### Git 与 SVN 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。

如果你是一个具有使用 SVN 背景的人，你需要做一定的思想转换，来适应 Git 提供的一些概念和特征。

Git 与 SVN 区别点：

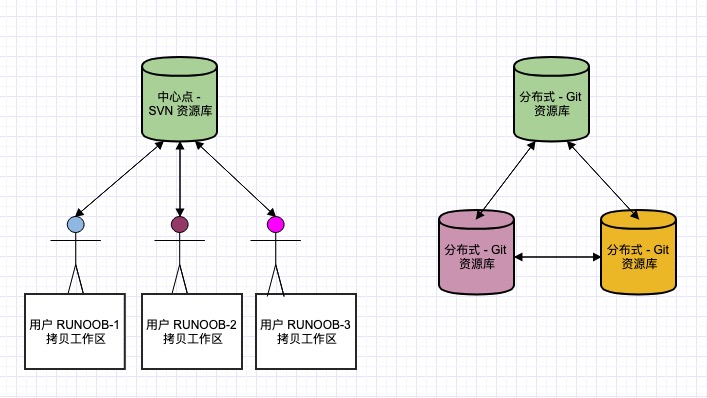
1、Git 是分布式的，SVN 不是：这是 Git 和其它非分布式的版本控制系统，例如 SVN，CVS 等，最核心的区别。

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。

3、Git 分支和 SVN 的分支不同：分支在 SVN 中一点都不特别，其实它就是版本库中的另外一个目录。

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。

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。



### Git仓库创建

工作区：就是你在电脑里能看到的目录。

暂存区：英文叫stage, 或index。一般存放在 ".git目录下" 下的index文件（.git/index）中，所以我们把暂存区有时也叫作索引（index）。

版本库：工作区有一个隐藏目录.git，这个不算工作区，而是Git的版本库。

一、git init

Git 使用 git init 命令来初始化一个 Git 仓库，Git 的很多命令都需要在 Git 的仓库中运行，所以 git init 是使用 Git 的第一个命令。

在执行完成 git init 命令后，Git 仓库会生成一个 .git 目录，该目录包含了资源的所有元数据，其他的项目目录保持不变（不像 SVN 会在每个子目录生成 .svn 目录，Git 只在仓库的根目录生成 .git 目录）

例：

git init newrepo

初始化后，会在 newrepo 目录下会出现一个名为 .git 的目录，所有 Git 需要的数据和资源都存放在这个目录中。

如果当前目录下有几个文件想要纳入版本控制，需要先用 git add 命令告诉 Git 开始对这些文件进行跟踪，然后提交：

$ git add \*.c

$ git add README

$ git commit -m '初始化项目版本'

以上命令将目录下以 .c 结尾及 README 文件提交到仓库中。

二、git clone

我们使用 git clone 从现有 Git 仓库中拷贝项目（类似 svn checkout）。

克隆仓库的命令格式为：git clone <repo> <directory>

例：

git clone 时，可以所用不同的协议，包括 ssh, git, https 等，其中最常用的是 ssh，因为速度较快，还可以配置公钥免输入密码。各种写法如下：

git clone git@github.com:fsliurujie/test.git --SSH协议

git clone git://github.com/fsliurujie/test.git --GIT协议

git clone https://github.com/fsliurujie/test.git --HTTPS协议

### GIT基本命令

Git 的工作就是创建和保存你的项目的快照及与之后的快照进行对比。本章将对有关创建与提交你的项目的快照的命令作介绍。

一、git add

git add 命令可将该文件添加到缓存，如我们添加以下两个文件：

$ git add README hello.php

二、git status

git status 以查看在你上次提交之后是否有修改。加了 -s 参数，以获得简短的结果输出。如果没加该参数会详细输出内容：

$ git status -s

A README

A hello.php

三、git diff

执行 git diff 来查看执行 git status 的结果的详细信息。

git diff 命令显示已写入缓存与已修改但尚未写入缓存的改动的区别。git diff 有两个主要的应用场景。

尚未缓存的改动：git diff

查看已缓存的改动： git diff --cached

查看已缓存的与未缓存的所有改动：git diff HEAD

显示摘要而非整个 diff：git diff --stat

在 hello.php 文件中输入以下内容：

$ git diff

diff --git a/hello.php b/hello.php

index e69de29..69b5711 100644

--- a/hello.php

+++ b/hello.php

@@ -0,0 +1,3 @@

+<?php

+echo '菜鸟教程：[www.runoob.com';](http://www.runoob.com';)

