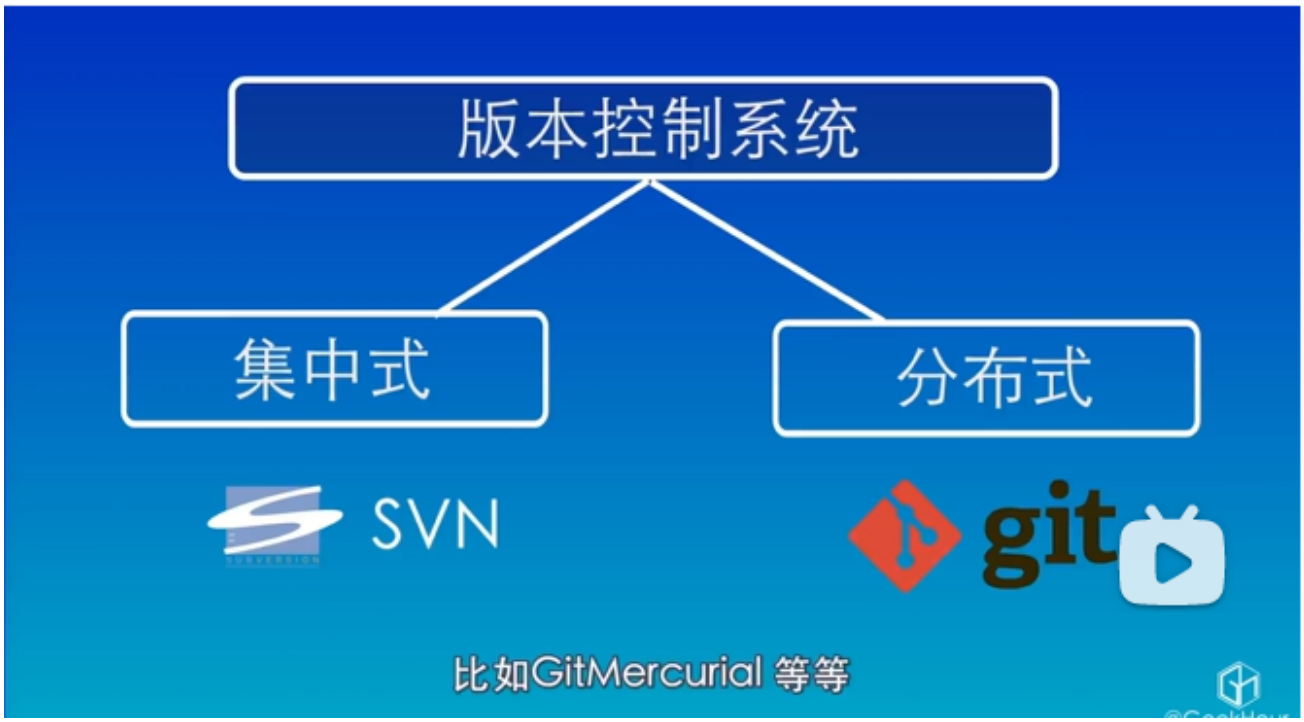
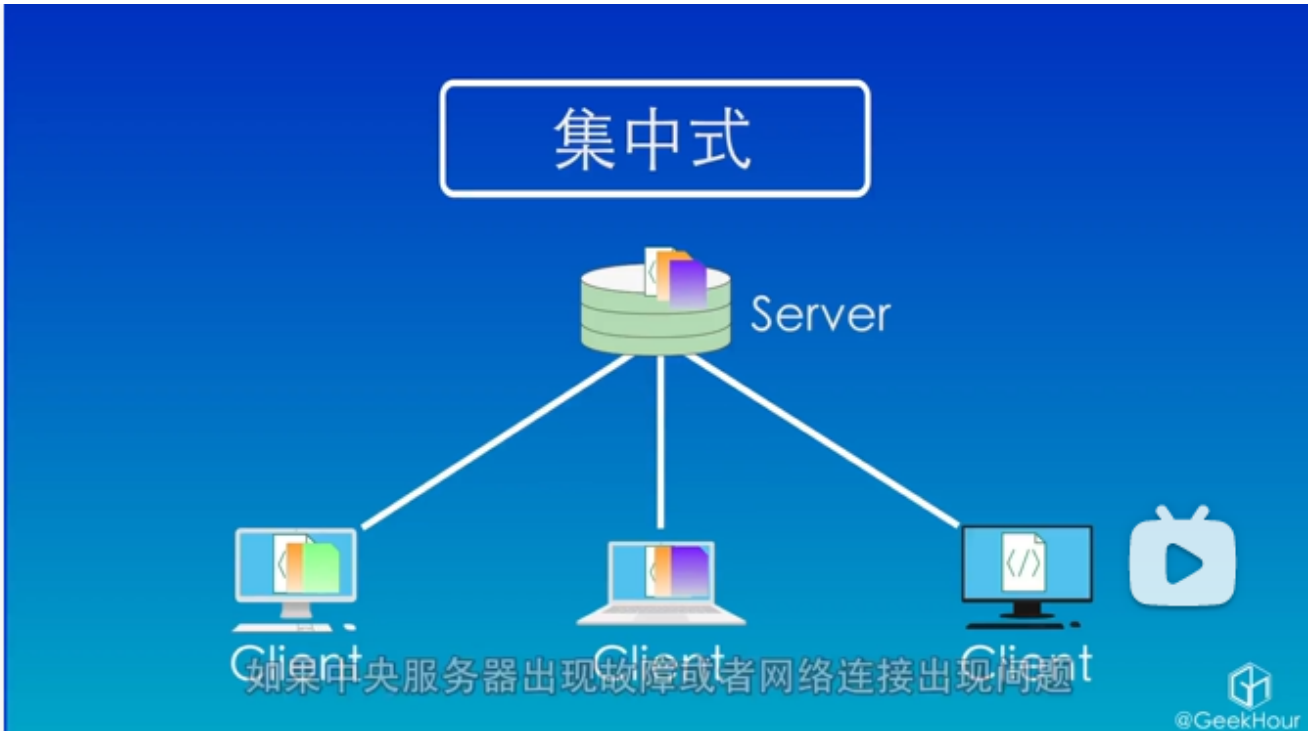
一、简介

