Git 版本管理工具（一）

git工具subversion版本控制系统svn

Git 是一个分布式版本控制工具，它的作者 Linus Torvalds 是这样给我们介绍 Git —— The stupid content tracker（傻瓜式的内容跟踪器）

1、 Git 背景

Git 最初由 Linus Torvalds 编写，用于 Linux 内核开发的版本控制工具。

Git 与常用的版本控制工具 CVS、Subversion 等不同，它采用了分布式版本库的方式，不必服务器端软件支持，使源代码的发布和交流极其方便。

Git 的速度很快，这对于诸如 Linux kernel 这样的大项目来说自然很重要，Git 最为出色的是它的合并跟踪（merge tracing）能力。

实际上内核开发团队决定开始开发和使用 Git 来作为内核开发的版本控制系统的时候，世界开源社群的反对声音不少，最大的理由是 Git 太艰涩难懂，从 Git 的内部工作机制来说，的确是这样。但是随着开发的深入，Git 的正常使用都由一些友好的脚本命令来执行，使 Git 变得非常好用，即使是用来管理我们自己的开发项目，Git 都是一个友好、有力的工具。现在，越来越多的著名项目采用 Git 来管理项目开发，例如：wine、hiphop-php 等。

Git 作为开源自由原教旨主义项目，没有对版本库的浏览和修改做任何的权限限制，但通过其他工具也可以达到有限的权限控制，比如：gitosis、CodeBeamer MR。原本 Git 的使用范围只适用于 Linux / Unix 平台，但逐步并成熟了在 Windows 平台下的使用，主要归功于 Cygwin 与 msysgit 环境与 TortoiseGit 这样易用的 GUI 工具。其实 Git 的源代码中已经加入了对 Cygwin 与 MinGW 编译环境的支持并被逐步完善，对于 Windows 使用者是个福音。

2、 为什么选择 Git

流行的软件版本开源管理软件，有 CVS、SVN、GIT 版本管理工具，Git 的优势在哪里呢？

Git 入门教程，请查看我的百度空间博客：*Blog*

Git 和 CVS、SVN 不同，是一个分布式的源代码管理工具，它很强，也很快，Linux 内核的代码就是用 Git 管理的，它给我们带来的直接好处有：

1. 初始化，git init, git commit -a, 就完了。对于随便写两行代码就要放到代码管理工具里的人来说，再合适不过。也可以拿 git 做备份系统，或者同步两台机器的文档，都很方便。

2. 绝大部分操作在本地完成，不用和集中的代码管理服务器交互，终于可以随时随地大胆地 check in 代码了。 只有最终完成的版本才需要向一个中心的集中的代码管理服务器提交。

3. 每次提交都会对所有代码创建一个唯一的 commit id。不像 CVS 那样都是对单个文件分别进行版本的更改。所以你可以一次性将某次提交前的所有代码 check 出来，而不用考虑到底提交过那些文件。（其实 SVN 也可以做到这点）

4. branch 管理容易多了，无论是建立新的 branch，还是在 branch 之间切换都一条命令完成，不需要建立多余的目录。

5. branch 之间 merge 时，不仅代码会 merge 在一起，check in 历史也会保留，这点非常重要。

Git 的优势主要在

1、更方便的 Merge

分布式管理必然导致大量的 Branch 和 Merge 操作。因此分布式版本控制系统都特别注意这方面。在传统的 CVS 里面制作 Branch 和 Merge 简直就是噩梦，Subversion 作为一个用于替代 CVS 的系统，专门改进了 Branch 操作。然而似乎人们没有注意到，Branch 是轻松了，可是 Merge 呢？如果不能很方便地 Merge 回来，做 Branch 仍然是噩梦。事实上，我就经历过在开发团队里面由于队友操作不对而在 Merge 的时候把我的许多代码都覆盖掉了。当时正是使用的 subversion 。虽然源代码仍然在历史里面，但是要去一个一个地找出被覆盖掉的文件并恢复过来确实是一件很难忘的事情。

