Git使用介绍

内容

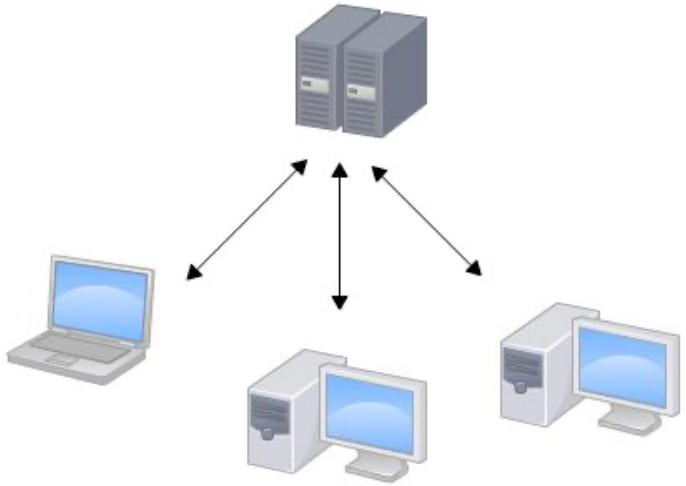
- **何为版本管理？**
  - ✓集中和分布式版本控制
- **何为Git？**
  - ✓Git是什么？
- **如何使用Git？**
  - ✓Git的操作
- **分布式协同开发**
  - ✓EduCoder中的Git

什么是版本控制（Revision Control）？

- **版本控制是对计算机程序、文档、数据等的更改和管理，它是软件配置管理的重要组成部分**
  - ✓实现跨区域、多人的协同开发
  - ✓记载和追踪一个或者多个文件的历史记录
  - ✓组织和保护软件制品：源代码和文档
  - ✓统计软件开发工作量
  - ✓跟踪记录软件开发过程

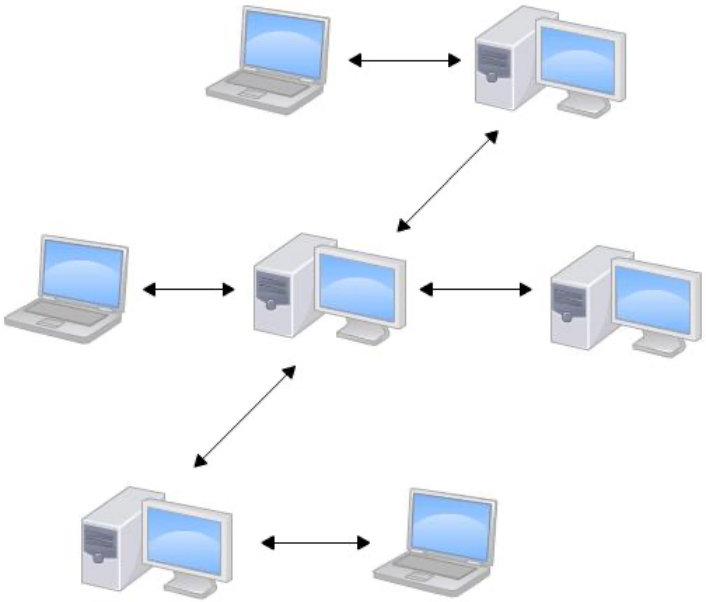
集中式版本控制

- **集中式版本控制**
  - ✓版本库集中存放在中央服务器之中
  - ✓开发前先从中央服务器取得最新版本
  - ✓开发完再把自己的工作推送给中央服务器
  - ✓中央服务器就好比是一个图书馆，你要改一本书，必须先从图书馆借出来，然后回到家自己改，改完后再放回图书馆
- **典型系统：CVS、SVN、ClearCase**
- **特点：高度依赖中央服务器、速度慢、难以有效支持协同开发**



分布式版本管理

- **分布式版本控制**
  - ✓有一个中央仓库
  - ✓开发前通过克隆在本机上拷贝一个完整的软件仓库
  - ✓开发完把自己工作提交到本地仓库中
  - ✓需要同步给协作者时再递交到中央仓库
  - ✓版本库分步存储于各协作者电脑中
- **典型系统：Git**
- **优点：不依赖中央服务器、可在本地开发、有效支持协同开发**



内容

- 何为版本管理？
  - ✓集中和分布式版本控制
- **何为Git？**
  - ✓Git是什么？
- **如何使用Git？**
  - ✓Git的操作
- **分布式协同开发**
  - ✓EduCoder中的Git

