历史提交的代码快照（commit）和源代码本身有什么区别，也就是说其实commit和push的都是这个代码快照吗，那怎么通过代码快照就知道变动的源代码是什么了呢

**ChatGPT 说：**

这个问题涉及到 Git 的存储机制，Git 并不是像传统的版本控制工具（比如 SVN）那样存储“文件的变更”，而是采用 **快照（snapshot）** 方式存储整个项目的状态。我们来详细拆解你的问题：

**1. 代码快照（commit）和源代码本身有什么区别？**

* **源代码（Source Code）**：是你当前在 **工作区（Work Directory）** 里看到的代码，通常存储在你的文件系统中，供开发者编辑、运行和测试。
* **代码快照（Commit）**：是 Git 在 **本地仓库（Local Repository）** 中存储的代码版本。每次 git commit，Git **不会** 记录代码的增量变更，而是把当前所有文件的完整快照存入 .git 目录下，形成一个新的 commit。

✅ **区别**：

| **对比项** | **源代码（工作区）** | **代码快照（commit）** |
| --- | --- | --- |
| 位置 | 硬盘上的文件夹 | .git 目录下 |
| 是否可编辑 | ✅ 可以直接修改 | ❌ 只能通过 checkout 还原 |
| 结构 | 直接存储代码 | 以 **快照（commit ID）** 形式存储 |
| 版本 | 只有最新的代码 | 记录所有历史版本 |

**2. Commit 和 Push 的都是代码快照吗？**

是的，**commit 和 push 的都是代码快照（snapshot），而不是直接推送文件本身**。不过，它们的作用范围不同：

* **git commit**：在本地仓库（Local Repository）创建一个代码快照，生成一个 commit ID。
* **git push**：把本地仓库的代码快照推送到远程仓库（Remote Repository）。

⏳ **简化理解**：

rust

复制编辑

（1）Work Directory（源代码） --> git add --> Staging Area（暂存区）

（2）Staging Area --> git commit --> Local Repository（代码快照）

（3）Local Repository --> git push --> Remote Repository（远程仓库）

所以，push 并不是把当前文件夹的所有代码上传，而是把 **Git 记录的 commit 快照** 发送到远程仓库。

**3. Git 如何通过代码快照知道变动的源代码是什么？**

Git 使用的是 **内容地址存储（Content Addressable Storage）**，它的核心机制有三个重要的概念：

**① Blob（文件对象）**

* Git 内部不会存储单个文件的历史修改，而是每次 commit 都保存**完整的文件内容**（即代码快照）。
* 每个文件都被存储为 **Blob 对象**，并用 **SHA-1 哈希值** 作为唯一标识。

**② Tree（目录对象）**

* Git 以 **树（Tree）** 结构存储整个项目快照，每个 commit 其实是一个根目录的 tree，指向多个 blob（文件）。
* 类似于文件系统的结构，每个 tree 代表一个目录，每个 blob 代表一个文件。

**③ Commit（提交对象）**

* commit 记录了：
  + 当前 tree（整个项目的状态）。
  + 父 commit（前一个快照的引用）。
  + 作者、提交信息等元数据。

📌 **示例：Git 内部存储的 Commit 结构**

scss

复制编辑

commit A (快照)

├── tree (根目录)

│ ├── blob (README.md)

│ ├── blob (index.html)

│ ├── tree (src/)

│ ├── blob (main.js)

│ ├── blob (app.js)

当你修改 app.js 并 commit，Git **不会存储文件的增量**，而是创建一个新的 commit B：

sql

复制编辑

commit B (快照)

├── tree (根目录)

│ ├── blob (README.md) <--（和A相同，不变）

│ ├── blob (index.html) <--（和A相同，不变）

│ ├── tree (src/)

│ ├── blob (main.js) <--（和A相同，不变）

│ ├── blob (app.js) <--（新版本）

Git 通过 **SHA-1 哈希值** 确保相同内容的文件不会重复存储。

**4. Git 如何计算代码的变更？**

虽然 Git 存的是**完整快照**，但在查看 git log -p 或 git diff 时，Git 会自动计算**相邻快照的差异**：

* Git **不会存储 diff，而是根据快照比较得出差异**。
* 这样可以提高性能，并保证数据完整性。

**🛠 示例：查看某次提交的变更**

bash

复制编辑

git diff HEAD~1 HEAD

* HEAD~1：表示上一个 commit。
* HEAD：表示当前 commit。
* git diff 计算的是 HEAD 和 HEAD~1 的差异。

📌 **示例输出**

diff

复制编辑

diff --git a/app.js b/app.js

index abc123..def456 100644

--- a/app.js

+++ b/app.js

@@ -5,7 +5,7 @@

console.log("Hello World");

- let x = 5;

+ let x = 10;

* - let x = 5; （旧代码）
* + let x = 10; （新代码）

Git 并不存储这个 diff，而是 **根据完整快照计算它**。

**5. 总结**

| **问题** | **答案** |
| --- | --- |
| 代码快照（commit）和源代码有啥区别？ | 源代码是可修改的文件，commit 是不可变的快照，存储在 .git 目录里。 |
| commit 和 push 的都是代码快照吗？ | 是的，commit 在本地存快照，push 把本地快照推送到远程仓库。 |
| Git 怎么知道代码的变更？ | Git 存的是完整快照，但可以通过比较相邻快照自动计算差异（diff）。 |

Git 采用 **快照存储** 而不是 **增量存储**，这样既保证了数据完整性，又可以快速计算变更，让 git checkout 和 git reset 变得高效。

**💡 你可以自己实验**

1. 在 Git 仓库里运行：

bash

复制编辑

git log --stat

* + 这会列出每次 commit 影响了哪些文件。

1. 运行：

bash

复制编辑

git cat-file -p HEAD

* + 这会显示 HEAD（最新 commit）的元数据。

1. 运行：

bash

复制编辑

git diff HEAD~1 HEAD

* + 这会显示最近一次提交的代码变化。

这样你就能更直观地理解 Git 的快照存储原理了！ 🚀