Git是一个免费开源的分布式版本控制系统,版本控制系统可以跟踪每个文件的变化,让项目成员之间的协作更加高效

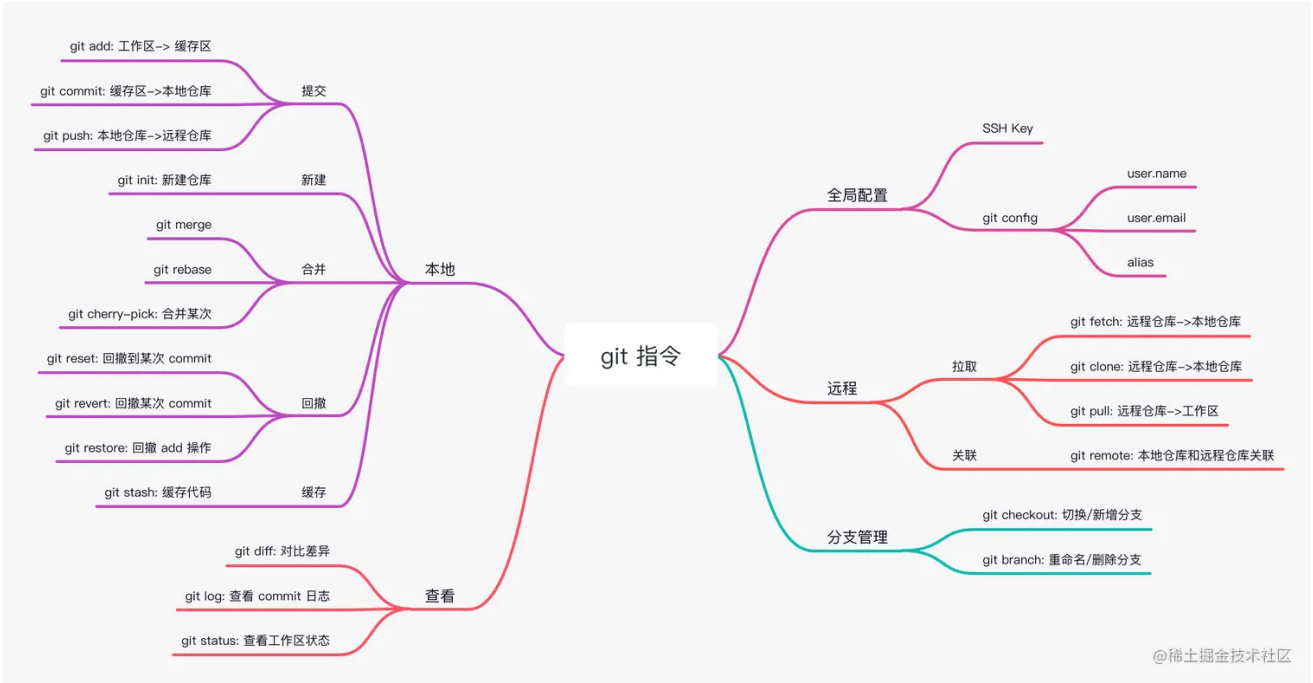


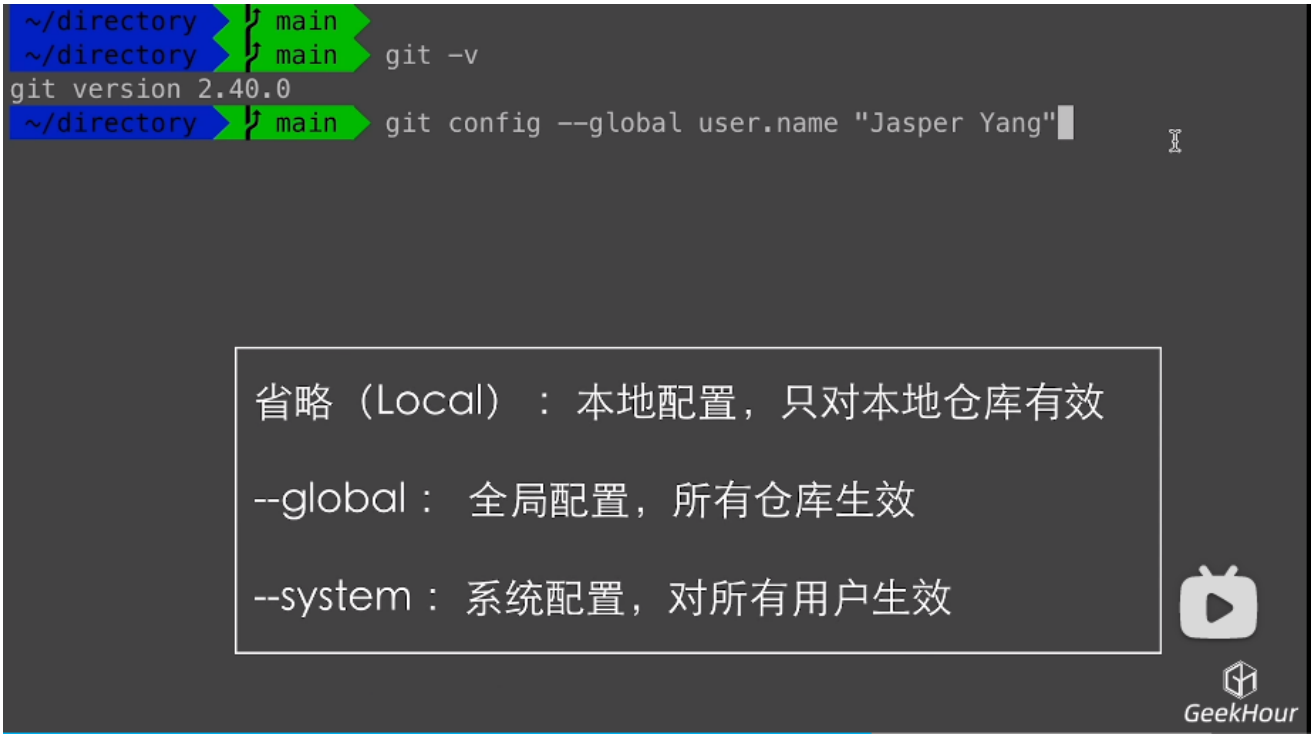


二、Git基础

1.Git命令

Git 指令看这一篇就够 —— 各种工作场景的 git 指令大全 - jamiechoo - 博客园 (cnblogs.com)