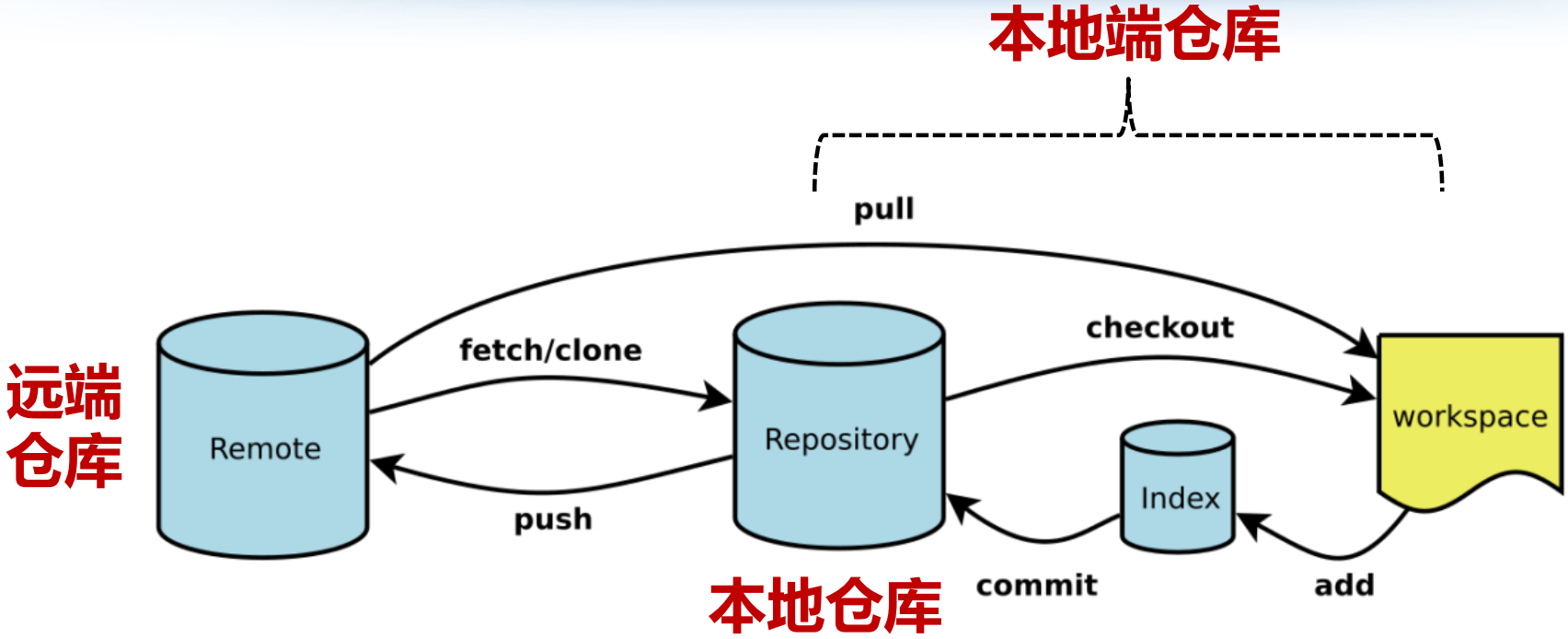
何为Git？

- Git 是一种**分布式的版本控制系统**，它支持存储代码、跟踪修订历史记录、合并代码更改等，可在需要时恢复较早的代码版本
  - ✓Linus Torvalds开发
  - ✓实现高效团队协作
  - ✓高可用性及冗余性
  - ✓庞大的社区支持
  - ✓行业标准

Git的基本功能

- **版本管理**
  - ✓管理各种源代码和文档、切换不同版本等
- **过程管理**
  - ✓跟踪开发过程、查阅历史记录等
- **代码评审**
  - ✓可视化评估代码质量，决断是否合并代码等；
- **扩展功能**
  - ✓企业DevOps自动化、代码Issue联动管理；
  - ✓… …

