git基础知识

1. 三个核心区域

Git 工作时依赖三个相互关联的区域，所有操作本质是“在不同区域间移动文件”：

区域名称	英文	作用	关键特点
工作区	Working Directory	本地编写代码的文件夹	直观可见，文件状态分为“未跟踪（Untracked）”和“已跟踪（Tracked）”
暂存区	Staging Area（Index）	临时存放待提交的文件	相当于“提交缓冲区”，可灵活选择部分文件提交，不影响其他修改
本地仓库	Local Repository	存储项目完整版本历史	包含所有提交记录（Commit），是 Git 数据的核心存储区，以 `.git` 文件夹形式存在

2. 四个核心对象

Git 底层通过四种对象存储数据，所有版本控制功能均基于这些对象实现：

- **Blob（二进制大对象）**：存储单个文件的内容（如代码文件、文档），不包含文件名和路径。
- **Tree（树对象）**：存储目录结构，记录“哪些 Blob/Tree 属于当前目录”，相当于文件夹的“索引”。
- **Commit（提交对象）**：记录一次代码提交的完整信息，包括：
  - 指向当前提交的 Tree 对象（即“这次提交对应的目录结构”）；
  - 父提交对象的哈希值（形成版本历史链）；
  - 提交者信息（姓名、邮箱）、提交时间、提交说明。
- **Tag（标签对象）**：给特定 Commit 打“标签”，常用于标记版本（如 `v1.0.0` 发布版），便于快速定位重要版本。

3. 两种仓库类型

- **本地仓库**：在本地电脑上初始化的仓库（通过 `git init` 创建），仅自己可见，用于管理个人代码版本。
- **克隆仓库**：从远程仓库复制到本地的仓库（通过 `git clone` 创建），自动关联远程仓库地址，便于后续同步代码。

4.分支

（1）. 本质

分支是 Git 中**代码的“平行版本”**，本质是一个指向代码提交记录的“指针”。创建分支时，Git 会基于当前分支的代码完整复制一份，形成独立的修改路径，后续修改仅在该分支生效，不影响其他分支。

（2）. 核心作用

- **隔离开发**：个人 / 功能的修改在专属分支进行，不干扰主分支（`main`）的稳定代码。
- **保护主分支**：主分支始终保持可运行、可发布的状态，新功能 / 修复在分支验证通过后再合并。
- **并行协作**：多人同时开发不同功能，各自在分支工作，避免代码互相覆盖。

- **风险可控**：分支开发中即使出错，删除分支即可，不会影响主分支和其他分支。