2、更方便的管理

传统的版本控制系统使用中央仓库，一些仓库相关的管理就只能在仓库上进行。赋予开发团队每一个人中央仓库的管理权限是非常不好的。但是有时候确实会比较不方便的地方。

3、更健壮的系统

分布式系统一般情况下总是比单服务端的系统要健壮，因为当服务端一旦挂掉了整个系统就不能运行了。然而分布式系统通常不会因为一两个节点而受到影响。

4、对网络的依赖性更低

虽然现在网络非常普及，但是并不是随时随地都有高速网络，甚至有时候根本没有网络可以访问。低速的网络会让人心情烦躁，有时候就呆呆地盯着屏幕上的 commit 进度，什么事情也干不了。而没有网络连接更是致命的：你无法 commit ！这表示你进行任何改动以前都必须小心翼翼，否则你可能再也找不会你曾经写的一些代码了。

5、更少的“仓库污染”

有时候你要做一个模块，它不是太大，所以没有必要为它新建一个 branch ，但是它又不是那么小，不可能一次提交就做好。于是便会提交一些不完整的代码到仓库，有时候会导致整个程序无法运行，严重影响团队里其他人的开发。大多数人在这种情况下的解决办法都是写完之后再提交。但是作为习惯了版本控制的人来说，进行不计后果的大幅修改是经常的事情，到后来突然发现自己先前的代码没有提交，就后悔莫及了。如果是分布式系统的话就不会存在这样的问题，因为本地仓库的修改不会影响到别人的仓库。当你完成并测试以后，就可以在邮件列表里面说：我已经把这个模块做好了。然后感兴趣的人就可以从你这里 pull 你的成果了。

虽然网上各种对 Git 的誉美之词决不止于此，但是在 Git 的主站上，还是尽可能客观的对 Git 和 Subversion 进行了一番比较（GitSvnComparsion）。另外，Subversion 目前通过 SVK 也已经提供了一定程度上的源代码库分布式的管理能力，能够实现源代码的离线提交等功能。

3、 Git 、CVS、SVN 比较

项目源代码的版本管理工具中，比较常用的主要有：CVS、SVN、Git 和 Mercurial （其中，关于 SVN，请参见我先前的博客：SVN 常用命令 和 SVN 服务器配置）

目前 Google Code 支持 SVN、Git、Mercurial 三种方式，例如：我上传的 linux-kernel-source（Git 方式）、sdk-java（SVN 方式），那么它们各有什么区别呢？

Git 与 CVS 的区别

- **分支更快、更容易。**
- **支持离线工作；本地提交可以稍后提交到服务器上。**
- **Git 提交都是原子的，且是整个项目范围的，而不像 CVS 中一样是对每个文件的。**

- **Git 中的每个工作树都包含一个具有完整项目历史的仓库。**
- **没有哪一个 Git 仓库会天生比其他仓库更重要。**

Git 与 SVN 的区别

Git 不仅仅是个版本控制系统，它也是个内容管理系统(CMS)、工作管理系统等。如果你曾是一个使用过 SVN 背景的人，那么你可以很容易的做一定的思想转换，来适应 Git 提供的一些概念和特征。这篇文章的主要目的就是通过介绍 Git 能做什么，以及它和 SVN 在深层次上究竟有什么不同，通过比较来帮助你更好的认识 Git

1. **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（SVN，CVS）最核心的区别。如果你能理解这个概念，那么你就已经上手一半了。需要做一点声明，Git 并不是目前第一个或唯一的分布式版本控制系统。还有一些系统如 Bitkeeper, Mercurial 等也是运行在分布式模式上的，但 Git 在这方面做的更好，而且有更多强大的功能特征。

   Git 跟 SVN 一样有自己的集中式版本库或服务器。但 Git 更倾向于被使用于分布式模式，也就是每个开发人员从中心版本库的服务器上 chect out 代码后会在自己的机器上克隆一个自己的版本库。可以这样说，如果你被困在一个不能连接网络的地方时，就像在飞机上，地下室，电梯里等，你仍然能够提交文件，查看历史版本记录，创建项目分支等。对一些人来说，这好像没多大用处，但当你突然遇到没有网络的环境时，这个将解决你的大麻烦。

   同样，这种分布式的操作模式对于开源软件社区的开发来说也是个巨大的恩赐，你不必再像以前那样做出补丁包，通过 email 方式发送出去，你只需要创建一个分支，向项目团队发送一个推请求。这能让你的代码保持最新，而且不会在传输过程中丢失，一个这样的优秀案例就是： GitHub.com

