# [Git简介以及与SVN的区别](http://blog.csdn.net/fengbingchun/article/details/42215091)

Git是由著名Linux内核(Kernel)开发者LinusTorvalds为了便利维护Linux而开发的。

Git是一个分布式的版本控制系统。作为一个分布式的版本控制系统，在Git中并不存在主库这样的概念，每一份复制出的库都可以独立使用，任何两个库之间的不一致之处都可以进行合并。

在windows下使用git，如果对命令不熟悉，可以使用tortoisegit. tortoisegit依赖于msysgit.

在git中，每个本地的代码库都是一个完整的git系统。使用tortoisegit提交代码(commit)，其实只是提交到本地的git版本库。更新本地版本库后，就可以将其推送(push)到远程版本库，合并到master或者其它分支。

Github是一个网站，同时也是一个项目仓库，能让多名开发人员很方便地进行协同软件开发。

**在Windows下安装Git步骤**：

1．  从 <http://download.tortoisegit.org/tgit/1.8.12.0/>下载最新的TortoiseGit-1.8.12.0-64bit.msi;

2．  从 <http://msysgit.github.io/> 下载Git-1.9.4-preview20140929.exe;

3．  先安装Git-1.9.4-preview20140929.exe，然后再安装TortoiseGit-1.8.12.0-64bit.msi，按默认设置即可，重启电脑；

4．  鼠标右键-->TortoiseGit-->Settings-->General-->Version，点击Check now，如果显示gitversion 1.9.4.msysgit.2说明配置成功；

5．  如果从网上下载相关代码，选择Git clone, 弹出Git clone对话框，在URL中输入地址即可，如果需要用户名和密码，还会弹出输入用户名和密码的对话框，然后点击OK，即可将源码下载到本地。

**tortoisegit的一些常见用法**：

1.      Git Clone… :克隆一个项目代码库到本地(对应于svn的checkout)；

2.      Git Create repository here… :git在这里创建新版本，建立git项目(Init)；

3.      Git Sync… : 用于同步两个版本库；

4.      TortoiseGit-->Pull : 将远程版本库拉到本地版本库；

5.      TortoiseGit-->Push: 将本地版本库推送到远程版本库；

6.      Git Commit->”master”… :提交到本地版本库，成功后弹出对话框可以推送到远程版本库。

**Git和SVN之间的区别**：

1.      Git是分布式的，SVN不是，SVN是集中式：分布式模式，就是每个开发人员从中心版本库/服务器上check out代码后会在自己的机器上克隆一个自己的版本库。在一个不能连接网络的地方时，仍然能够提交文件，查看历史版本记录，创建项目分支等。集中式的版本控制系统都有一个单一的集中管理的服务器，保存所有文件的修订版本，而协同工作的人员都通过客户端连到这台服务器，取出最新的文件或者提交更新。SVN特点：(1).每个版本库有唯一的URL，每个用户都从这个地址获取代码和数据；(2).获取代码的更新，也只能连接到这个唯一的版本库，同步以取得最新数据；(3).提交必须有网络连接(非本地版本库)；(4).提交需要授权，如果没有写权限，提交会失败；(5).提交并非每次都能成功。与SVN不同，Git记录版本历史只关心文件数据的整体是否发生变化。Git并不保存文件内容前后变化的差异数据。在分布式版本控制系统中，客户端并不只是提取最新版本的文件快照，而是把原始的代码仓库完整地镜像下来。因为Git在本地磁盘上就保存着所有有关当前项目的历史更新，并且Git中的绝大多数操作都只需要访问本地文件和资源，不用连网，所以处理起来速度飞快。Git特点：(1).Git中每个克隆(clone)的版本库都是平等的。你可以从任何一个版本库的克隆来创建属于你自己的版本库，同时你的版本库也可以作为源提供给他人；(2).Git的每一次提取操作，实际上都是一次对代码仓库的完整备份；(3).提交完全在本地完成，无须别人给你授权，你的版本库你做主，并且提交总是会成功；(4).甚至基于旧版本的改动也可以成功提交，提交会基于旧的版本创建一个新的分支；(5).Git的提交不会被打断，合并会发生在PULL和PUSH过程中，不能自动解决的冲突会提示你手工完成；(6).Git也可以模拟集中式的工作模式。

