_feature_extraction.py
create mode 100644 utils/test_module/custom_modeling.py
create mode 100644 utils/test_module/custom_processing.py
create mode 100644 utils/test_module/custom_tokenization.py
create mode 100644 utils/test_module/custom_tokenization_fast.py
create mode 100644 utils/tests_fetcher.py
create mode 100644 utils/tf_ops/onnx.json
create mode 100644 utils/update_metadata.py
create mode 100644 valohai.yaml

Aimer@DESKTOP-9LSFK0H MINGW64 /d/Github/transformers-main (master)
$
```

Github > transformers-main > .git

名称	类型
hooks	文件夹
info	文件夹
logs	文件夹
objects	文件夹
refs	文件夹
COMMIT_EDITMSG	.file
config	.file
description	.file
HEAD	.file
index	.file



二 Git环境搭建

仓库与远程 (GitHub) 的交互:

Pull 远程分支代码到本地分支:

- 1、在推送变更之前先确认远程库的名称: `git remote`

```
thanos18@lifecompanion:~/articles-of-the-week$ git remote  
origin  
thanos18@lifecompanion:~/articles-of-the-week$
```

- 2、把变更推送到GitHub: `git push [Remote Name] [Branch Name]`

```
remote:  
To https://github.com/ThanoshanMV/articles-of-the-week.git  
* [new branch]      my-article -> my-article  
thanos18@lifecompanion:~/articles-of-the-week$
```

Push 提交本地分支代码到远程分支:

- 1、将代码提交到暂存区: `git add .`
- 2、添加提交信息: `git commit -m "提交的信息"`
- 3、提交本地分支代码到远程分支(注意: 该命令应该在本地分支下执行):
`git push [Remote Name] [Branch Name]`

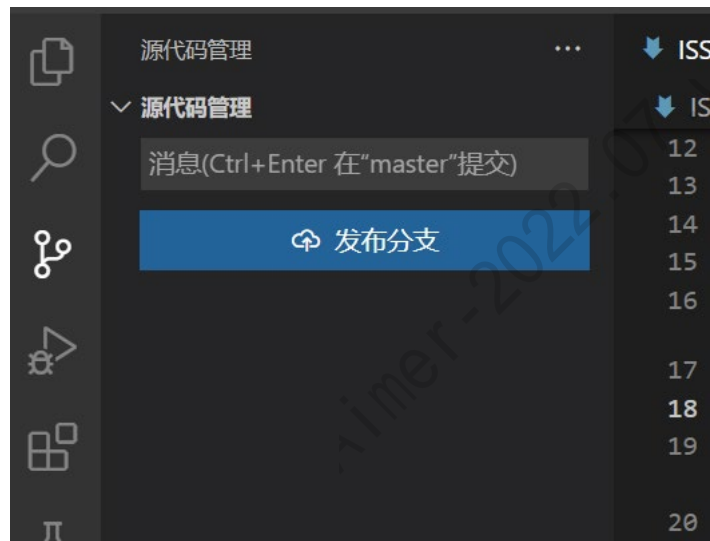
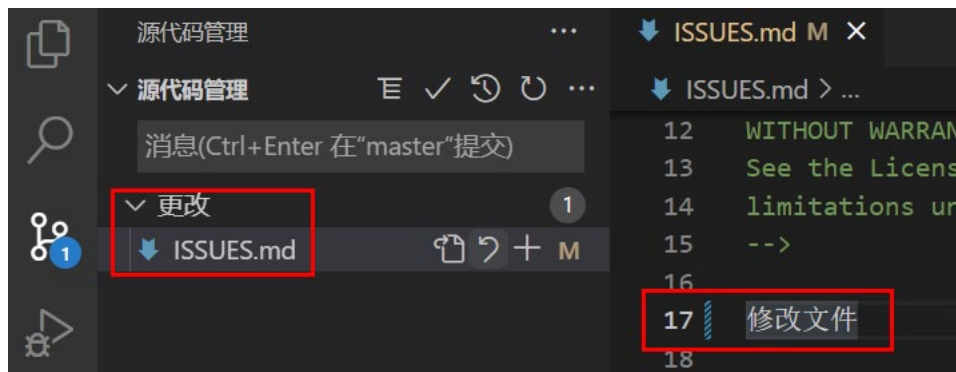
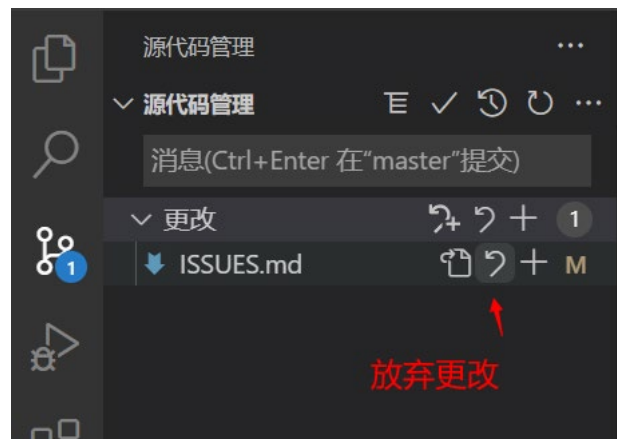
二 Git环境搭建

仓库日常维护:

工作区撤回: `git checkout <filename>`