Git的工作流程

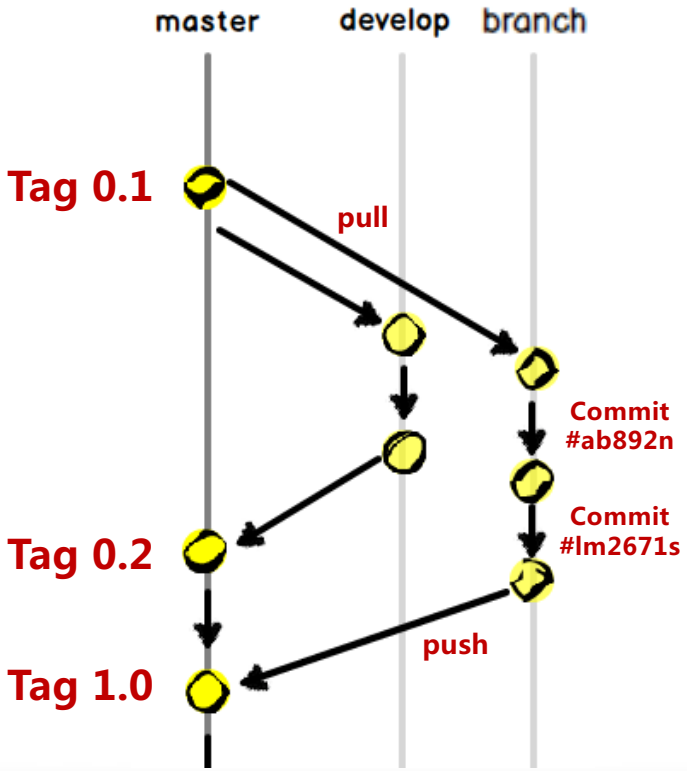


- Workspace：工作区
- Index / Stage：暂存区
- Repository：仓库区（或本地仓库）
- Remote：远程仓库

Git的基本工作原理

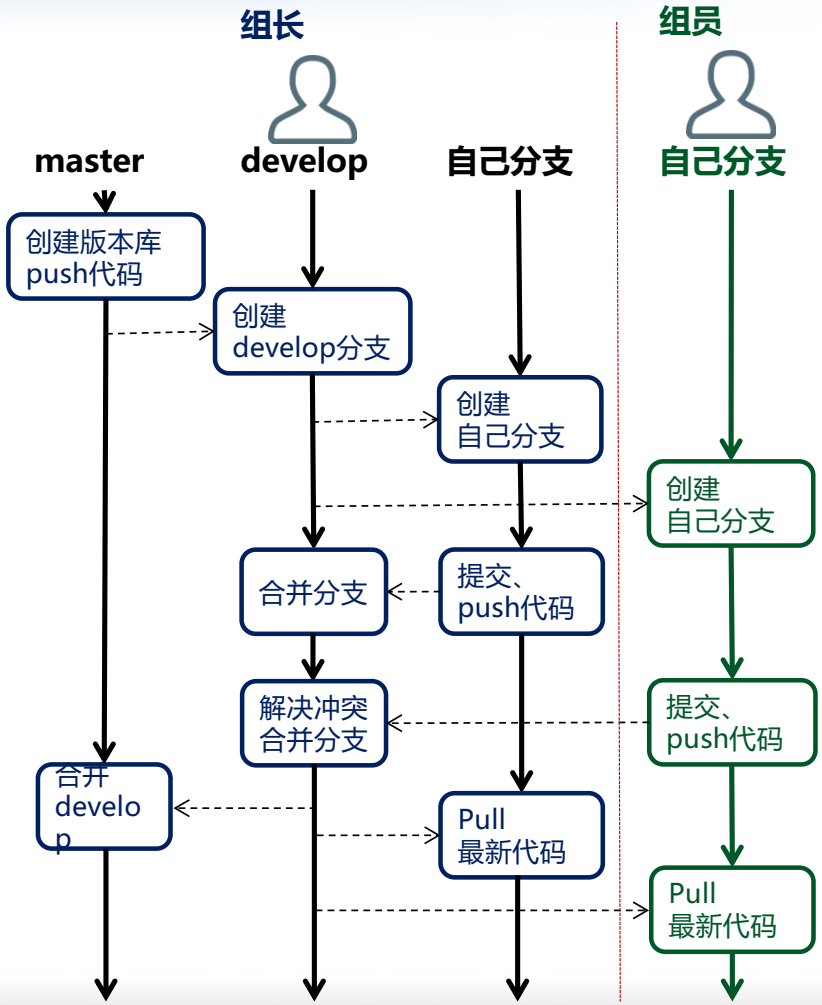
- 仓库（Repository）软件所有文件的完整修订历史
- 版本（Revision）代码库的编号方案，如Tag 0.1
- 分支（Branch）对代码库并行修改时的代码库副本
  - ✓如master, develop, branch1,…
- 提交（Commit）对分支的一次修订
- 下拉（Pull）将远程的一个分支读取并保存到本地分支
- 推送（Push）将本地分支代码发送到远程某个分支
- 合并（Merge）将对相同文件在不同分支的修改合并到一个分支中
- 冲突（Conflict）当两个分支中存在对同一文件的不同修改，并试图合并这两个分支时，就会发生冲突

master和develop是标准分支，master是缺省的主分支



Git的使用

- **分支管理流程：初始化过程（组长）**
  - ✓**1 创建版本库(默认master分支)，push初始代码**
  - ✓2 创建远程develop分支
- **分支管理流程：基本过程（组员）**
  - ✓**3 克隆代码，切换到develop分支**
  - ✓4 新建自己的分支my_branch（各组员分支名不同）
  - ✓5 修改my-branch：正常开发，修改完成后将修改的内容推送到远程my_branch分支
- **合并分支：基本过程（组长）**
  - ✓**6 切换develop分支，pull代码**
  - ✓7 合并组员的分支（my_branch）到develop分支
  - ✓8 解决冲突，然后将合并好的develop分支推送到远程
  - ✓9 将develop分支合并到master分支（这一步可以按实际需求，master上一般为稳定版本）