配置用户名和邮箱

保存用户名和密码

查看配置信息

2.创建仓库

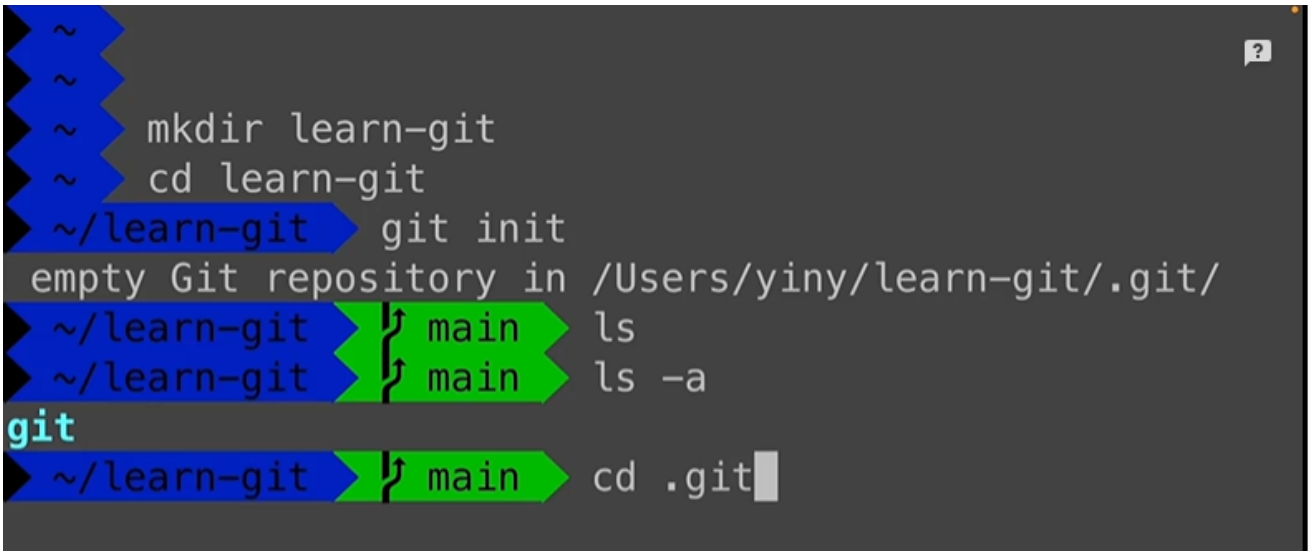
(仓库就相当于目录,这个目录里所有文件都可以被Git管理起来,每个文件的修改,删除,添加等功能,Git都能跟踪到,以便任何时候都可以追踪历史或者还原到以前的某一个版本)

创建仓库就是把一个目录变成Git可以管理的仓库

方式一是在自己电脑本地之间创建一个仓库

方式二是从远程服务器克隆一个已经存在的仓库

创建一个目录并在这个目录下创建一个仓库,ls展示这个目录下有的文件,加-a展示所有文件包括隐藏的



ls -altr以长格式显示当前目录下的所有文件（包括隐藏文件），并按修改时间从旧到新排序

```
yiny@Ares ~/learn-git/.git ls -altr
total 24
drwxr-xr-x   3 yiny  staff   96  3 20 23:36 ..
drwxr-xr-x   3 yiny  staff   96  3 20 23:36 info
-rw-r--r--   1 yiny  staff   73  3 20 23:36 description
drwxr-xr-x  15 yiny  staff  480  3 20 23:36 hooks
drwxr-xr-x   4 yiny  staff  128  3 20 23:36 refs
-rw-r--r--   1 yiny  staff   21  3 20 23:36 HEAD
-rw-r--r--   1 yiny  staff  137  3 20 23:36 config
drwxr-xr-x   9 yiny  staff  288  3 20 23:36 .
drwxr-xr-x   4 yiny  staff  128  3 20 23:36 objects
yiny@Ares ~/learn-git/.git cd ..
yiny@Ares ~/learn-git main \rm -rf .git
yiny@Ares ~/learn-git git init my-repo
Initialized empty Git repository in /Users/yiny/learn-git/my-repo/
.git/
yiny@Ares ~/learn-git
```

\rm -rf .git这个命令会彻底、强制且无提示地删除当前目录的 Git 版本库核心数据，执行后：

本地所有的 Git 历史记录（提交、分支、标签等）会被永久删除。

无法再通过 git 命令回溯版本或恢复历史。

仅保留工作区的文件（即当前编辑的文件），但失去了版本控制能力。

```
yiny@Ares ~/learn-git/my-repo main cd ..
yiny@Ares ~/learn-git git clone https://github.com/geekhall-
aoyang/remote-repo.git
Cloning into 'remote-repo'...
warning: You appear to have cloned an empty repository.
yiny@Ares ~/learn-git ls -ltr
total 0
drwxr-xr-x  3 yiny  staff  96  3 20 23:38 my-repo
drwxr-xr-x  3 yiny  staff  96  3 20 23:41 remote-repo
yiny@Ares ~/learn-git cd remote-repo
yiny@Ares ~/learn-git/remote-repo main ls -a
.    ..    .git
yiny@Ares ~/learn-git/remote-repo main
```

3.Git的工作区域和文件状态

(1) Git的本地数据管理分为三个区域

工作区(也叫工作目录或者本地工作目录)