+?>

1. git commit

使用 git add 命令将想要快照的内容写入缓存区， 而执行 git commit 将缓存区内容添加到仓库中。

Git 为你的每一个提交都记录你的名字与电子邮箱地址，所以第一步需要配置用户名和邮箱地址。

$ git config --global user.name 'runoob'

$ git config --global user.email [test@runoob.com](mailto:test@runoob.com)

五、git reset HEAD

git reset HEAD 命令用于取消已缓存的内容。

git reset --hard “”（commit哈希值）

六、git rm

如果只是简单地从工作目录中手工删除文件，运行 git status 时就会在 Changes not staged for commit 的提示。

要从 Git 中移除某个文件，就必须要从已跟踪文件清单中移除，然后提交。可以用以下命令完成此项工作

例：

git rm <file>

如果删除之前修改过并且已经放到暂存区域的话，则必须要用强制删除选项 -f

git rm -f <file>

如果把文件从暂存区域移除，但仍然希望保留在当前工作目录中，换句话说，仅是从跟踪清单中删除，使用 --cached 选项即可

git rm --cached <file>

如我们删除 hello.php文件：

$ git rm hello.php

rm 'hello.php'

$ ls

README

不从工作区中删除文件：

$ git rm --cached README

rm 'README'

$ ls

README

可以递归删除，即如果后面跟的是一个目录做为参数，则会递归删除整个目录中的所有子目录和文件：

git rm –r \*

进入某个目录中，执行此语句，会删除该目录下的所有文件和子目录。

七、git mv

git mv 命令用于移动或重命名一个文件、目录、软连接。

我们先把刚移除的 README 添加回来：

$ git add README

然后对其重名:

$ git mv README README.md

$ ls

README.md

### Git 分支管理

创建分支命令：

git branch (branchname)

切换分支命令：

git checkout (branchname)

当你切换分支的时候，Git 会用该分支的最后提交的快照替换你的工作目录的内容， 所以多个分支不需要多个目录。也可以使用 git checkout -b (branchname) 命令来创建新分支并立即切换到该分支下，从而在该分支中操作。

合并分支命令:

git merge

合并冲突

合并并不仅仅是简单的文件添加、移除的操作，Git 也会合并修改。在 Git 中，我们可以用 git add 要告诉 Git 文件冲突已经解决，合并完后就可以删除分支:

git branch -d newtest

对其他分支的更改不会反映在主分支上。如果想将更改提交到主分支，则需切换回master分支，然后使用合并。

git checkout master

git merge test

### Git 查看提交历史

使用 git log 命令列出历史提交记录，用 --oneline 选项来查看历史记录的简洁的版本；还可以用 --graph 选项，查看历史中什么时候出现了分支、合并；用 --reverse 参数来逆向显示所有日志

如果只想查找指定用户的提交日志可以使用命令：git log --author

如果你要指定日期，可以执行几个选项：--since 和 --before，但是你也可以用 --until 和 --after。

### Git 远程仓库(Github)

如果你想通过 Git 分享你的代码或者与其他开发人员合作。 你就需要将数据放到一台其他开发人员能够连接的服务器上。

git clone <repo> [<dir>]

提取远程仓库

Git 有两个命令用来提取远程仓库的更新。

1、从远程仓库下载新分支与数据：git fetch

该命令执行完后需要执行git merge 远程分支到你所在的分支。

2、从远端仓库提取数据并尝试合并到当前分支：git merge

该命令就是在执行 git fetch 之后紧接着执行 git merge 远程分支到你所在的任意分支。

假设你配置好了一个远程仓库，并且你想要提取更新的数据，你可以首先执行 git fetch [alias] 告诉 Git 去获取它有你没有的数据，然后你可以执行 git merge [alias]/[branch] 以将服务器上的任何更新（假设有人这时候推送到服务器了）合并到你的当前分支。

# 提交到 Github

$ git remote add origin git@github.com:tianqixin/runoob-git-test.git

$ git push -u origin master

如果您已经按上面的进行push，下面命令表示，当前分支自动与唯一一个追踪分支进行合并。git pull

从非默认位置更新到指定的url。

git pull http://git.example.com/project.git