2.      SVN的工作区和版本库是截然分开的，而Git的工作区和版本库是如影随形的：SVN的版本库和工作区是存储在不同的路径下，一般是在不同的主机中。SVN的企业级部署中，版本库在服务器上，只能通过https,http,svn等协议访问，而不能直接被用户接触到。SVN的工作区是一份版本库在某个历史状态下的快照。Git的版本库和工作区在同一个目录下，工作区的根目录有一个.git的子目录。这个名为.git的目录就是版本库本身，它是Git用来保存元数据和对象数据库的地方。该目录非常重要，每次克隆镜像仓库的时候，实际拷贝的就是这个目录里面的数据。Git版本库可以脱离工作区而存在，但是工作区不能脱离版本库而存在，即工作区的根目录下必须有一个名为.git的版本库克隆文件。

3.      Git把内容按元数据方式存储，而svn是按文件：.git目录是处于你的机器上的一个克隆版的版本库，它拥有中心版本库上的所有的东西，例如标签、分支、版本记录等。

4.      版本号：Git没有一个全局的版本号，而SVN有。SVN每一次提交都具有整个版本库全局唯一的版本号。Git的版本号则更进一步，版本号是全球唯一的。SVN的版本号是连续的版本号，每一次新的提交都会版本号+1. Git对于每一次提交，通过对文件的内容或目录的结构计算出一个SHA-1哈希值，得到一个40位的十六进制字符串，Git将此字符串作为版本号。所有保存在Git数据库中数据都是用此哈希值作索引的，而不是靠文件名。使用哈希值作版本号的好处就是对于一个分布式的版本控制系统,每个人每次提交后形成的版本号都不会出现重复.另一好处是保证数据的完整性,因为哈希值是根据内容或目录结构计算出来的,所以还可以据此来判断数据内容是否被篡改.Git的版本号虽然不连续，但是是有线索的，即每一个版本都有对应的父版本(一个或者两个)，进而可以形成一个复杂的提交链。

5.      Git的内容完整性要优于SVN：Git的内容存储使用的是SHA-1哈希算法，这能确保代码内容的完整性，确保在遇到磁盘故障和网络问题时降低对版本库的破坏。

6.      版本库(repository)：SVN只能有一个指定中央版本库，当这个中央版本库有问题时，所有工作成员都一起瘫痪直到版本库维修完毕或者新的版本库设立完成。而Git可以有无限个版本库，或者每一个Git都是一个版本库，区别是它们是否拥有活动目录。如果主要版本库发生了什么事，工作成员仍然可以在自己的本地版本库提交，等待主要版本库恢复即可。工作成员也可以提交到其它的版本库。

7.      重新设立起点(rebase)：在Git，如果你想把别人的最新提交设立为现在这个分支的起点，只要执行git rebase branch\_name即可。这个和合并(merge)不同的是，merge会依据修改的时间视为最新，而rebase会要求你去解决双方都有修改过的地方的矛盾(confict).

8.      系统档案：SVN会在每一个目录放一个.svn，如果想移除这些.svn是很累的(最新版本的svn只有在根目录有.svn，子目录没有了)。而Git会在目录起点拥有一个.git目录。

9.      Git是压缩后传输，svn是一个一个文件传输，所以git的网络流量比svn少很多。

10.  部分检出(checkout)：SVN可以将整个库检出到工作区，也可以将某个目录检出到工作区。但是Git只能全部检出，不支持按照目录进行的部分检出。在SVN中，从仓库checkout的一个工作树，每个子目录下都维护着自己的.svn目录，记录着该目录中文件的修改情况以及和服务器端仓库的对应关系。Git没有部分检出，这并不是说只有将整个库克隆下来才能查看文件。

