**分布式版本控制**

Overview



1. 每个repository都是分布式对等的节点，只是选取一个作为管理中心而已。

2. 每个版本保存的是变更后的完整文件的快照，而不是保存差异。若未修改，则保存链接。

3. 几乎所有操作都是本地的。

4. SHA-1散列，作为文件内容的索引和校验。

File Lifecycle



1. 未跟踪: 不纳入git管理

2. 已跟踪: unmodified, modified, staged

3. add添加变更到staging area，commit提交staging area的变更到repository

Sharing and Updating Projects远程仓交互

fetch, pull(fetch+merge), push

**Branch分支(有向无环图):** 分支的本质，是一个指针/引用/地址，指向commit object

****

blob object: 文件的快照，以SHA-1值为文件名

tree object: 与目录结构关联，记录下属的blob/tree索引，以SHA-1值为文件名

commit object: 记录对应的tree和父commit，以SHA-1值为文件名





HEAD指针，指向当前的本地分支

**分支的创建与延伸**



**分支的合并**

1. 前进式merge hotfix into master



2. 最近公共祖先法merge iss53 into master





a) 新建的commit，指向C4和C5，并移动被合入的分支指针

b) 合并冲突: 同一个文件的同一个部分进行了不同的修改

Usage: https://git-scm.com/docs



-------------------------------------------------------------------

远程仓库：

    clone 远程克隆至本地

fetch 抓取变更，但不更新工作目录

    pull 远程拉取最新内容至本地(fetch远程变更 + merge本地分支)

    push 本地修改内容推送至远程

pull request 拉取请求

本地仓库：

    checkout 切换分支

    merge 图式合并分支(non-destructive but pollute history)

    rebase 线性合并分支(cleaner history but rewrite)

    commit 提交变更

revert 相反方向的提交，实现撤销，保留提交记录(安全)

reset 回退到指定版本，并删除后续提交记录(危险)

[This usage of git reset is a simple way to undo changes that haven' t been shared with anyone else.]

log 提交历史记录

status 检查staged snapshot状态

    diff 差异比对

目录结构

.gitignore 忽略的文件列表

/.git/objects/

blob object: 文件名是SHA-1值，只保存文件内容。

持续集成Continuous Integration：频繁地将代码合并到主干。

持续交付Continuous Delivery：频繁地将软件的新版本，交付给质量团队或者用户，以供评审。通过后，手动部署。

持续部署Continuous Deployment：持续交付的基础上，自动部署到生产环境。