**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

4. HEAD 上一次提交的快照，下一次提交的父结点

Index 预期的下一次提交的快照

Working Directory 沙盒

5. 当三个区保持一致时，git status显示无改动。

Sharing and Updating Projects:

fetch下载远程分支的引用, pull(fetch+merge), push上传本地分支的引用和object

**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: 文件的快照压缩，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:** 非常不鼓励在使用git merge时存在未commit的文件

1. 前进式merge hotfix into master(不推荐)



2. 最近公共祖先法merge iss53 into master





a) 新增一个差异整合的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

c) merge: 保持有向无环图的结构，并扩展。

3. 变基rebase master server



a) 将源commit在目标分支上重新播放一次

b) Do not rebase commits that exist outside your repository!

如果rebase远程commit，可能会把别人的父commit删除！

c) rebase: 破坏有向无环图的结构，并线性化。

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



**目录结构**

.gitignore 忽略的文件列表

/.git/objects/ 每个文件的压缩对象

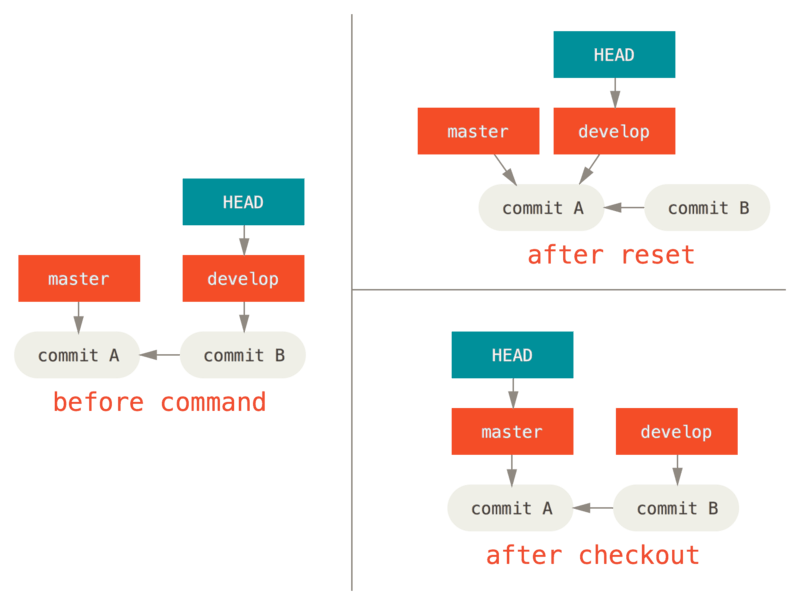
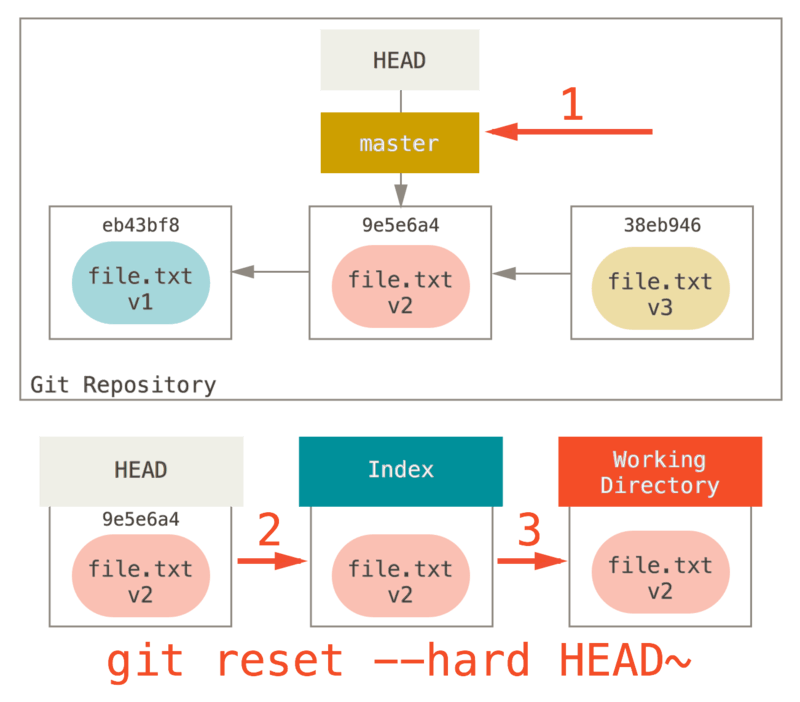
/.git/refs/heads/ 本地分支的引用

/.git/refs/remotes/ 远程分支的引用

index 暂存区

**回退(对已跟踪的文件替换文件快照)**

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

****

checkout: move HEAD -> replace index -> replace working directory

reset: move HEAD/master -> replace index -> replace working directory

revert: add index -> commit repository

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。