Git分布式版本控制工具

1、概述

1.1 Git应用场景

- **版本控制**：Git被广泛用于软件开发项目中进行版本控制，允许团队协作开发，并追踪代码的变化历史。
- **分支管理**：Git提供了强大的分支管理功能，使得团队可以并行开发不同的功能或修复不同的bug，然后将它们合并到主代码库中。
- **备份与恢复**：通过Git，可以轻松地备份项目的整个历史记录，并在需要时进行恢复。
- **同步与共享**：Git通过远程仓库的方式，使得团队成员可以方便地同步和共享代码。

1.2 版本控制的方式

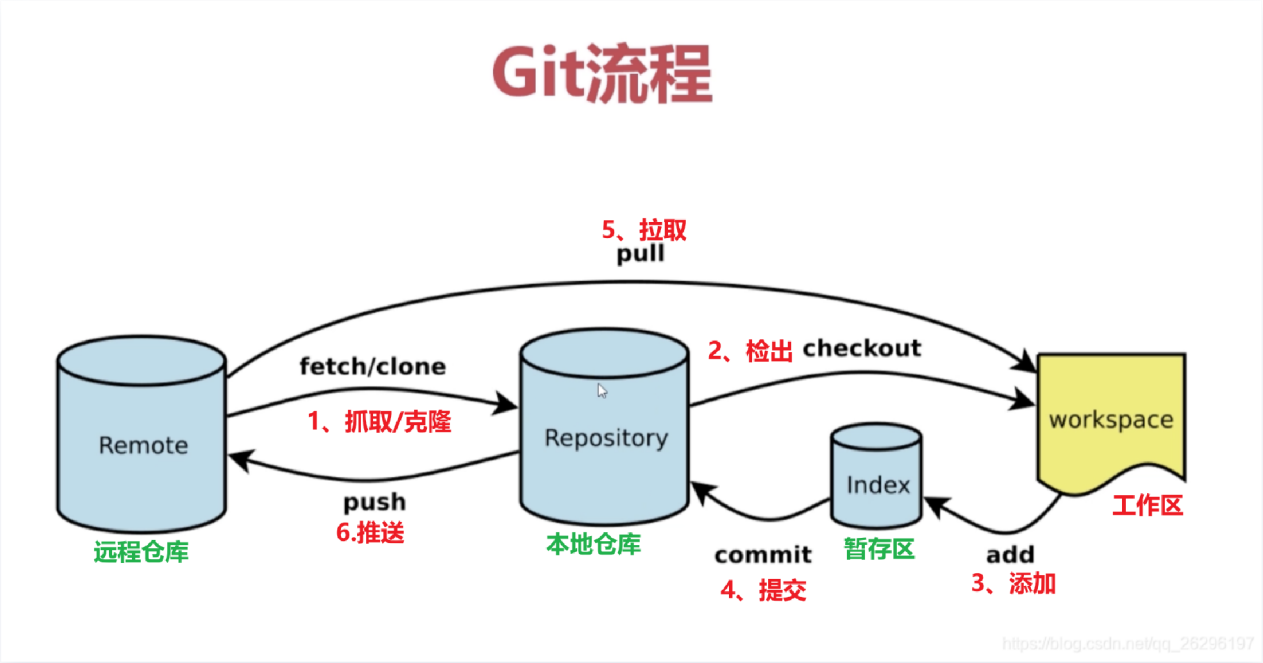
- **集中版本控制工具**
  - 简介：集中版本控制工具是指所有的文件版本都存储在中央服务器上，团队成员通过与中央服务器进行交互来管理文件版本。
  - 举例：SVN（Subversion）是一种流行的集中版本控制工具，它将所有的文件版本都存储在中央仓库中，并通过客户端与中央仓库进行交互。
- **分布式版本控制工具**
  - 简介：分布式版本控制工具是指每个开发者都拥有一份完整的代码仓库，并可以在本地进行版本控制操作，不需要与中央服务器直接交互。
  - 举例：Git是一种流行的分布式版本控制工具，每个开发者都可以在本地进行代码提交、分支管理等操作，然后再与其他开发者进行同步。

1.3 Git的特点

- **本地仓库**：每个开发者都有一份完整的本地代码仓库，无需网络连接即可进行版本控制操作，提高了灵活性和效率。
- **共享版本库**：团队成员可以通过远程仓库轻松共享代码，实现协作开发，方便代码同步和团队合作，提高效率。
- **分支管理**：强大的分支管理功能允许并行开发不同功能，保持代码稳定性，便于实验性修改而不影响主要代码。
- **高效的性能**：Git采用高效算法和数据结构处理大型项目和大量文件，保持较高速度和效率，适用于各种规模的项目。

- **本地化操作**：操作本地化减少对网络连接的依赖，提高操作速度，使得即使在没有网络连接的情况下也可以进行版本控制操作。
- **灵活的分布式架构**：每个开发者拥有完整的代码仓库，无需与中央服务器直接交互，提高容错性和稳定性，即使中央服务器故障，开发者仍可继续工作并进行协作和同步。

1.4 Git的工作流程



- 名词解释：
  - **远程仓库**：存放在网络上的`共享版本库`，用于团队成员之间的`协作`和`代码共享`。
  - **本地仓库**：每个开发者在本地计算机上拥有的`完整代码副本`，用于进行版本控制操作。
  - **暂存区**：存放`待提交的修改`的区域，通过`git add`命令将工作区中的修改添加到暂存区。
  - **工作区**：开发者在本地文件系统中进行`编辑和修改`的区域，包含项目的实际文件。
- 基本工作流程：
  1. **抓取/克隆（fetch/clone）**：从`远程仓库`获取`最新的代码`到`本地仓库`。
  2. **检出（checkout）**：`切换`到指定的`分支`或者`版本`，开始工作或者查看代码。
  3. **添加（add）**：将工作区中的修改`添加到暂存区`，准备提交到本地仓库。
  4. **提交（commit）**：将暂存区中的修改`提交到本地仓库`，形成一个`新的提交记录`。
  5. **拉取（pull）**：从远程仓库`获取`最新的代码更新到本地仓库，并`自动合并(merge)`到当前分支，相当于`fetch+merge`。
  6. **推送（push）**：将本地仓库中的提交`推送到远程仓库`，将本地的修改同步到共享版本库中。

2、Git安装与常用指令