```
Aimer@DESKTOP-9LSFK0H MINGW64 /d/Github/transformers-main (master)
$ git checkout ISSUES.md
Updated 1 path from the index

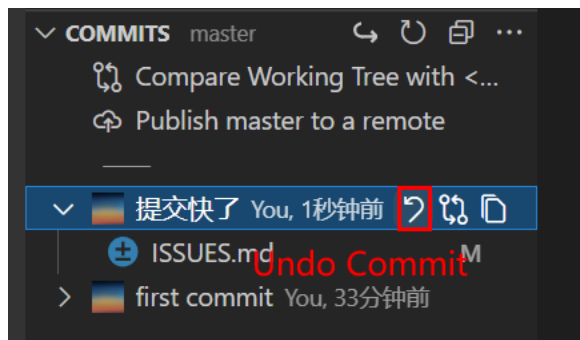
Aimer@DESKTOP-9LSFK0H MINGW64 /d/Github/transformers-main (master)
$
```



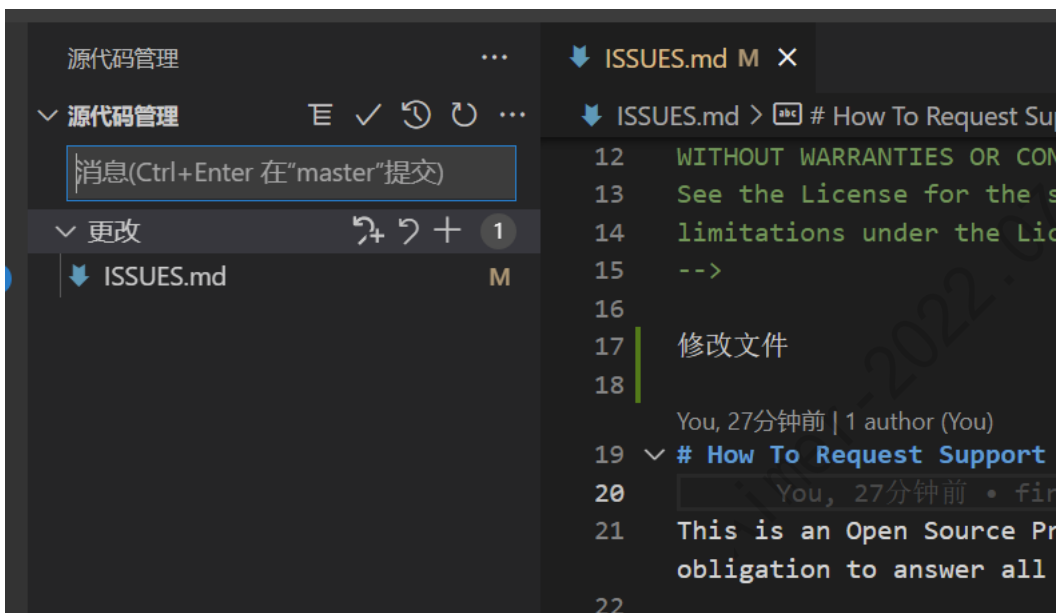
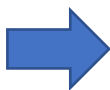
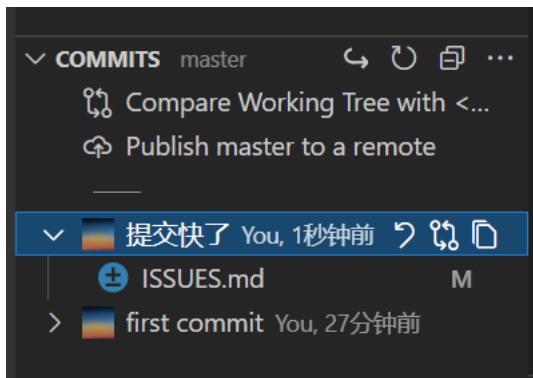
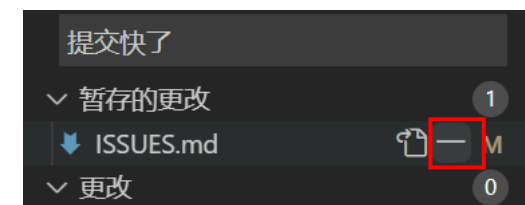
二 Git环境搭建

仓库日常维护:

提交后撤回: `git reset HEAD^`



