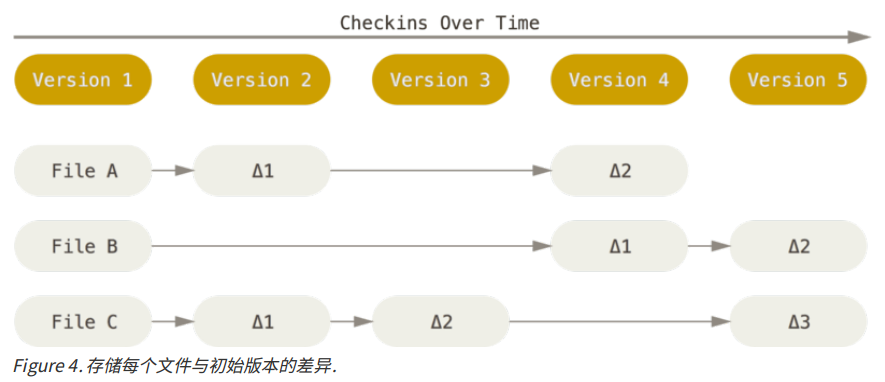
# Git分布式板控制

## Git基本概念

**诞生宗旨**：

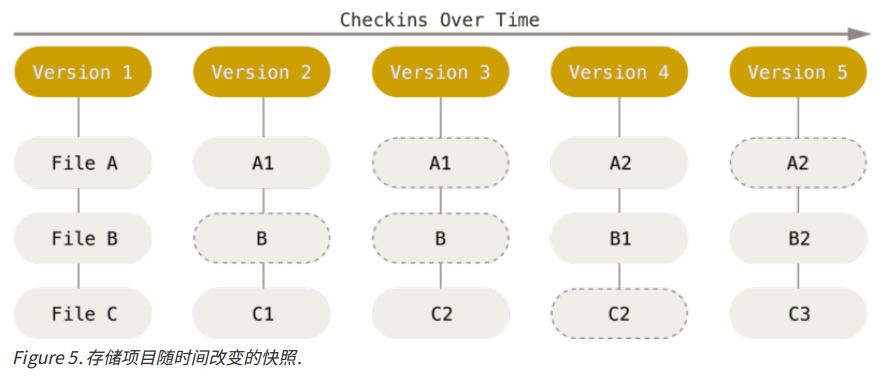
• 速度  
• 简单的设计  
• 对非线性开发模式的强力支持（允许成千上万个并行开发的分支）  
• 完全分布式  
• 有能力高效管理类似 Linux 内核一样的超大规模项目（速度和数据量）

**直接记录快照，而非差异比较:**



Git 不按照以上方式对待或保存数据。 反之，Git 更像是把数据看作是对小型文件系统的一组快照。 每次你提交更新，或在 Git 中保存项目状态时，它主要对当时的全部文件制作一个快照并保存这个快照的索引。

为了高效，如果文件没有修改，Git 不再重新存储该文件，而是只保留一个链接指向之前存储的文件。 Git 对待数据更像是一个 **快照流**。



**Git 保证完整性.**

Git 中所有数据在存储前都计算校验和，然后以校验和来引用。 这意味着不可能在 Git 不知情时更改任何文件内容或目录内容。 这个功能建构在 Git 底层，是构成 Git 哲学不可或缺的部分。 若你在传送过程中丢失信息或损坏文件，Git 就能发现。

Git 用以计算校验和的机制叫做 SHA-1 散列（hash，哈希）。 这是一个由 40 个十六进制字符（0-9 和 a-f）组成字符串，基于 Git 中文件的内容或目录结构计算出来。

SHA-1 哈希看起来是这样：  
24b9da6552252987aa493b52f8696cd6d3b00373

Git 中使用这种哈希值的情况很多，你将经常看到这种哈希值。 实际上，Git 数据库中保存的信息都是以文件内容的哈希值来索引，而不是文件名。

**Git 一般只添加数据。**

**三种状态.**

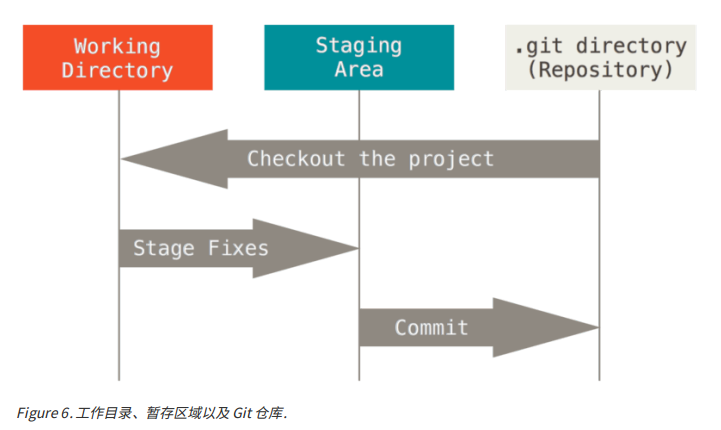
Git 有三种状态，你的文件可能处于其中之一：已提交（committed）、已修改（modified）和已暂存（staged）。

已提交表示数据已经安全的保存在本地数据库中。

已修改表示修改了文件，但还没保存到数据库中。

已暂存表示对一个已修改文件的当前版本做了标记，使之包含在下次提交的快照中。

由此引入 Git 项目的三个工作区域的概念：Git 仓库、工作目录以及暂存区域。



基本的 Git 工作流程如下：

1. 在工作目录中修改文件。

2. 暂存文件，将文件的快照放入暂存区域。

3. 提交更新，找到暂存区域的文件，将快照永久性存储到 Git 仓库目录。

**检查配置信息**

使用 git config --list 命令来列出所有 Git 当时能找到的配置。