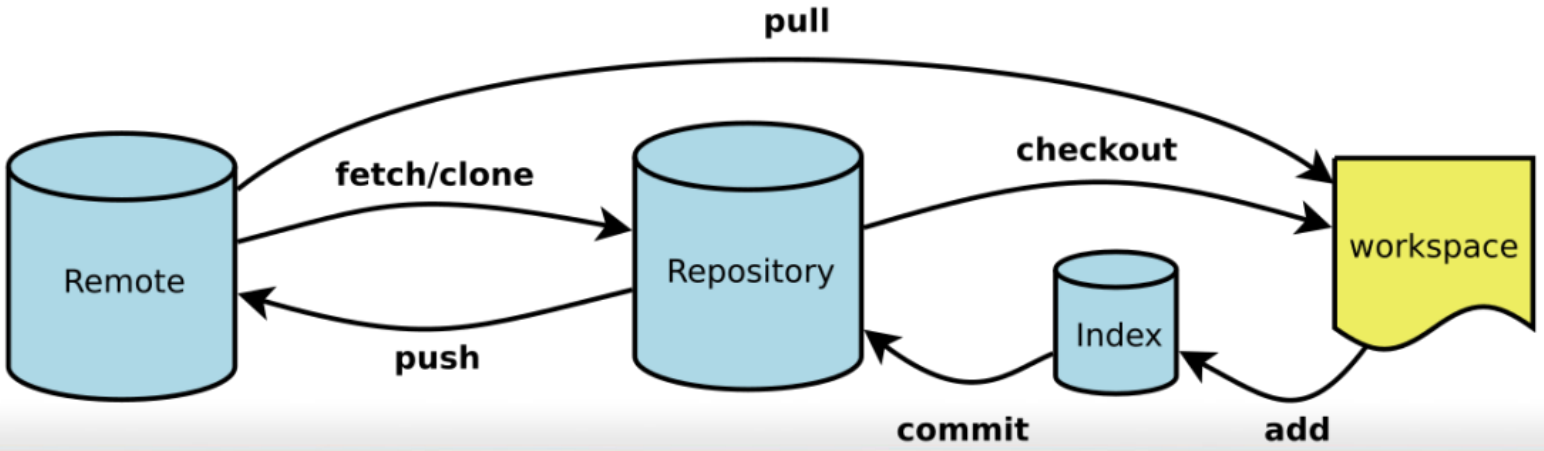
内容

- 何为版本管理？
  - ✓集中和分布式版本控制
- 何为Git？
  - ✓Git是什么？
- **如何使用Git？**
  - ✓Git的操作
- **分布式协同开发**
  - ✓EduCoder中的Git

1. 创建本地版本库

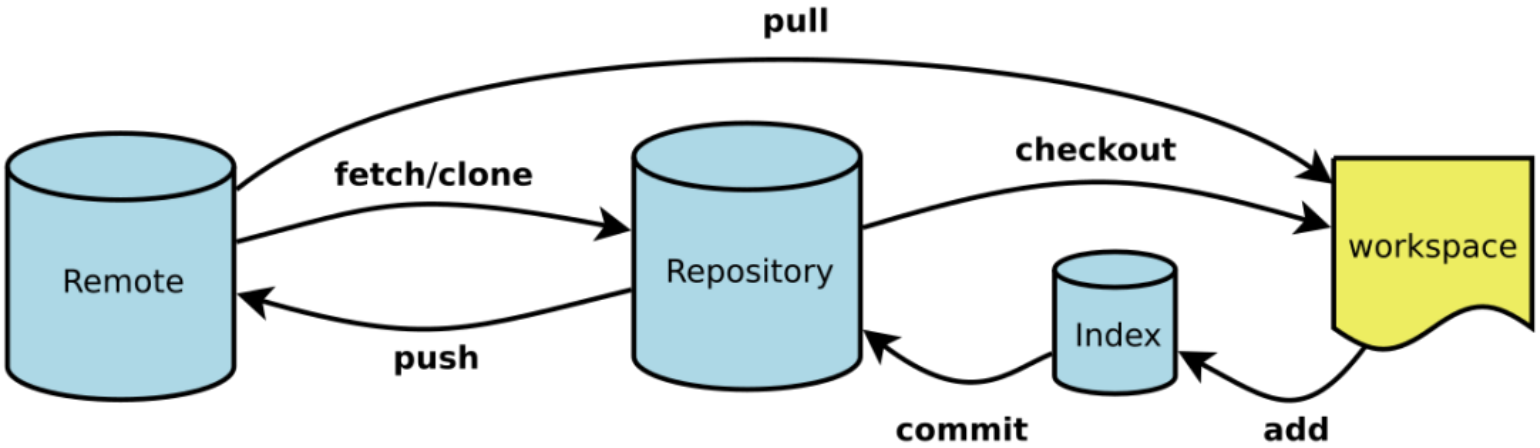
❑命令：git init

- ✓在执行完成 git init 命令后，Git 仓库会生成一个 .git 目录，该目录包含了资源的所有元数据
- ✓使用当前目录作为git仓库，直接在当前目录进行初始化
- ✓使用指定目录作为git仓库，在初始化命令之后加上指定目录路径，如 git init newrepo