就是电脑上的目录,能在资源管理器里面能够看到的文件夹就是工作区

简单来说,工作区就是我们实际操作的目录

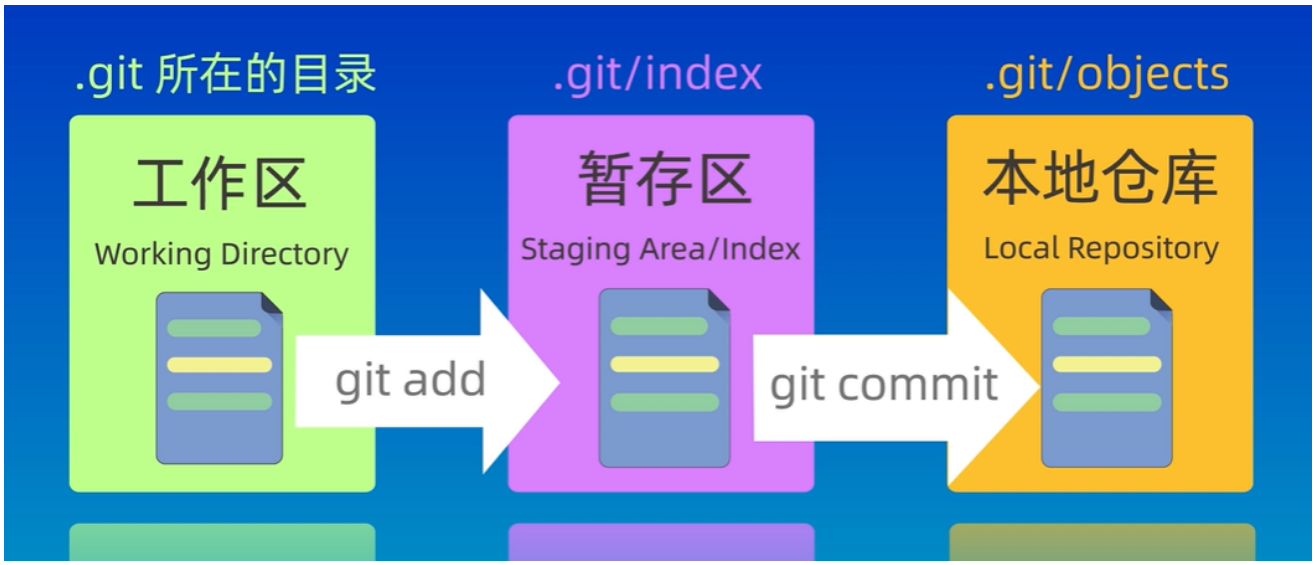
暂存区(是一种临时存储区域)

用于保存即将提交给Git仓库的修改内容

简单来说,暂存区就是个中间区域,用于临时存放即将提交的修改内容

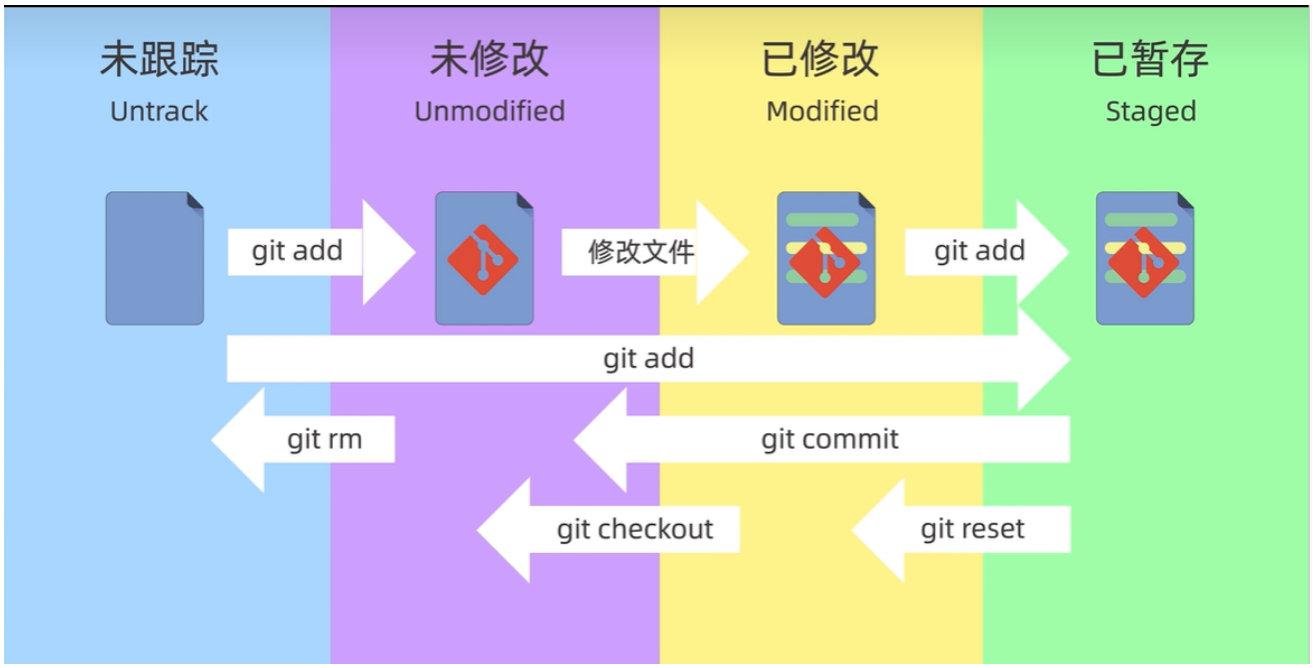
本地仓库

就是用git init命令创建的那个仓库,包含了完整的项目历史和元数据,是Git存储代码和版本信息的主要位置



git add相当于把修改部分装箱任何用git commit 一起提交到本地仓库中

(2) 四种文件状态



未跟踪

我们新建的还没有被git管理的文件

未修改

我们已经被Git管理起来,但是文件的内容没有发生变化,还没有被修改的

已修改

我们修改了文件,但是还没有添加到暂存区里面

已暂存

我们修改后,已经添加到暂存区域内的文件

以下是对这些指令的详细解释：

1.git add

作用：将文件从 “未跟踪” 或 “已修改” 状态转移到 “已暂存” 状态，是准备提交（commit）的关键步骤。

对于未跟踪（Untrack）的文件，执行 git add 文件名 后，文件进入 “未修改”（Unmodified）状态，成为 Git 版本库的跟踪文件。

对于已修改（Modified）的文件，执行 git add 文件名 后，文件进入 “已暂存”（Staged）状态，标记为 “即将提交的修改”。

2. git rm

作用：从 Git 版本库中移除文件，同时也会删除工作区的文件。执行后，文件从 “未修改”“已修改” 或 “已暂存” 状态回到 “未跟踪” 状态（因为文件被彻底删除了）。