11.  更新：在SVN中,因为只有一个中心仓库,所以所谓的远程更新,也就是svn update，通过此命令来使工作区和版本库保持同步。Git使用git fetch和git pull来完成远程更新任务。fetch操作只是将远程数据库的object拷贝到本地，然后更新remotes head的refs，git pull的操作则是在git fetch的基础上对当前分支外加merge操作。

12.  提交(commit)：在SVN，当你提交你的代码时，它将直接记录到中央版本库。当你发现你的代码存在严重问题时，你已经无法阻止事情的发生。如果网络中断，你根本没办法提交。对于SVN来说，所有的commit操作都可以认为是对远程仓库的更新动作。在工作区中对文件进行添加(add)、修改、删除(delete)操作要同步到版本库，必须使用commit命令。而Git的提交完全属于本地版本库的活动。而你只需”推”(git push)到主要版本库即可。Git的”推”其实是在执行”同步”(Sync)。在Git中，要将一个文件纳入版本管理的范畴，首先是要用git add将文件纳入stage(暂存区域，介于workcopy和版本库head版本的一种中间状态)的监控范围，只有更新到stage中的内容才会在commit的时候被提交。Git的stage让你在提交的时候清楚的知道git将要提交哪些改动。

13.  分支(branch)：在SVN，分支是一个完整的目录，且这个目录拥有完整的实际文件。而Git，每个工作成员可以任意在自己的本地版本库开启无线个分支。Git的分支相当简单，你可以从同一个工作目录下快速的在几个分支间切换，很容易发现未被合并的分支，能简单而快捷的合并这些文件。Git中的分支实际上仅是一个包含所指对象校验和(40个字符长度SHA-1哈希值)的文件，新建一个分支就是向一个文件写入41个字节(版本号加一个换行符)，自然速度很快。Git的实现与项目负责度无关，它永远可以在几毫秒的时间内完成分支的创建和切换。Git的分支是完全隔离的，而SVN则没有。Git的里程碑是只读的，Git完全遵守历史不可更改这一时空法则。用户不能向git的里程碑中提交，否则里程碑就不是标记，而成了一个分支。SVN中提供了一个功能switch，使用switch可以在同一个工作树上，在不同的分支中进行切换。Git在分支中进行切换使用的命令是checkout.

14.  优缺点：SVN优点：(1)、管理方便、逻辑明确，符号一般人思维习惯；(2)、易于管理，集中式服务器更能保证安全性；(3)、代码一致性非常高；(4)、适合开发人数不多的项目开发。SVN缺点：(1)、服务器压力太大，数据库容量暴增；(2)、如果不能连接到服务器上，基本上不可以工作，就不能提交、还原、对比等等；(3)、不适合开源开发，但是一般集中式管理的有非常明确的权限管理机制，可以实现分层管理，从而很好的解决开发人数众多的问题。Git优点：(1)、适合分布式开发，强调个体；(2)、公共服务器压力和数据量都不会太大；(3)、速度快、灵活；(4)、任意两个开发者之间可以很容易的解决冲突；(5)、离线工作。Git缺点：(1)、学习周期相对而言比较长；(2)、不符合常规思维；(3)、代码保密性差，一旦开发者把整个库克隆下来就可以完全公开所有代码和版本信息。

参考文献：

1.      <http://www.vaikan.com/5-fundamental-differences-between-git-svn/>

2.      [http://wenku.baidu.com/link?url=gc9leYhGnsMWhXz2r25yCnsKMdAAmEW5p7YH6ems\_1VPstXkbLe08qfc0EpT2vo71QAc\_-jBLxfDC55VGkKUJw-iuS6\_lJYS2X2yHXr1HXS](http://wenku.baidu.com/link?url=gc9leYhGnsMWhXz2r25yCnsKMdAAmEW5p7YH6ems_1VPstXkbLe08qfc0EpT2vo71QAc_-jBLxfDC55VGkKUJw-iuS6_lJYS2X2yHXr1HXS" \t "_blank)