2. 克隆远程版本库

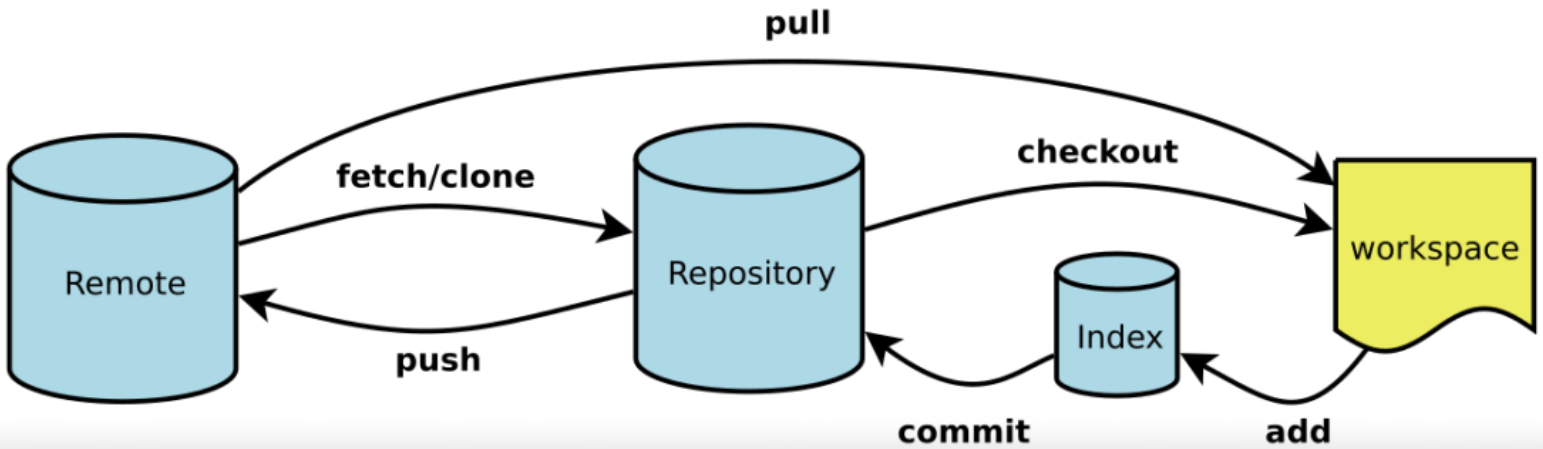
- **git clone <repo> <directory>**
  - ✓其中repo表示远程仓库目录，directory代表本地仓库目录
  - ✓将远程仓库中的代码拷贝项目到本地仓库



3. 在暂存区添加/撤销文件

- **git add [file1] [file2] ... or [dir]**
  - ✓ 将文件添加到暂存区

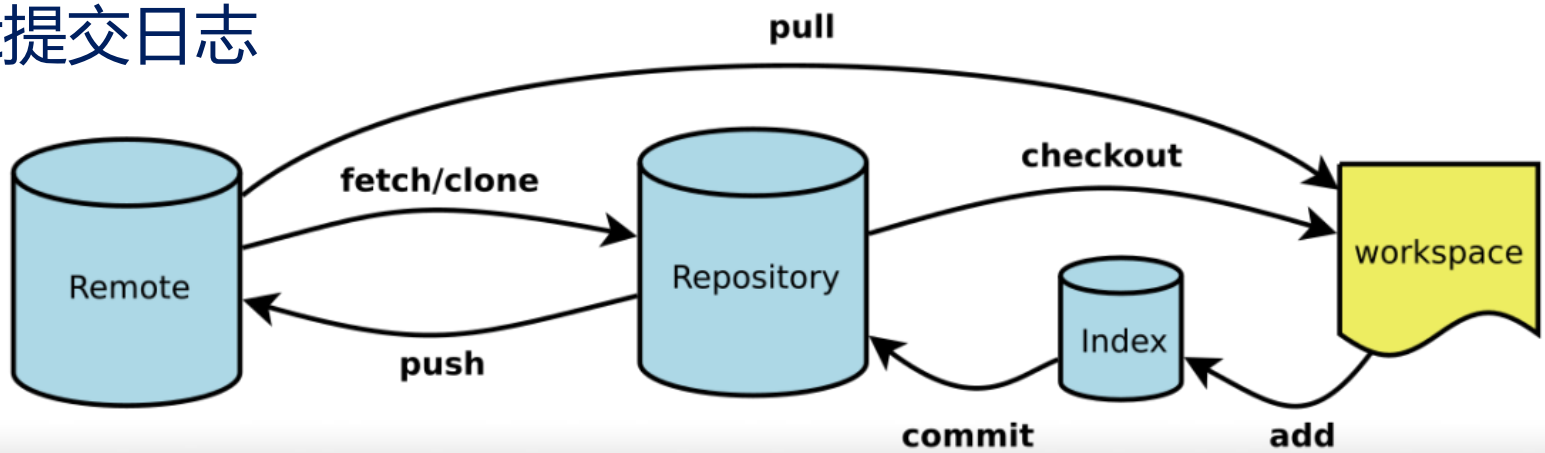
- **git checkout [file1]**
  - ✓撤销不需要的修改：