3. git commit

作用：将 “已暂存”（Staged）的文件提交到本地版本库，生成一个版本记录。提交后，文件回到 “未修改”（Unmodified）状态，表示当前工作区文件与版本库中的最新版本一致。

4. git checkout

作用：可以撤销文件的修改，将 “已修改”（Modified）状态的文件恢复到 “未修改”（Unmodified）状态（即恢复到上一次 commit 或 add 时的版本）。也可用于切换分支等场景，图中主要体现其撤销文件修改的功能。

5. git reset

作用：可以将 “已暂存”（Staged）的文件回退到 “已修改”（Modified）状态，取消暂存操作（常用 git reset HEAD 文件名 命令）。也可用于回退版本历史，图中主要体现其取消暂存的功能。

总结来说，这些指令围绕 Git 文件的四个状态（未跟踪、未修改、已修改、已暂存） 实现流转和控制，是 Git 版本管理的核心操作。

4.添加和提交文件

git status用于查看当前仓库的状态,可以查看有哪些文件以及文件当前处于怎样的一个状态

创建文件的方式 vim VSCode echo等

ls

- 直接执行 `ls`：列出当前目录下的**可见文件和目录**（不包括以 `.` 开头的隐藏文件），按字母顺序排列。
- `ls -a`：显示**所有文件**（包括隐藏文件，如 `.bashrc`、`.git` 等）。
- `ls -l`：以**长格式**显示文件信息，包括权限、所有者、大小、修改时间等详细内容。
- `ls -t`：按**修改时间**排序（最新修改的文件排在前面）。
- `ls -r`：**反向排序**（配合 `-t` 则按时间从旧到新排，配合字母排序则逆序）。

cat的作用

`cat` 是 Linux/Unix 系统中常用的命令，核心作用是**连接并显示文件内容**，其名称来源于英文 "concatenate"（连接）的缩写。

主要用法和功能包括：

1. **查看文件内容**：直接输出文件的全部内容到终端。

   例：`cat filename.txt` 会在终端显示 `filename.txt` 的内容。

2. **合并文件内容**：将多个文件的内容拼接后输出（可重定向到新文件）。

   例：`cat file1.txt file2.txt > combined.txt` 会把 `file1.txt` 和 `file2.txt` 的内容合并到 `combined.txt` 中。

3. **创建简单文件**：通过输入内容并以 `Ctrl+D` 结束，快速创建小型文本文件。

   例：`cat > newfile.txt` 后输入内容，按 `Ctrl+D` 保存，即可生成 `newfile.txt`。

4. **显示行号**：配合 `-n` 参数，显示内容时在每行前加上行号。

   例：`cat -n script.sh` 会带行号显示 `script.sh` 的内容。

```
 yiny@Ares  ~/learn-git  ls
my-repo     remote-repo
 yiny@Ares  ~/learn-git  cd my-repo
 yiny@Ares  ~/learn-git/my-repo  main  git status
On branch main

No commits yet

nothing to commit (create/copy files and use "git add" to track)
 yiny@Ares  ~/learn-git/my-repo  main  echo "这是第一个文件"
> file1.txt
 yiny@Ares  ~/learn-git/my-repo  main  ls
file1.txt
 yiny@Ares  ~/learn-git/my-repo  main  cat file1.txt
这是第一个文件
```

```
 yiny@Ares  ~/learn-git/my-repo  main  git status
On branch main

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        file1.txt

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

git add将文件提交到暂存区

```
yiny@Ares  ~/learn-git/my-repo  main  git add file1.txt
yiny@Ares  ~/learn-git/my-repo  main +  git status
On branch main