   有些谣言传出来说 subversion 将来的版本也会基于分布式模式。但至少目前还看不出来。

2. **Git 把内容按元数据方式存储，而SVN是按文件**

   所有的资源控制系统都是把文件的元信息隐藏在一个类似.svn、.cvs 等的文件夹里。如果你把 .git 目录的体积大小跟.svn 比较，你会发现它们差距很大。因为 .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所有的东西，例如标签、分支、版本记录等。

3. **Git 分支和SVN的分支不同**

   分支在 SVN 中一点不特别，就是版本库中的另外的一个目录。如果你想知道是否合并了一个分支，你需要手工运行像这样的命令 *svn propget svn:mergeinfo*，来确认代码是否被合并。所以，经常会发生有些分支被遗漏的情况。

   然而，处理 Git 的分支却是相当的简单和有趣，你可以从同一个工作目录下快速的在几个分支间切换。你很容易发现未被合并的分支，你能简单而快捷的合并这些文件。

4. **Git 没有一个全局的版本号，而SVN有**

目前为止这是跟 SVN 相比 GIT 缺少的最大的一个特征。你也知道，SVN 的版本号实际是任何一个相应时间的源代码快照，它是从 CVS 进化到 SVN 的最大的一个突破。Git 可以使用 SHA-1 来唯一的标识一个代码快照，但这个并不能完全的代替 SVN 里容易阅读的数字版本号。

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。这有一个很好的关于 Git 内容完整性的讨论。（英文原文参考：diff）

CVS-SVN-GIT 综合比较

首先，介绍几个版本控制软件相互比较的重要依据：

（1）版本库模型（Repository model）：描述了多个源码版本库副本间的关系，有客户端/服务器和分布式两种模式。在客户端/服务器模式下，每一用户通过客户端访问位于服务器的主版本库，每一客户机只需保存它所关注的文件副本，对当前工作副本（working copy）的更改只有在提交到服务器之后，其它用户才能看到对应文件的修改。而在分布式模式下，这些源码版本库副本间是对等的实体，用户的机器出了保存他们的工作副本外，还拥有本地版本库的历史信息。

（2）并发模式（Concurrency model）：描述了当同时对同一工作副本/文件进行更改或编辑时，如何管理这种冲突以避免产生无意义的数据，有排它锁和合并模式。在排它锁模式下，只有发出请求并获得当前文件排它锁的用户才能对对该文件进行更改。而在合并模式下，用户可以随意编辑或更改文件，但可能随时会被通知存在冲突（两个或多个用户同时编辑同一文件），于是版本控制工具或用户需要合并更改以解决这种冲突。因此，几乎所有的分布式版本控制软件采用合并方式解决并发冲突。

（3）历史模式（History model）：描述了如何在版本库中存贮文件的更改信息，有快照和改变集两种模式。在快照模式下，版本库会分别存储更改发生前后的工作副本；而在改变集模式下，版本库除了保存更改发生前的工作副本外，只保存更改发生后的改变信息。

（4）变更范围（Scope of change）：描述了版本编号是针对单个文件还是整个目录树。

（5）网络协议（Network protocols）：描述了多个版本库间进行同步时采用的网络协议。

（6）原子提交性（Atomic commit）：描述了在提交更改时，能否保证所有更改要么全部提交或合并，要么不会发生任何改变。

（7）部分克隆（Partial checkout/clone）：是否支持只拷贝版本库中特定的子目录。

称	版本库模型	并发模式	历史模式	变更范围	
VS	Client-server	Merge	Changeset	File	Pserver,ssh
/N	Client-server	3-way merge, recursive merge, octopus merge	Changeset and Snapshot	Tree	custom (sv HTTP and
it	Distributed	Merge or lock	Snapshot	Tree	custom, cu HTTP/HT

