git工具链的笔记

一、仓库初始化 & 远程仓库

`git init`

作用：初始化一个本地 Git 仓库

`git remote add origin <url>`

作用：绑定远程仓库

`git remote -v`

作用：查看远程仓库地址

二、状态查看

`git status`

作用：查看当前仓库状态

场景：

- 修改了哪些文件?
- 当前在哪个分支?
- ……

`git diff`

作用：查看**未 add** 的修改

`git diff --staged`

作用：查看**已 add、未 commit** 的修改

三、提交相关

`git add <file>` / `git add .`

作用：把修改加入暂存区

```
git commit -m "message"
```

作用：生成一次提交（message->自己写的版本快照）

```
git log
```

作用：查看提交历史

```
git log --oneline --graph --all
```

作用：可视化分支历史

```
git show <commit-id>
```

作用：查看某一次提交的具体改动

四、推送 & 拉取（本地 ↔ 远程）

```
git push
```

作用：推送当前分支到远程

```
git push -u origin main
```

作用：首次推送并建立分支关联

场景：第一次 push 一个分支（只做一次）

```
git pull
```

作用：拉取远程并合并

```
git pull --rebase
```

作用：拉取远程并"整理式合并"

```
git fetch
```

作用：只更新远程信息，不改本地代码

五、分支

`git branch`

作用：查看本地分支

`git branch <branch>`

作用：创建分支

`git branch -a`

作用：查看所有分支（含远程）

`git switch <branch>`

作用：切换分支

`git switch -c <branch>`

作用：创建并切换分支

`git merge <branch>`

作用：合并分支

`git branch -d <branch>`

作用：删除分支

六、撤销 & 回退

`git restore <file>`

作用：撤销**未 add** 的修改
场景：写崩了，想回到上一次 commit

`git restore --staged <file>`

作用：撤销 add
场景：加错文件进暂存区

`git reset --soft HEAD~1`

作用：撤销上一次 commit，保留修改
场景：commit 信息写错了

`git reset --hard HEAD~1`

作用：强制回退，丢弃修改

七、临时保存

`git stash`

作用：临时保存当前修改

场景：改到一半，突然要切分支 / pull

`git stash pop`

作用：恢复临时保存

场景：切回后继续工作

八、挑提交 & 改历史

`git cherry-pick <commit-id>`

作用：把某一次提交"复制"到当前分支

场景：修 bug 提交要补到别的分支

`git rebase -i HEAD~n`

作用：交互式修改提交历史

场景：

- 合并多个提交
- 改 commit 信息
- 清理提交记录

九、辅助命令

`git blame <file>`

作用：查看每一行是谁写的

场景：协作项目定位责任 / 逻辑来源

`git tag`

作用：给版本打标签

场景：发布版本，如（v1.0.0）

遇到的某些场景

一、第一次初始化仓库并远程

一般流程

```
git init
git status
git remote add [网址].git
git remote -v
git commit -m "message"
git push -u origin main
git push
```

二、新root commit发现与原仓库root内容不符报错

场景: 远程仓库的 `main` 分支上已经有内容（比如 README / .gitignore / 初始化提交），而本地是新的**root commit**，所以 Git 不允许直接把本地历史“盖过去”，必须先把远程历史拉下来并整合。

A 方案（推荐）：先 pull（rebase）再 push ——保留远程已有内容

在当前目录直接执行：

```
git pull origin main --rebase
git push
```

可能会遇到 1：出现冲突

如果提示冲突:

1. 查看冲突文件：

   ```
   git status
   ```
2. 用 VS Code 打开冲突文件，选择保留方式（Accept Current / Incoming / Both）
3. 冲突解决后：

```
git add .
git rebase --continue
```

4. 最后推送：

```
git push
```

可能会遇到 2：提示需要设置用户名邮箱

```
git config --global user.name "xxx"
git config --global user.email "xxx"
```

B 方案（覆盖远程）：强推（会改写远程历史，谨慎）

如果确定远程仓库里那些内容不需要（比如只是空仓库初始化的 README），可以直接强推覆盖：

```
git push -u origin main --force
```