1. 什么是git？

Git是一个分布式版本控制系统，无需联网，不但能自动记录每次文件的改动，还可以协作编辑。

1. git诞生

Linux是由世界各地人群撰写代码并更新 早期2002年需要Linus（Linux创建人）需要本人手动合并代码，引起社区许多人不满，于是linus选择了商用的Bitkeeper，并且Bitkeeper将Linux社区免费试用，

在2005，有社区大佬想要试图破解Bitkpper协议，被Bitmove公司发现了，公司红温了就收回了Linux社区免费使用权。随后linus花了两周时间使用c语言写了一个分布式版本控制系统，这就出现了Git。

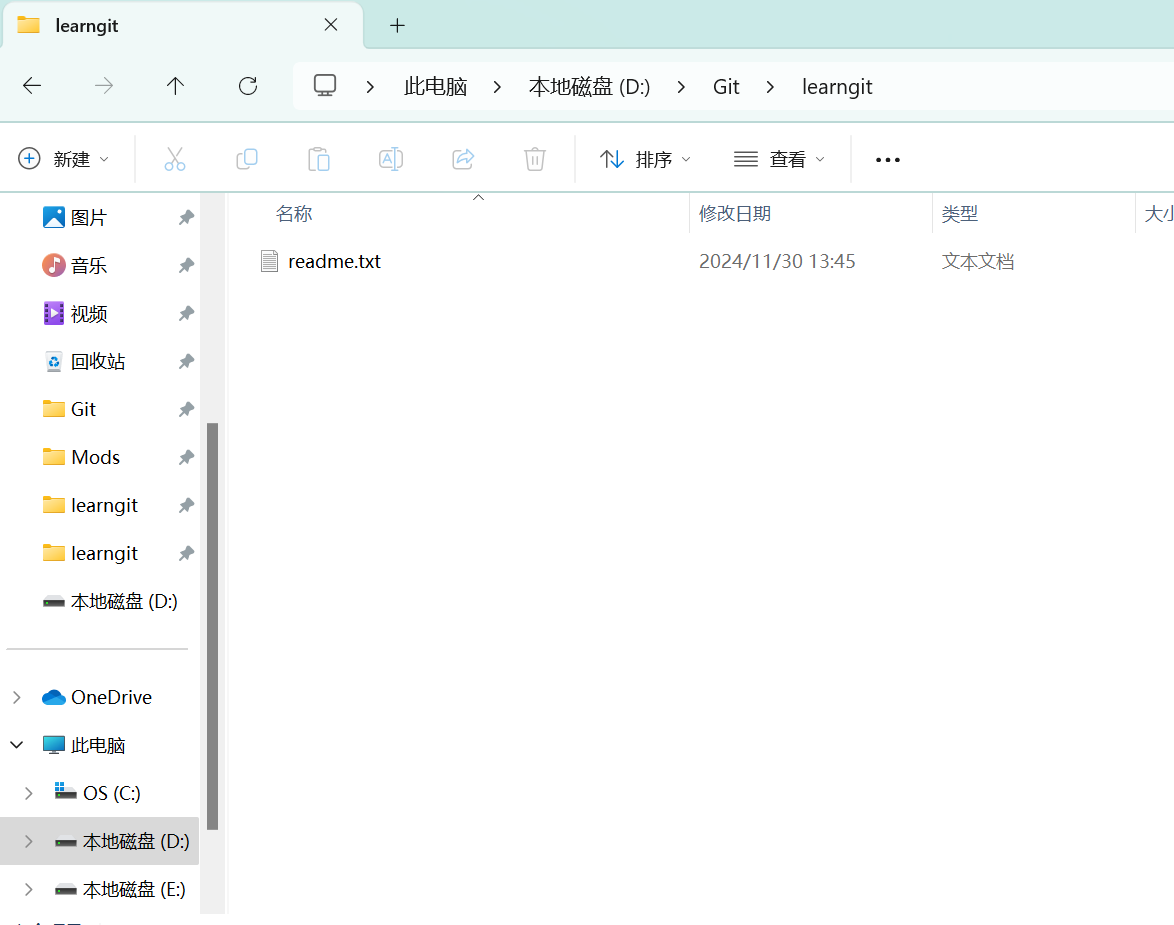
1. 集中式控制系统和分布式控制系统

**分布式版本控制系统，每一台计算机都有完整的版本库（比集中式好使，因为集中式服务器一挂就无法继续工作。而分布式自身就有完整版本库即使挂掉一台用其他的进行工作）。**

**集中式版本控制系统，版本库存放在中央服务器。**

工作区（Working Directory）

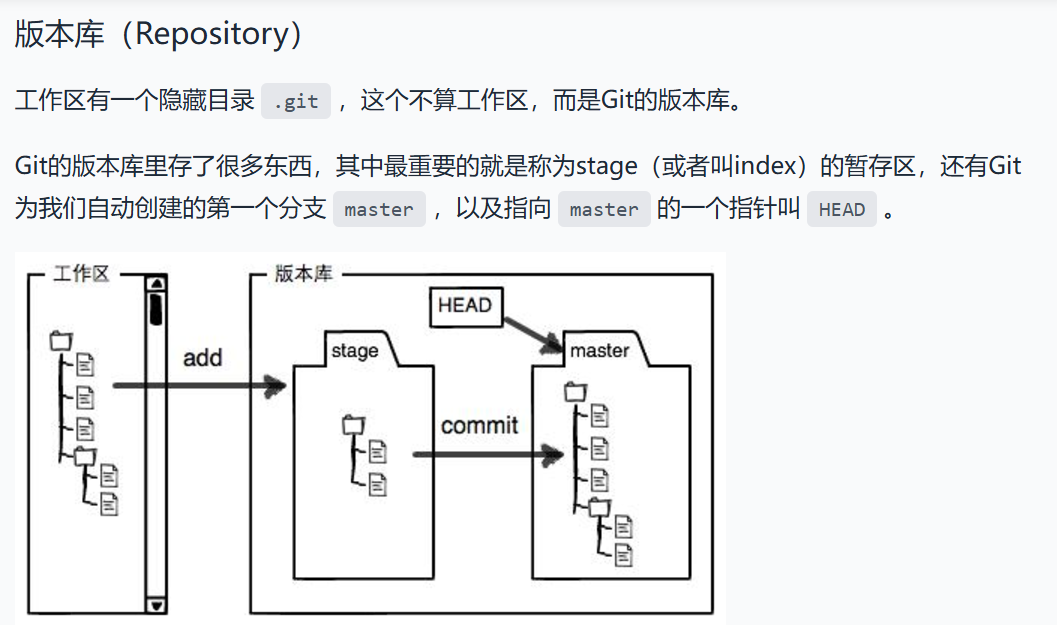
就是你在电脑里能看到的目录，比如我的learngit文件夹就是一个工作区：



版本库（Repository）

工作区有一个隐藏目录.git，这个不算工作区，而是Git的版本库。

Git的版本库里存了很多东西，其中最重要的就是称为stage（或者叫index）的暂存区，还有Git为我们自动创建的第一个分支master，以及指向master的一个指针叫HEAD。



总体意思就是git add 先加入到 stage（暂寸区域） 在之后使用git commit 全部添加至master文件中

已知命令大全

Mkdir 文件名 创造一个新的空文件目录

Cd 进入 Git 仓库所在的目录

例子： cd learngit 进入在learngit目录下的git仓库

Pwd 显示当前的目录

Git add将这些文件的修改加入暂存区，准备提交。

Git commit 提交更新（暂存区的内容提交到本地 Git 仓库。）

在使用git commit 最好加上-m说明信息

语法：git commit -m “XXXXXXXXXX”

Git init 将目录转化为一个git仓库

Git status 用于查看当前工作目录和暂存区的状态。它可以帮助你了解哪些文件已被修改、哪些文件被添加到暂存区，哪些文件还没有提交。

Git diff 会展示文件内容的修改情况，帮助你查看有哪些更改尚未提交。

Cat 用于 **显示文件内容**、**连接文件** 或 **创建文件**。它的全称是 **concatenate**，意味着“连接”。

例句：

1. cat readme.txt 查看readme.txt文件内容
2. cat file1.txt file2.txt 查看file1.txt file2.txt的内容按顺序连接在一起