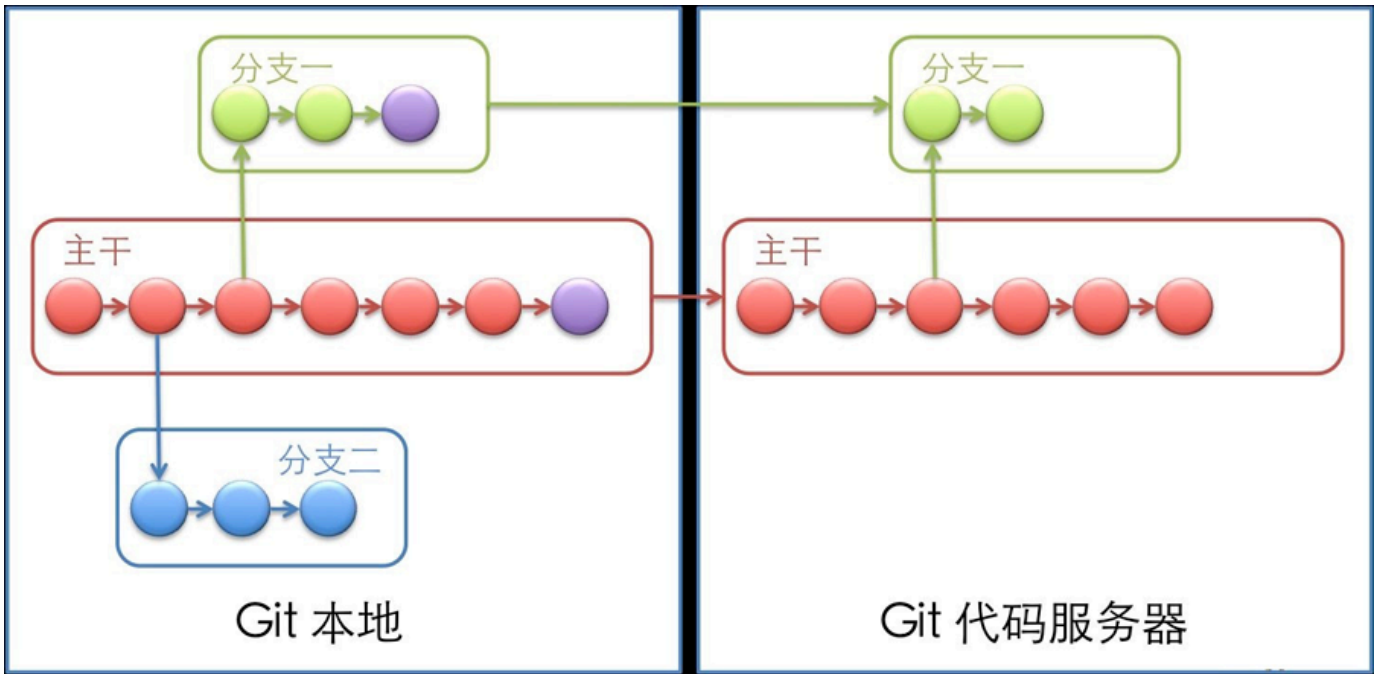
Trunk、Branches、Tags 区别：

Trunk：软件开发过程中的主线，开发时版本存放的目录，即在开发阶段的代码都提交到该目录上，保存了从版本库建立到当前的信息。

Branches：软件开发过程中的分支，发布版本存放的目录，即项目上线时发布的稳定版本存放在该目录中，保存了从版本库的某一特定点（不一定是版本库建立时）到当前的信息。

tags：表示标签存放的目录，tags 只可读，不可写

分支主要用于在不影响 Trunk 其它用户情况下进行一些关于新功能的探索性或实验性的开发，待新功能完善后它也可以合并到 Trunk 中。



（原文，请参考我在百度空间的博客：Git 命令参数及用法详解）

4、 Git 在 Windows 上的使用

Git 是为 Linux 而生的，其最初创建人就是 Linux 的创始人—— Linus Torvalds

Linux 环境下，使用 Git 与任何 Linux 中的命令行工具没有什么区别，甚至在击键数上还有明显的优势。

Windows 环境下，使用 Git 在目前看来只有两种方法：

1、使用 **Cygwin**（一个在 Windows 上运行的 Linux 环境）

2、使用 **msysgit**（Windows 下提供图形界面和命令行）

Cygwin 和 msysgit 的使用方法类似，Cygwin 具有大量 Linux 的功能，如果只是想使用 Git 功能，msysgit 还是最简单和快速的方法。