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   file1.txt
```

git commit只会提交暂存区中的文件,而不会提交工作区中的任何文件

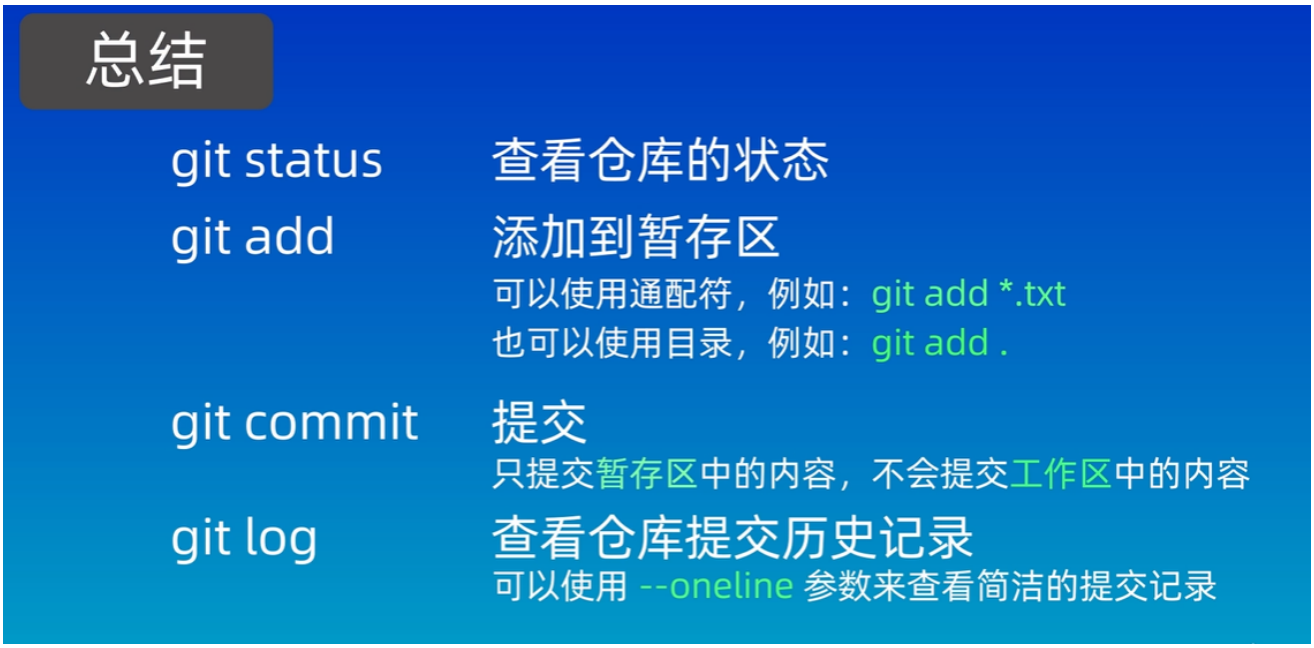
git commit -m "第一次提交" 用-m参数来指定提交的信息,这个信息会被保存到仓库中,如果不用,就会进入到一个交互式页面,默认用vim来编辑提交信息

git add

git add . 提交所有

git log 查看提交信息

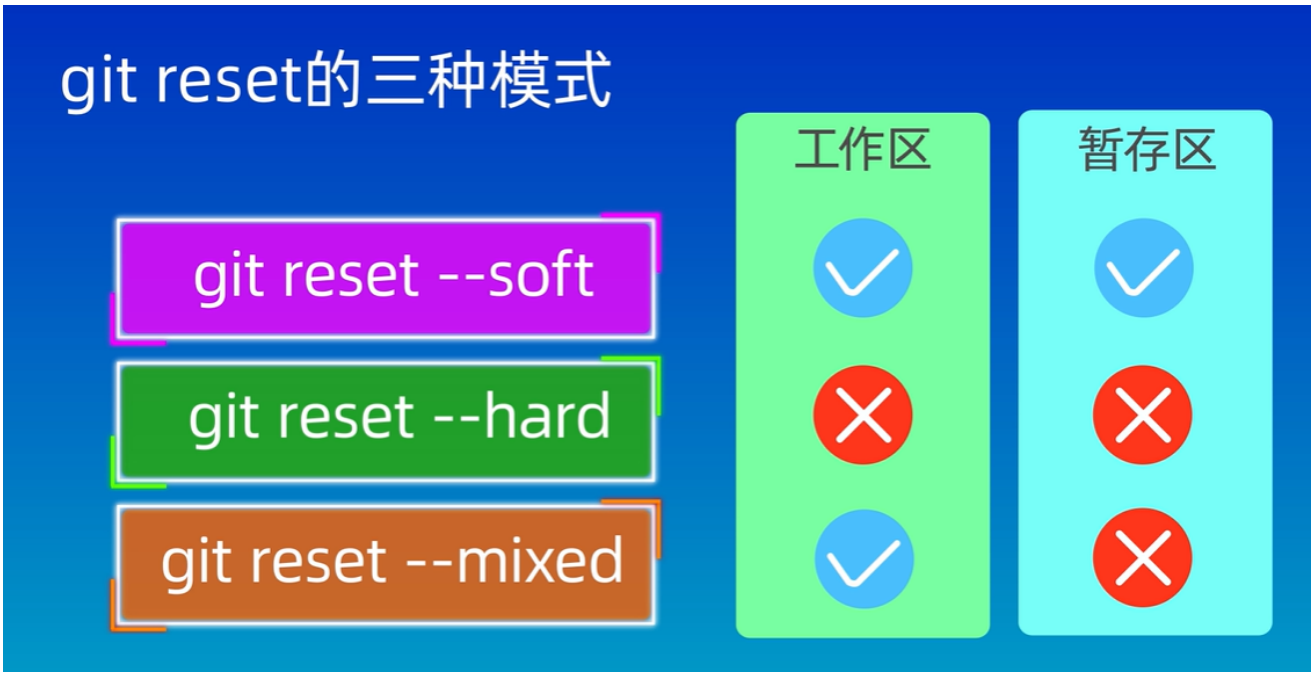
git log --online 只显示每次提交的ID和提交信息



5.了解git reset命令

用于回退版本,可以退回到之前的某一个提交的状态

git reset的三种模式



git reset --soft

soft参数表示回退到某一版本,并且保留工作区和暂存区的所有修改内容

使用场景:我们连续提交多个版本,太多提交没有意义,可以合并成一个版本的时候,就可以通过这个参数来进行回退之后重新提交,在重新提交之前不需要执行以下git add操作来将变动的内容重新添加到暂存区,因为暂存区并没有被清空

git reset --hard

hard参数表示回退到某一版本,并且丢弃工作区和暂存区的所有修改内容

使用场景:真的要放弃目前本地的所有修改内容的时候,会删除这两个版本之间的工作区和暂存区的所有修改内容

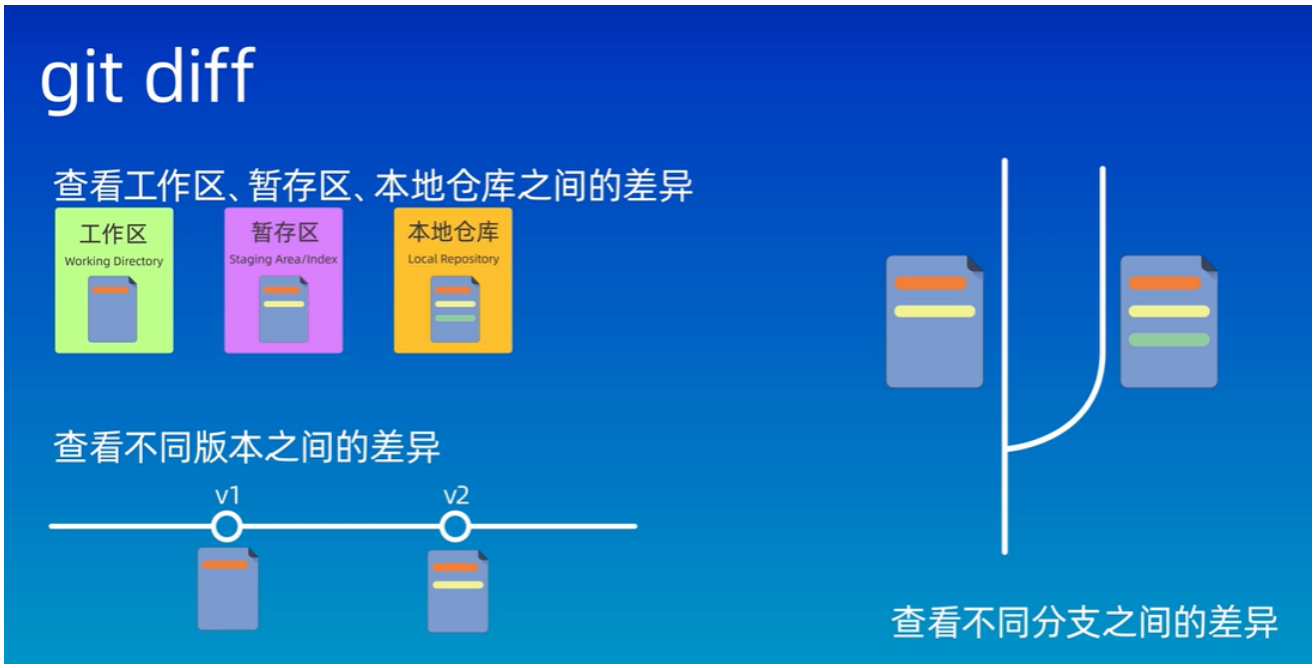
git reset --mixed(默认指令)

mixed参数表示回退到某一版本,并且只保留工作区的修改内容而丢弃暂存区的修改内容

使用场景:我们连续提交多个版本,太多提交没有意义,可以合并成一个版本的时候,就可以通过这个参数来进行回退之后重新提交,在重新提交之前需要执行以下git add操作来将变动的内容重新添加到暂存区

6.使用git diff查看差异

默认比较的是**工作区和暂存区**之间的差异内容,显示发生更改的文件以及更改的详细信息



git diff HEAD比较工作区和版本库之间的差异

git diff --cached比较暂存区和版本库之间的差异

git diff 后面加上两个id就可以比较两个版本之间的差异

git diff HEAD~(HEAD^) HEAD HEAD~或者HEAD^表示上一个版本,HEAD表示当前版本,用来比较当前版本和上一个版本的差异

git diff HEAD~2 HEAD 表示提交之前的第二个版本

