# git

## 什么是版本控制：

什么是版本控制？我为什么要关心它呢？版本控制是一种记录一个或若干文件内容变化，以便将来查阅特定版本修订情况的系统。在本书所展示的例子中，我们仅对保存着软件源代码的文本文件作版本控制管理，但实际上，你可以对任何类型的文件进行版本控制。

举例说明：编写word

## git介绍：

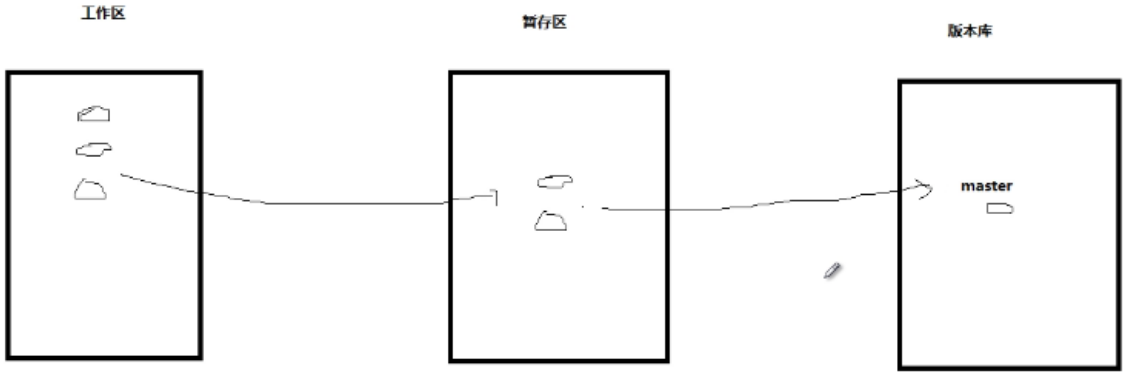
分布式版本系统的最大好处之一是在本地工作完全不需要考虑远程库的存在，也就是有没有联网都可以正常工作，而SVN在没有联网的时候是拒绝干活的！当有网络的时候，再把本地提交推送一下就完成了同步，真是太方便了！

git的三个区：

工作区：就是在你电脑里能够看到的目录

暂存区：作为过渡层，避免误操作，保护工作区和版本库，分支处理（例如，开发一半可以提交到暂存区）

版本库：工作区有一个隐藏的.git，这是是git的版本库；这里存放了很多东西，其中最重要的是称为暂存区，git还未我们自动创建了第一个分支master




## github

网站，社交平台，开源项目，远程仓库

官网：https://github.com/

## windows

可视化

命令行（推荐）

## 设置：

配置用户名和密码：

```
git config --global user.name '你自己的用户名'
git config --global user.email  '你自己的邮箱'
```

## 创建版本库（文件目录）：

创建一个目录，进入到目录中执行git init，之后可以看到.git文件说明成功；这里面所有的文件都可以被git管理起来，每个文件的修改都可以被追踪；

注意：请确保目录名为英文，包括父级目录；

.git文件是用来跟踪管理版本库的，不要手动修改删除；

## 把文件添加到公司Git服务器：

建立一个文件放到仓库中，例如readme.txt

1，git status　　查看当前工作区 状态（红色：在工作区；绿色：暂存区）

2，git add 文件名称/.　. 全部

3，git commit -m -a ’本次修改注释’ 提交到版本库

3.1，git commit -a -m ’本次修改注释’ 省略掉add命令，直接提交到版本库

4,git push　　推送到远程仓库

## 备注：

执行git add readme.txt （添加文件到暂存区，git add可多次使用）

提交到当前分支的版本库 git commit -m ’本次提交的说明’

## 版本回退：

git log 查看历史版本

git reset --hard HEADˆ 回退到上一个版本（打开本地readme.txt看下是不是已经恢复到了上一个版本）

git log 再次查看历史版本（最新的版本已经看不到了，怎么办呢？）

只要命令窗口还没有关闭，网上找，找到最新版本的commit id，于是就可以指定回到未来的某个版本，执行命令：git reset --hard commitID；然后在查看readme.txt文件，已经回退到了指定的版本。

如果说命令窗口关闭了怎么办？执行命令git reflog，这个命令会记录你的每一次命令。

## 远程仓库：

github 请自行注册账号

由于你的本地Git仓库和GitHub仓库之间的传输是通过SSH加密的，所以，需要一点设置：

创建SSH Key。在用户主目录下，看看有没有.ssh目录，如果有，再看看这个目录下有没有id_rsa和id_rsa.pub这两个文件，如果已经有了，可直接跳到下一步。如果没有，打开Shell（Windows下打开Git Bash），创建SSH Key：`$ ssh-keygen -t rsa -C "你的邮箱地址"`

登陆GitHub，打开“Account settings”，“SSH Keys”页面：然后，点“Add SSH Key”，填上任意Title，在Key文本框里粘贴id_rsa.pub文件的内容：

当然，GitHub允许你添加多个Key。假定你有若干电脑，你一会儿在公司提交，一会儿在家里提交，只要把每台电脑的Key都添加到GitHub，就可以在每台电脑上往GitHub推送了。