为了能够具备通过互联网实现与别人协作开发的能力，对于项目需要一个公开的源代码托管服务。好在，现在已经有不少可以供我们选择的，尤其是 **githost**，更是一个中文的源代码托管服务提供方。从目前看来，在 Githost 上落户的项目还很少，貌似是一个新近诞生的服务提供方。如果项目对服务提供的稳定性有比较高的要求的话，还是选择较老的 git 源代码托管服务比较好吧。如果是在局域网内工作的小组，要使用 Git 做源代码管理，那就更简单了，大家安装好自己的 Git，并指定一个人负责对 Git 版本库进行管理就好了。

（1） GitHub 简介

GitHub 是使用 Ruby 开发的，具有清爽的界面。

GitHub 提供免费的源代码库托管，同时也提供付费的托管服务。通过付费私有库托管服务在财务上支持免费部分的持续运营。

GitHub 提供了一套独特的代码库管理界面功能，并提供项目 Wiki 的能力。

GitHub 提供了一系列的指南，官方网址：GitHub

（2）Windows 系统上安装 Git

首先，下载并安装 msysgit 程序：download

接着，安装下载的 Git-1.7.10-preview20120409.exe，可以选择最新的 Git 版本，以取得更好的使用效果。

安装的过程很简单，基本上可以使用默认设置。只是在设置路径的时候要注意一下，为了避免与 Windows 路径导致的意外情况，还是使用“Use Git Bash Only”比较安全。

Msysgit 有命令行和图形 UI 两种使用方式，根据你的喜好选择吧，要说的是，图形 UI 可能不能完成所有的工作，因此在某些情况下（例如创建 SSH Key），命令行还是必不可少的。Msysgit 的 Bash 命令行对中文的支持不好，所有的中文字符都显示成了“？”。因此，为了避免麻烦，最好避免使用中文的文件名、目录名和用户名等

Msysgit 图形界面如下：

（原文，请参考我在百度空间的博客：Git 界面 GUI 和命令行 Command 两种操作方式）

（3） 设定 GitHub

要使用 GitHub，首先需要创建 SSH Key，SSH 将用来加密本机与远端服务器之间的通信，同时也是识别你对代码所做的变更的方法。

SSH Key 可以使用 Git 命令行来产生，如果你已经有一个 SSH Key，那么在这里也可以直接使用。

要使用 Git 创建 SSH Key 首先需要打开 Git Bash 命令行，输入命令：

```
ssh-keygen  -C  "username@email.com"  -t  rsa
```

说明：username@email.com 需要更换成你自己的 email 地址

程序将提出一些问题，接受文件默认存放位置，当要求输入 pass phrase 时，如果本机安全没有问题，也可以不输入。找到当时制定的文件存储位置中 id_rsa.pub 文件，这就是在 GitHub 上申请帐户时需要使用的 SSH 公钥文件。

在 github.com 的 register 中选择 Free account，在后续的界面中按照要求填入相应的内容即可完成注册，很简单的。

5、 Git 服务相关

1、 建立 Git 远程服务器：

目前貌似还没有在 Windows 上建立 Git 服务器的，足见 Linux 在开源社区里强大的优势啊！^_^

Hosting Git repositories, The Easy (and Secure) Way ：gitosis

2、 基于 Git 的源代码托管

Gitorious

Gitorious is another free hosting site with a custom web interface, supporting multiple repositories per project, local installations and with open source

repo.or.cz

repo.or.cz is the oldest hosting site, accommodating many hundreds of projects, with open-sourced infrastructure and aimed at open source software. It provides full push features as well as simple mirroring mode and gitweb interface with various enhancements.

GitHub

GitHub provides both free hosting for smaller projects and paid options for private hosting and large-sized projects. It uses a custom web interface including a wiki hosting and puts emphasis on social networking of project developers

3、 关于 Git 的有用的联结：

Git-scm：go url

Git Reference：go url

Git - SVN Crash Course：go url

Everyday GIT With 20 Commands Or So：go url

Git User's Manual (for version 1.5.3 or newer) ：go url

Getting Started with Git and GitHub on Windows：go url

注：本文原文，请见我在百度空间的博客 Windows 环境中使用版本管理工具 Git

参考推荐：

Git 命令参数及用法详解

Git 常用命令（图表）

SVN 常用命令

SVN 服务器安装

Git 详解之一：Git 起步（系列教程，推荐）