```
git diff HEAD~ HEAD
git diff HEAD^ HEAD
git diff HEAD~2 HEAD
git diff HEAD~3 HEAD
```

加上文件名,比较这个文件下的差异

```
git diff HEAD~3 HEAD file3.txt
```

git diff 加上两个分支名,比较两个分支之间的差异

7.如何从版本库中删除文件

第一个方法

第一步在本地工作区删除文件 linux系统用rm指令,Windows之间放到回收站

git ls-files查看暂存区中的内容

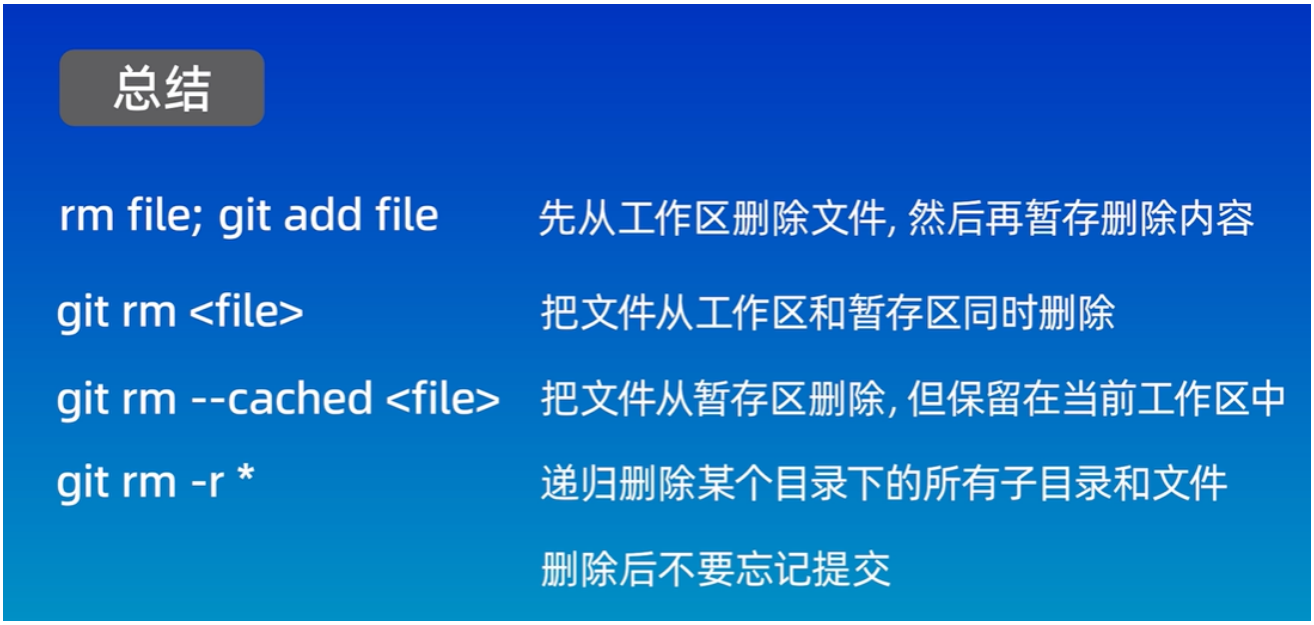
第二步执行git add .指令,告诉Git要删除这个文件

第二个方法

使用git rm这个命令

