# 1理解版本管理软件

## 1.1 SVN基本原理

SVN基本原理：是一个跨平台的版本管理系统，类似一个文件服务器，对每个文件的每一次修改更新进行记录，通过这些记录可以实现回溯到任意时刻的旧版本，从而对程序源码、各种类型文件（文本、视频、图片等）进行管理。

SVN工作流程：

（1）在中央版本库中创建一个项目，项目中包含主干和分支，分支是从主干复制的；

（2）开发人员从中央版本库check out 下这个分支的代码；

（3）增加自己的代码、修改或删除现存的代码；

（4）commit 代码，如果修改期间其他人提交了代码，就会提示过期无法提交，就需要先up，之后再提交；如果up代码时出现冲突，就需要解决冲突之后在提交（开发人员协商解决冲突）

（5）测试环境测试，解决问题之后合并到主干中，在之后更新生产环境代码。

## 1.2 SVN基本原理

Git基本原理：Git也是一款版本管理器，不同于SVN的是它是分布式的，不仅有一个中心的服务器控制最新版本代码，而且每个开发者自己还有个本地仓库，所以在开发过程中都是先将代码提交到本地仓库再推送到中心服务器上的，这样的好处就是每个人都依赖于中心服务器来实现交互，但又不会被中心服务器限制，就算中心服务器挂了，也能很容易的找到最新版本的代码，而且自己的工作依然可以顺利进行，提交到本地仓库，当中心服务器修复之后，再将自己仓库的东西推送到中心服务器。

Git工作流程：

（1）本地创建一个Git库，并将其add到远程Git库中；

（2）在本地添加、删除、修改文件，然后commit，此时commit 是提交到本地Git仓库中；

（3）将本地Git库分支push到远程Git库的分支，如果此时远程已经有别人push过，那远程Git将不允许你push，必须先将远程仓库pull到本地，并解决冲突；

（4）解决冲突之后，重新commit到本地Git仓库，再push到远程Git仓库。

## 1.3 SVN与Git区别分析

（1）管理方式：GIT是分布式的，SVN不是，这是GIT和其它非分布式的版本控制系统，最核心的区别，如图1和图2所示为两种版本控制系统示意图：

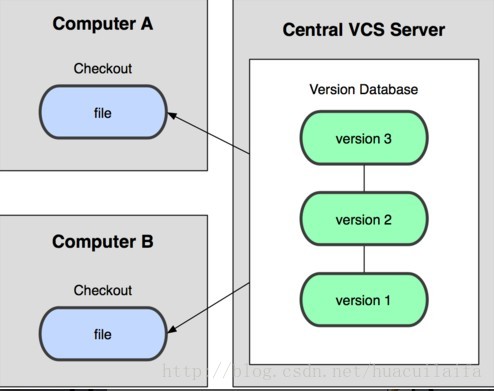


图1 集中化版本控制系统

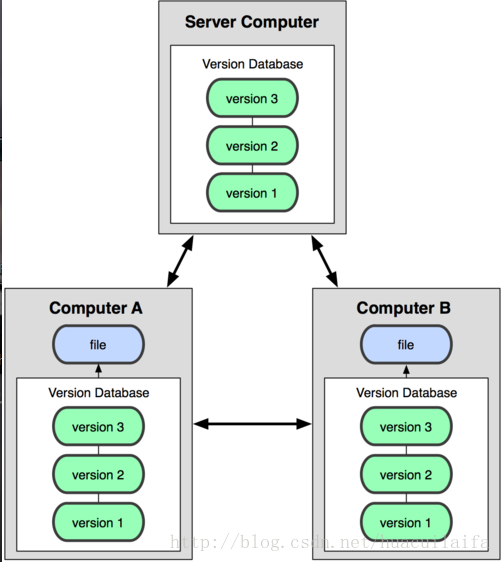


图2 分布式版本控制系统

（2）文件记录：SVN关心文件内容的具体差异，记录原始文件存储，占用体积较大，而Git直接记录快照，只关心文件数据的整体是否发生变化并不保存这些前后变化的差异数据，每次提交更新时，Git会纵览一遍所有文件的指纹信息并对文件作一快照，然后保存一个指向这次快照的索引。为提高性能，若文件没有变化，Git 不会再次保存，而只对上次保存的快照作一链接。如图3和图4所示：

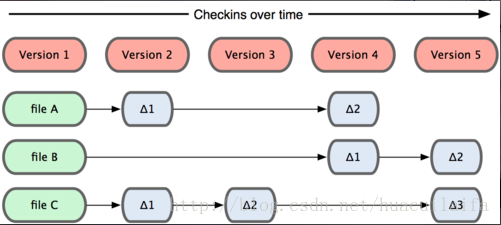


图3 SVN记录文件差异

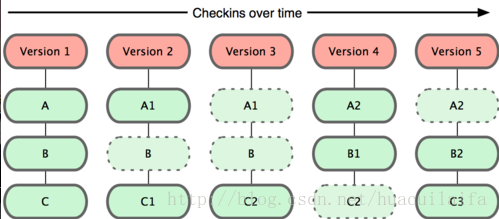


图4 Git记录文件快照

（3）网络依赖：SVN必须要联网才能工作，而且对网络的依赖性较强，如果推送的文件比较大而且网络状况欠佳，则提交文件的速度会受到很大的限制。Git在没有网络的情况下也可以执行commit、查看版本提交记录、以及分支操作，在有网络的情况下执行 push 到远程。

（4）分支影响：SVN单个用户创建新的分支则所有的用户都会拥有一样的分支。Git分支操作不会影响其他开发人员。

## 1.4 应用场景

SVN适用于开发人员对于版本管理的水平参差不齐的时候，或者对于代码的安全性要求更高一点的时候：一个研发队伍的成员正常包括：需求分析、设计、美工、程序员、测试、实施、运维，每个成员在工作中都有产出物， 包括了文档、设计代码、程序代码，这些都需要按项目集中进行管理的。SVN能清楚的按目录进行分类管理， 使项目组的管理处于有序高效的状态。

Git适用于多人参与开发，代码量比较大，或者高频次协作，跨公司，跨地域合作的情况下，而且有无网络均可工作：

（1）无远程：在本地 .git 文件夹内维护历时文件；

（2）有远程：在本地 .git 文件夹内维护历时文件，同时也将历时文件托管在远程仓库。

## 1.5应用意义

使用意义：通过版本控制解决了（1）代码管理混乱；（2）解决代码冲突困难；（3）在代码整合期间引发bug；（4）无法对代码的拥有者进行权限控制（4）项目不同版本的发布困难。