3. cat > newfile.txt 创建一个新文件并输入内容输入完毕后使用ctrl + D保存并退出

cat的常见选项

* -n：显示行号。
* -b：显示行号，但不为空行显示行号。
* -E：在每行的末尾显示 $，帮助识别行尾空格。
* -T：显示制表符（tab）为 ^I。
* -s：压缩多个连续的空行为一行。

Git diff HEAD作用：

工作区与 HEAD 之间的差异：显示你当前工作目录和最近一次提交之间的更（包括未暂存的更改）。

暂存区与 HEAD 之间的差异：显示已暂存的文件（通过 git add）和最新提交之间的差异。

git checkout --文件名 将这个文件在工作区的修改全部撤销。

分两种情况：

（1）一种是readme.txt自修改后还没有被放到暂存区，现在，撤销修改就回到和版本库一模一样的状态；

（2）一种是readme.txt已经添加到暂存区后，又作了修改，现在，撤销修改就回到添加到暂存区后的状态。

Git rm 文件名 删除文件名的文件 放置暂存区。

Git commit 文件名 提交删除 （加入删错了check out 也可以一键救活）

创建git ssh key

cat ~/.ssh/id\_rsa.pub 抓取ssh key （也可以当检查有没有ssh key使用）

也可以使用 ls -al ~/.ssh (如果看到了id\_rsa或者id\_eq25519文件，也说明你有ssh key了

ssh-keygen -t rsa -b 4096 -C "github的邮箱账户"(建立ssh key)【-t rsa 指定类型为rsa ，

-b 4096 指定密钥长度为4096

ssh -T [git@github.com](mailto:git@github.com) (检测ssh链接是否建立成功)

[成功会显示 Hi “你的用户名” You've successfully authenticated, but GitHub does not provide shell access]

创建git远程仓库

1. 先进入到你的github主页找到右上角按的+会显示create new repository 点击

New repository

1. 在Repository name填入你的git 仓库的文件名字
2. 点击create repository

将本地仓库上传到git远程仓库上（注：需要先完成ssh key的注册及绑定）

1. 打开git bash输入以下

git remote add origin [git@github.com](mailto:git@github.com):你的用户名 /你的本地仓库名.git

1. 将本地仓库内容推送到git.hub的远程仓库上

git push -u origin master

注：如果第一次上传则会弹出ssh警告

大概弹出：The authenticity of host 'github.com (xx.xx.xx.xx)' can't be established.