4. 提交修改到本地仓库

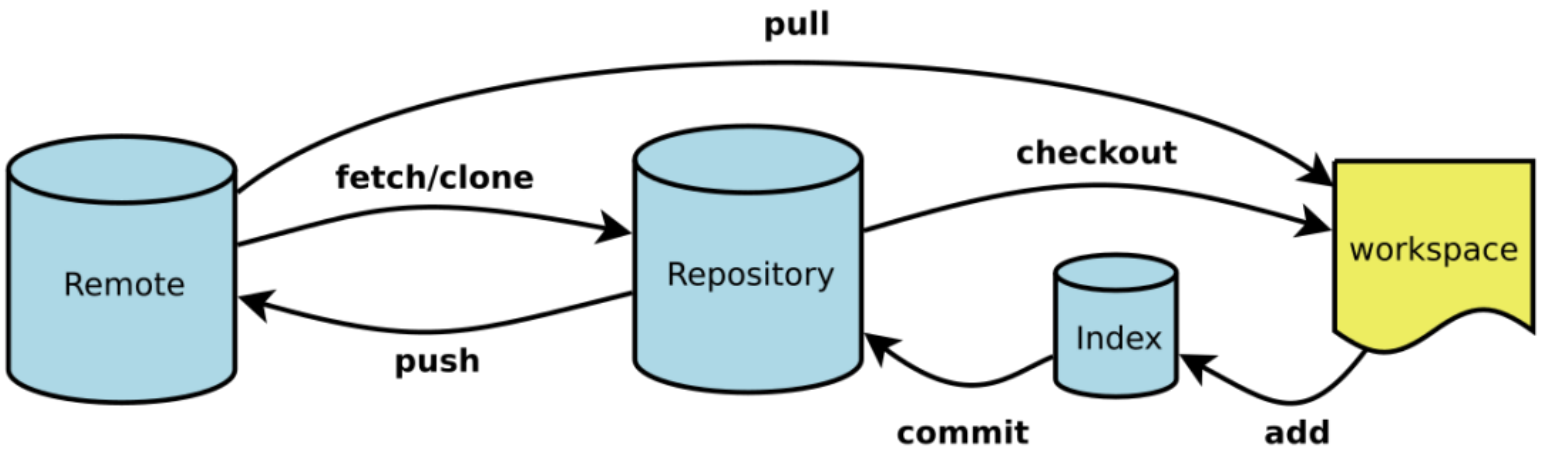
- **git commit -m [message]**
- **git commit [file1] [file2] ... -m [message]**
  - ✓将暂存区内容添加到本地仓库中

