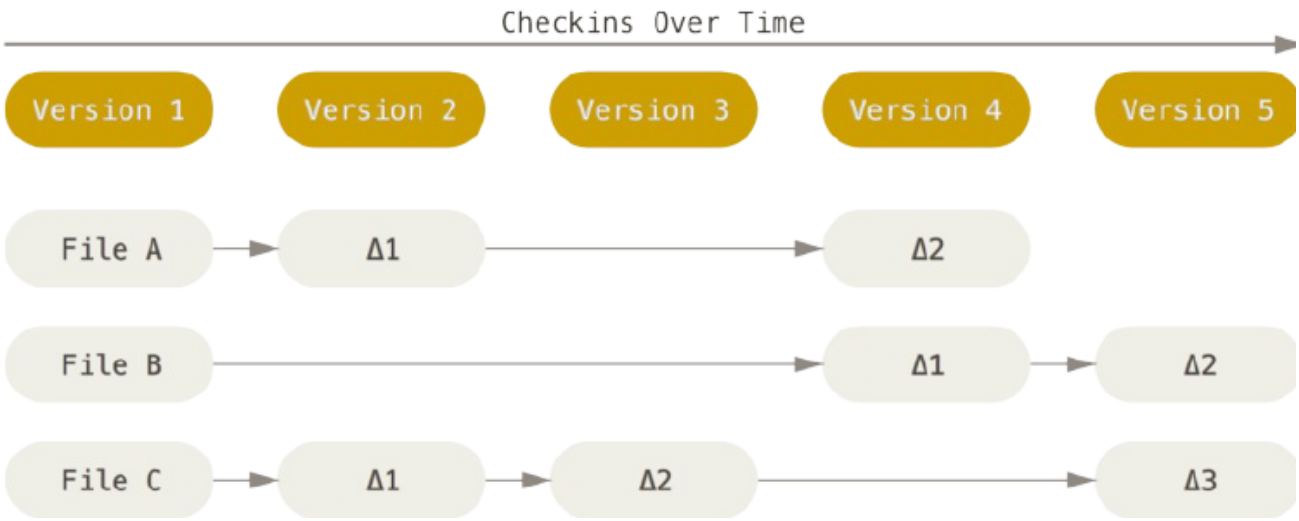
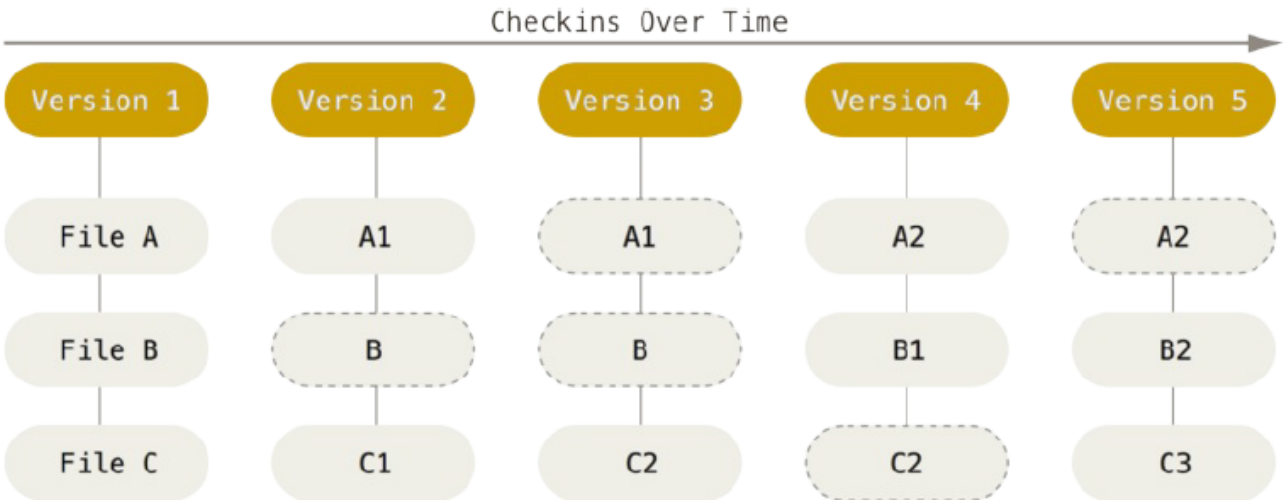
Git学习记录

一、概述

Git是分布式的版本控制系统，客户端会把代码仓库完整地镜像下来。每一个clone都是对代码仓库的完整备份。Git和其他版本控制的主要差别在于Git对待数据的方式，其他大部分系统以文件变更列表的方式储存信息，将保存的信息看作基本文件和每个文件随时间逐步积累的差异。



而Git把数据看作小型文件系统的一组快照并保存这个快照的索引，如果没有修改，Git只保留一个链接指向之前存储的文件。（所谓的增量式开发） 在本地磁盘上你就拥有项目的完整历史，使得操作非常迅速。Git 中所有数据在存储前都计算校验和，然后以校验和来引用。 这意味着不可能在 Git 不知情时更改任何文件内容或目录内容。



二、Git分支

Git中的“杀手级特性”，正是因为这一特性，使得Git从众多版本控制系统中脱颖而出。与许多其它版本控制系统不同，Git 鼓励在工作流程中频繁地使用分支与合并。

相对使用仅有的一个`master`分支，`Gitflow`工作流使用2个分支来记录项目的历史。`master`分支存储了正式发布的历史，而`develop`分支作为功能的集成分支。这样也方便`master`分支上的所有提交分配一个版本号。

开发新功能时，从develop分支中fork一个新的分支，当开发完成后合并回develop分支。

一旦`develop`分支上有了做一次发布（或者说快到了既定的发布日）的足够功能，就从`develop`分支上`fork`一个发布分支。新建的分支用于开始发布循环，所以从这个时间点开始之后新的功能不能再加到这个分支上 —— 这个分支只应该做`Bug`修复、文档生成和其它面向发布任务。一旦对外发布的工作都完成了，发布分支合并到`master`分支并分配一个版本号打好`Tag`。

维护分支或说是热修复（`hotfix`）分支用于生成快速给产品发布版本（`production releases`）打补丁，这是唯一可以直接从`master`分支`fork`出来的分支。修复完成，修改应该马上合并回`master`分支和`develop`分支（当前的发布分支），`master`分支应该用新的版本号打好`Tag`。

从Git的工作流和merge时的特性来看，在开发时需要注意:

1. 开发时分工明确，每个人负责的文件要区分清楚，否则merge时会出错误。
2. 命名规范要确定清楚，在同一个生存周期里有命名冲突会很难处理。