```
Aimer@DESKTOP-9LSFK0H MINGW64 /d/Github/transformers-main (master)
$ git reset HEAD^1
Unstaged changes after reset:
M   ISSUES.md
```



二 Git环境搭建

仓库的分支命令：

以当前分支为基础新建分支

git checkout -b <branchname>

列举所有的分支

git branch

单纯地切换到某个分支

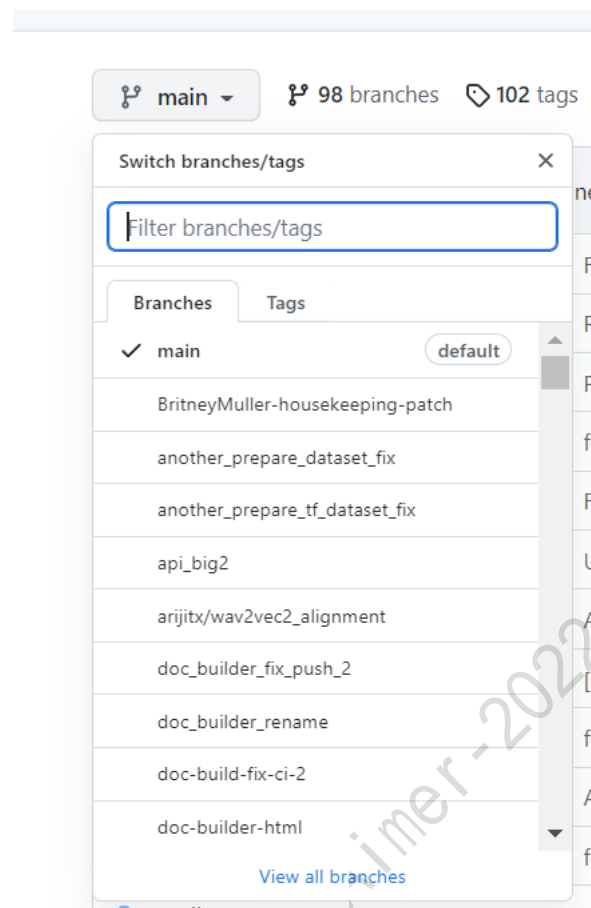
git checkout <branchname>

删掉特定的分支

git branch -D <branchname>

合并分支

git merge <branchname>



三 使用技巧

Github的镜像加速器

关于 FastGit 的使用，本质上与 `git` 有关。常规的面向 GitHub 的 `clone` 命令可能如下：

```
git clone https://github.com/author/repo
```

sh

使用 FastGit 时，可使用如下命令：

```
git clone https://hub.fastgit.xyz/author/repo
```

sh

正如您所见，FastGit 仅仅是 GitHub 的代理，所以我们仅需要替换远程地址。

当然，您也可以直接修改 `git` 的配置，使用 FastGit 替换所有指向 GitHub 的链接：

```
git config --global url."https://hub.fastgit.xyz/".insteadOf "https://github.com/"
git config protocol.https.allow always
```

sh

示例：`git clone https://github.com/FastGitORG/document.git`
`git clone https://hub.fastgit.xyz/FastGitORG/document.git`

三 使用技巧

Release 和源码存档的下载

对于正常的 `clone` , `push` 操作, FastGit 已经提供了相当完善的操作。对于 Release 和源码存档的下载, 我们可以使用如下方法进行操作。

```
# Release
# 假设下载链接为 https://github.com/A/A/releases/download/1.0/1.0.tar.gz
wget https://download.fastgit.org/A/A/releases/download/1.0/1.0.tar.gz

# Codeload
# 假设下载链接为 https://hub.fastgit.xyz/A/A/archive/master.zip
# 或者 https://codeload.github.com/A/A/zip/master
wget https://download.fastgit.org/A/A/archive/master.zip
```

示例:

<https://download.fastgit.org/huggingface/transformers/archive/refs/tags/v4.20.1.tar.gz>

<https://download.fastgit.org/CorentinJ/Real-Time-Voice-Cloning/archive/refs/heads/master.zip>

参考资料

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

Git Documentation: <https://git-scm.com/doc>

Github快速入门: <https://docs.github.com/cn/get-started>

GitHub 词汇表:

<https://docs.github.com/cn/get-started/quickstart/github-glossary>

Git 和 GitHub 教程——版本控制入门:

<https://chinese.freecodecamp.org/news/git-and-github-for-beginners/>

GitHub的的镜像加速器项目地址:

<https://github.com/FastGitORG/document>

Merci! 谢谢!

timer-2022.07.2