- **git log**
  - ✓查看commit提交日志



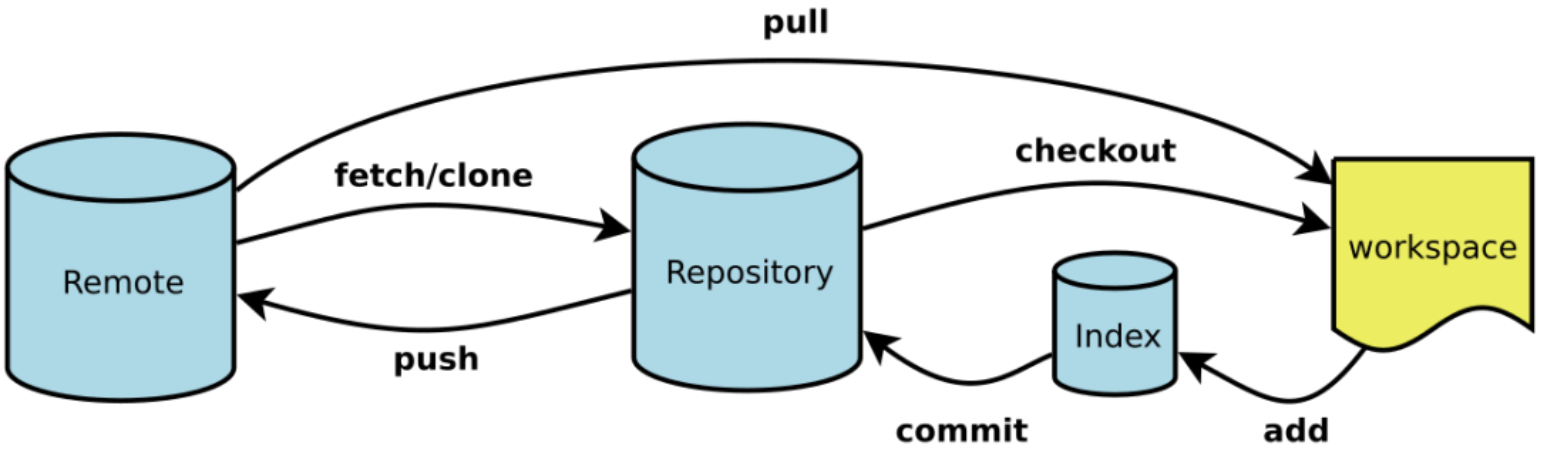
5. 添加远程版本库

- **git remote add “远程仓库名” “远程仓库地址”**
  - ✓添加远程版本库



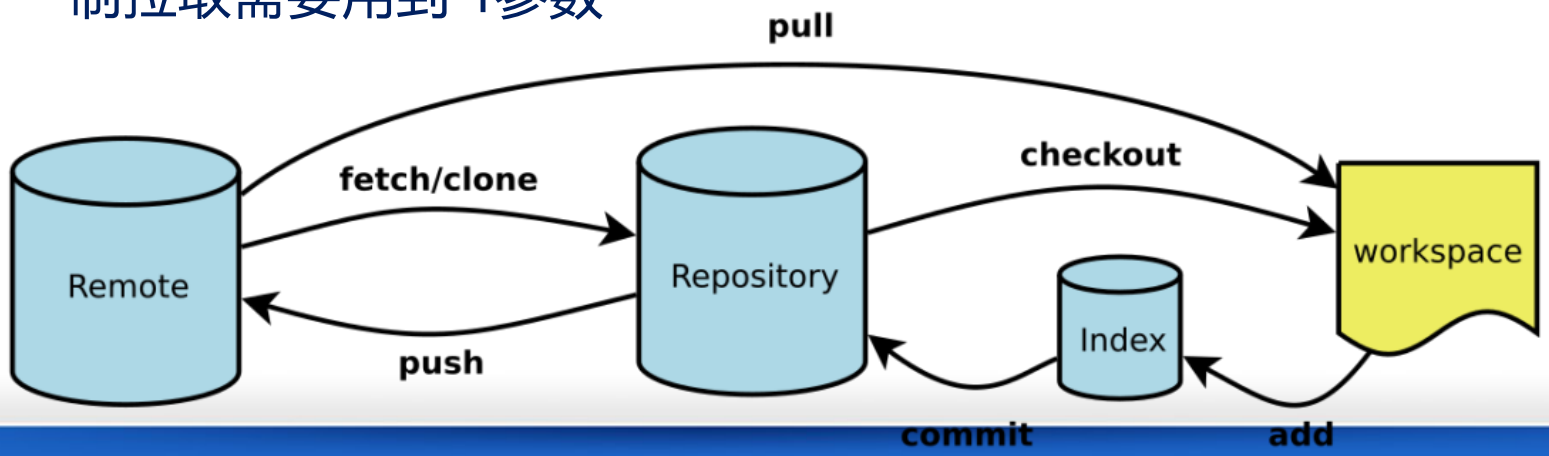
6. 推送本地内容到远程仓库

- **git push 远程仓库名 本地分支名 远程分支名**
  - ✓将你的修改推送到远程仓库，便于测试或者和团队中其他人协作
  - ✓推送本地内容时，会将所有未推送至远程仓库的内容，都提交到远程仓库



7. 拉取远程分支到本地

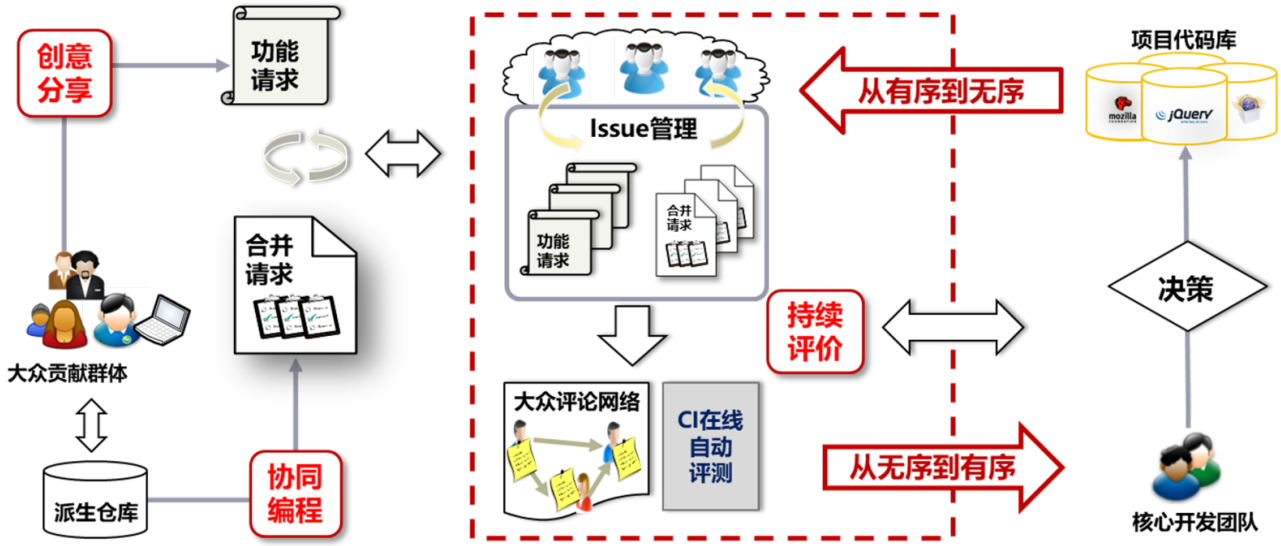
- **git pull 远程主机名 远程分支名:本地分支名**
  - ✓在团队协作中，如果别人对项目做了修改，而你需要将这些修改合并到你本地时，需要使用git pull命令
- **git pull 远程主机名 远程分支名:本地分支名 -f**
  - ✓如果远程分支和本地分支对同一内容做了修改，这将导致将远程分支修改合并到本地分支时会发生冲突
  - ✓可以选择直接强制拉取，使用远程分支的修改，覆盖本地分支的修改。强制拉取需要用到-f参数