RSA key fingerprint is xx.xx.xx.xx.xx.

Are you sure you want to continue connecting (yes/no)?

输入yes信任即可

删除远程仓库和本地仓库的链接（如需删除远程需要github上手动删除）

1. 先使用git remote -v指令查询一下远程库的信息
2. 随后根据名字进行删除 git remote rm 库名字

从远程库克隆到本地库

1. 在创建库时需要多一个操作 下拉找到Initialize this repository with a README

创建成功后会加入一个readme.md的文件

1. 使用git clone [git@github.com:你的用户名/你的github远程库名.git](mailto:git@github.com:你的用户名/你的github远程库名.git)

Git分支：就是在head和master之间拆入一个分支可以将文件暂时提交在这里，对于未完成文件有着极大帮助（如果提交下家工作无法执行，不提交会文件丢失）

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>或者git switch <name>（最好使用，checkout比较强大）

创建+切换分支：git checkout -b <name>或者git switch -c <name>（最好使用checkout比较强大）

合并某分支到当前分支：git merge <name>

删除分支：git branch -d <name>

查看历史分支：git log --graph --pretty=oneline --abbrev-commit（其实git log就行这条命令可以查看提交历史并以简洁、图形化的方式显示分支和合并的情况）

–-graph 这会在提交历史中显示一个 ASCII 图形，帮助你直观地看到分支和合并的结构。它会以图形化的方式显示分支的历史。

**--pretty=oneline**：这将每个提交显示为一行。它可以让你快速浏览所有提交，而不会显示详细的提交信息

**--abbrev-commit**：这会将每个提交的哈希值简化为较短的形式，通常是 7 个字符，而不是显示完整的 40 位 SHA-1 哈希。这样可以使输出更加简洁。

假如忘记了两条分支都提交了新的记录，此时在合并就会报错显示冲突，需要进行下一部操作必须先解决冲突，使用git status查看后进入要合并文件夹里面有详细的信息，将箭头所指向二者删除一个即可。

分支策略

**快速合并**：没有历史分支，Git 直接将当前分支的指针移动到合并分支的最新提交。

**非快速合并**：有历史分支，Git 创建一个新的合并提交以连接两个分支的历史。

**可看出使用默认的快速合并在查询时分支时看不到历史分支而使用非快速合并就可以查看**

git merge --no-ff -m “备注” 分支名 使用非快速合并

加入你被要求去修复一个bug的工作需要创建一个分支去修复，但是你当前的分支工作还没有完成提交这时就可以使用

Git stash 将工作暂存

Git stash list查看暂存列表

完成之后如何回复呢

一是用git stash apply恢复，但是恢复后，stash内容并不删除。

另一种方式是用git stash pop，恢复的同时把stash内容也删了

**注：在主分支上修复了bug后，我们要想一想，子分支是早期从主分支分出来的，所以，这个bug其实在当前子分支上也存在。**

git cherry-pick 对应提交的哈希值 这条命令能复制一个特定的提交到当前分支：

开发一个新功能，最好新建一个分支；feature-<name>

如果要丢弃一个没有被合并过的分支，可以通过git branch -D <name>强行删除。

多人协作

1. 可以尝试用git push origin <分支名>推送自己的修改；
2. 如果推送失败，则因为远程分支比你的本地更新，需要先用git pull试图合并；
3. 如果合并有冲突，则解决冲突，并在本地提交；
4. 没有冲突或者解决掉冲突后，再用git push origin <分支名>推送就能成功！

git rebase把分叉的提交历史“整理”成一条直线，看上去更直观但是本地分叉提交会被修改

打标签

* git tag <tagname>用于新建一个标签，默认为HEAD，也可以指定一个commit id；
* 命令git tag -a <tagname> -m "XXXXXXX"可以指定标签信息；
* 命令git tag可以查看所有标签。

推标签

* 命令git push origin <tagname>可以推送一个本地标签；
* 命令git push origin --tags可以推送全部未推送过的本地标签；
* 命令git tag -d <tagname>可以删除一个本地标签；
* 命令git push origin :refs/tags/<tagname>可以删除一个远程标签。

**保命大法！（回退上一个版本【加入退多了还能回】）**

**先用git log查看提交历史 （使用q键退出）**

**Git reflog查看历史命令**

**git reset --hard commit\_id（回退或者穿梭）【commit id就是那一坨黄色的】**