```
~/learn-git/my-repo  main +  git rm file2.txt
rm 'file2.txt'
~/learn-git/my-repo  main +  git status
On branch main
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        deleted:    file1.txt
        deleted:    file2.txt
```

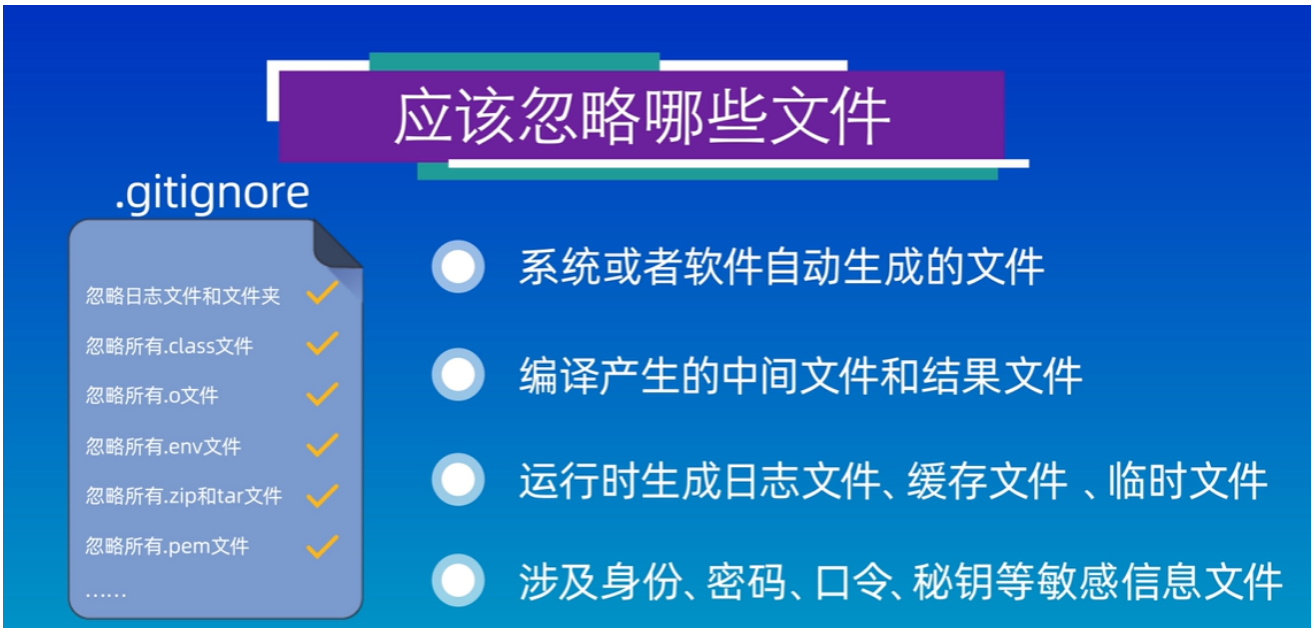
所有方法最后都要git commit提交到版本库中



8. .gitignore忽略文件

生效的前提是这个文件不能是已经被添加到版本库中的文件

这个文件的作用可以让我们忽略掉一些不应该被加入到版本库中的文件,这样可以让我们的仓库体积更小,更加干净



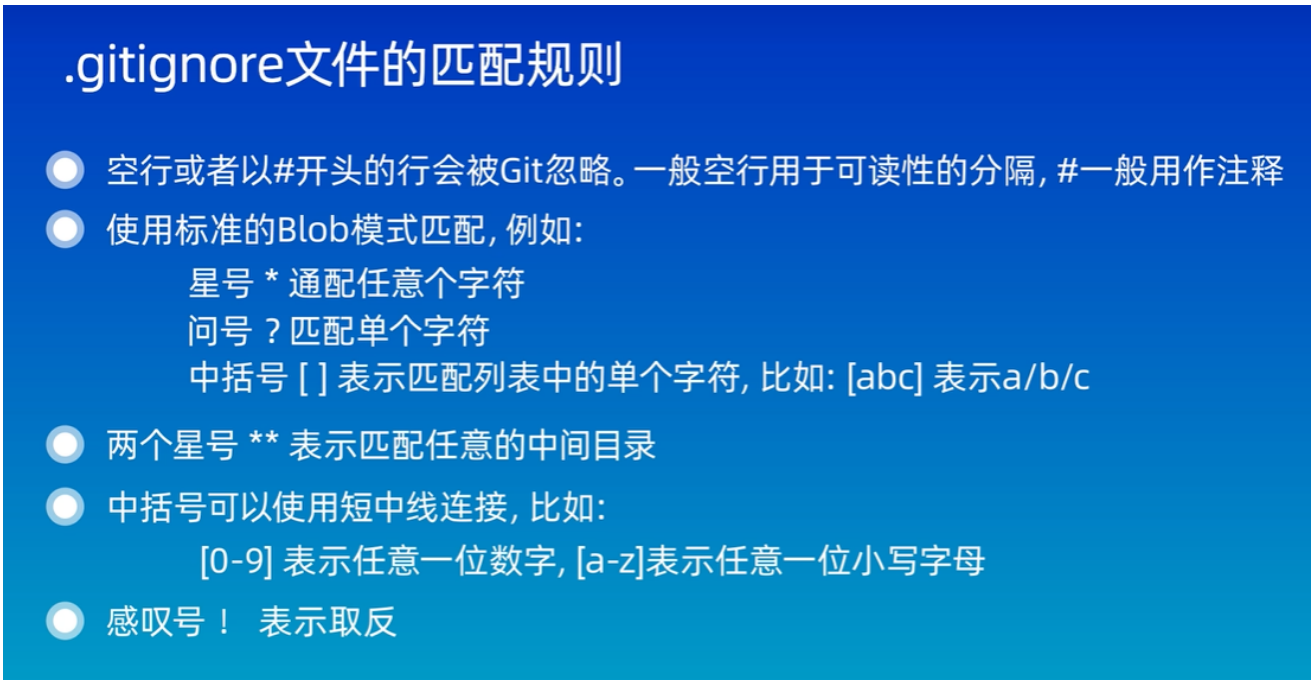
*.log忽略所有日志

Git默认是不会将空的文件夹添加到仓库里面的

git status -s

查看简略模式

文件夹写入是以/结尾的



例子

```
# 忽略所有的 .a 文件
*.a

# 但跟踪所有的 lib.a, 即便你在前面忽略了 .a 文件
!lib.a

# 只忽略当前目录下的 TODO 文件, 而不忽略 subdir/TODO
/TODO

# 忽略任何目录下名为 build 的文件夹
build/

# 忽略 doc/notes.txt, 但不忽略 doc/server/arch.txt
doc/*.txt

# 忽略 doc/ 目录及其所有子目录下的 .pdf 文件
".gitignore" 17L, 383B
```

9.SSH配置和克隆仓库

10.关联本地仓库和远程仓库

11.gitee使用和gitlab部署

12.在VSCode中使用Git

13.Git的分支