内容

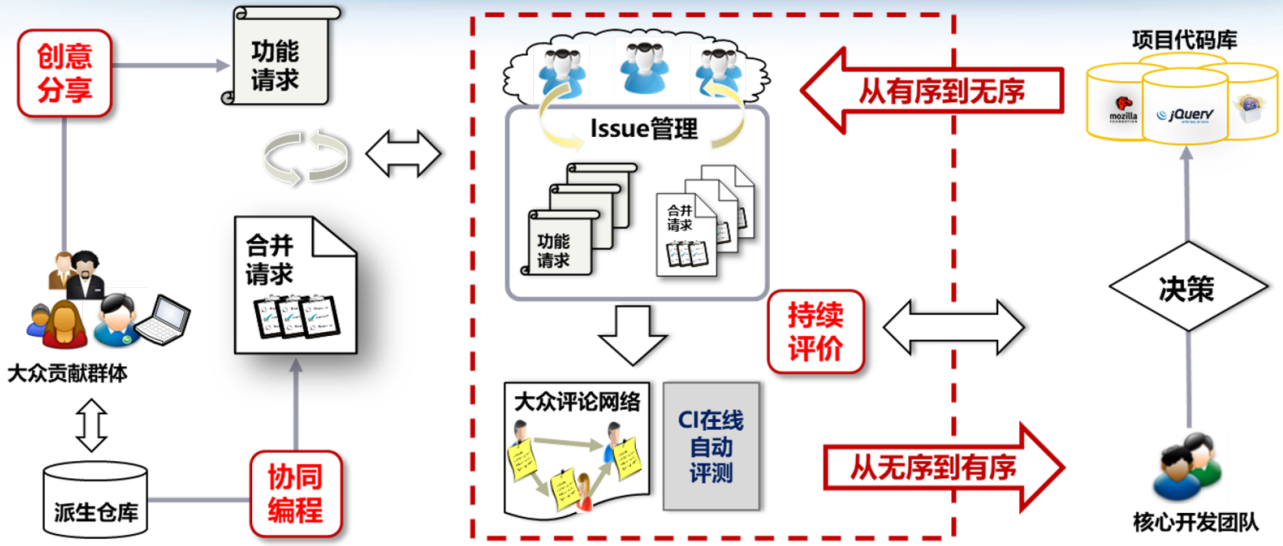
- ❑何为版本管理？
  - ✓集中和分布式版本控制
- ❑何为Git？
  - ✓Git是什么？
- ❑如何使用Git？
  - ✓Git的操作
- ❑分布式协同开发
  - ✓EduCoder中的Git

基于Pull-Request的群体协作模式



- 大众贡献开发者在使用开源仓库代码时发现代码的缺陷，或对代码有新的功能建议，想参与到项目的开发。此时其并不具备开源项目的写权限，须先将**这个开源项目 Fork 出自己的项目副本**
- 开发者在自己项目副本的分支完成编码，**提交到远端仓库**，而后需要发起一次**Pull-Request**，待源项目管理人员讨论审核通过后方可合并到源仓库分支

基于Issue的协同开发



- **在群体协同开发过程中，问题跟踪和开发规划等对于推进项目顺利实施具有重要影响，Git社区通过Issue跟进问题及开发计划**
  - ✓ Issue创建：项目管理员发布项目存在的缺陷或者需要增加的新功能，并将其视为一项开发任务，可设置任务类型、优先级、开始日期、结束日期等。
  - ✓ Issue评论：开发人员在Issue评论区域补充描述说明Issue或解决思路

基于Issue和Pull-Request模式的开发流程

- **EduCoder 协同开发平台提供了基于Issue和Pull-Request的协同开发支持功能，帮助项目管理者管理项目实施**
  - ✓ **创建项目和Issue**：组长基于开发项目模块创建项目，拆分项目需求创建对应Issue；
  - ✓ **项目Fork**：组员基于组长的项目Fork形成自己的案例项目副本；
  - ✓ **代码开发**：使用Git工具克隆副本代码到本地，根据组长指派的Issue进行代码开发，并将代码提交到项目副本远程仓库；
  - ✓ **项目讨论**：小组成员针对Issue展开探讨，交流问题并提供解决方案；
  - ✓ **项目跟进**：小组成员完成，更新Issue状态，组长根据Issue状态跟进项目进度；
  - ✓ **PR提交**： 小组成员在项目副本中创建PR，指定目标分支为组长指定的源项目分支；
  - ✓ **PR审核合并**：组长对提交的PR进行Code Review审核，审核通过合并不通过打回。

详细操作步骤参考：https://www.educoder.net/forums/2784

问题和讨论