最后友情提示，在GitHub上免费托管的Git仓库，任何人都可以看到喔（但只有你自己才能改）。所以，不要把敏感信息放进去。

如果你不想让别人看到Git库，有两个办法，一个是交点保护费，让GitHub把公开的仓库变成私有的，这样别人就看不见了（不可读更不可写）。另一个办法是自己动手，搭一个Git服务器，因为是你自己的Git服务器，所以别人也是看不见的。这个方法我们后面会讲到的，相当简单，公司内部开发必备。

## 添加远程库：（把本地代码提交到github）

1，在GitHub上创建一个新仓库；

2，把本地仓库与GitHub进行关联:git remote add origin git@github.com:xiaosi0707/09A.git

注意：把git@github.com:xiaosi0707/09A.git改成你自己的

3，把本地仓库内容推送到GitHub：git push -u origin master

注意：首次推动到远程库需要添加-u，以后就不必了

备注：当你第一次使用git的clone或push命令连接GitHub时,会得到一个警告，这是因为Git使用SSH连接，而SSH连接在第一次验证GitHub服务器的Key时，需要你确认GitHub的Key的指纹信息是否真的来自GitHub的服务器，输入yes回车即可。

## 把文件添加到远程仓库github：

建立一个文件放到仓库中，例如readme.txt

1，git status　　查看当前工作区 状态（红色：在工作区；绿色：暂存区）

2，git add 文件名称/.　. 全部

3，git commit -m -a ’本次修改注释’ 提交到版本库

3.1，git commit -a -m ’本次修改注释’ 省略掉add命令，直接提交到版本库

4,git push -u origin master　推送到远程仓库

## 查看远程库信息：

1，git remote -v查看远程库详细信息；（如果没有推送权限就看不到push地址）

## 从远程库克隆：

1，登录GitHub创建一个仓库，勾选Initialize this repository with a README;

2，克隆到本地：git clone git@github.com:xiaosi0707/09A.git

注意：git@github.com:xiaosi0707/09A.git换成你自己的

## GitHub地址类型：

GitHub给出的地址不止一个，还可以用git@github.com:xiaosi0707/09A.git这样的地址。实际上，Git支持多种协议，默认的git://使用ssh，但也可以使用https等其他协议。

使用https除了速度慢以外，还有个最大的麻烦是每次推送都必须输入口令，但是在某些只开放http端口的公司内部就无法使用ssh协议而只能用https。

删除文件：

```
git rm filename
git commit -m '删除文件'
```

误删本地文件：（从版本库恢复文件到工作区）

```
git checkout -- filename
```

## 分支：

比如jQuery，有一个稳定版本，但是我们肯定不能在当前稳定版本进行开发，这个时候就需要新建分支在新的分支上进行版本迭代。如果此分支上的测试已经没有问题那么可以和主分支master进行合并发布。

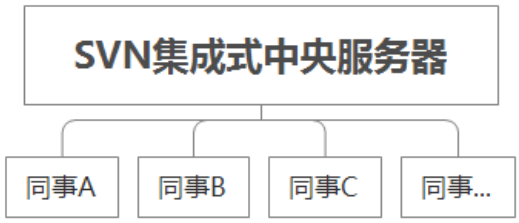
主分支master

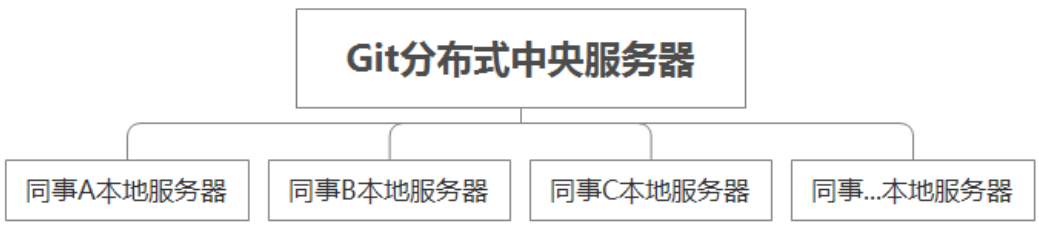
## 扩展阅读：（可跟踪的文件类型）

所有的版本控制系统，其实只能跟踪文本文件的改动，比如TXT文件，网页，所有程序代码等等，git也不例外。图片，视频这些二进制文件，虽然也能由版本控制系统管d理，但没办法跟踪文件的变化，只能把二进制文件每次改动串起来，也就是只能知道图片从100KB改成了120KB，但是 到底改了什么，版本控制系统不知道，也没法知道。

## Git和SVN的区别：

1，Git是分布式的，SVN是集成式的；Git与SVN一样都具有自己的集中式版本库或服务器，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/服务器拉取代码后会在自己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。



2，Git分支和SVN分支不同：

分支在SVN中一点都不特别，就是版本库中的另外一个目录。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。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。

3，Git的内容存储使用的是SHA-1哈西算法，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。

4，查看日志：

Git： 本地包含了完整的日志，速度很快无需网络；

SVN：需要从服务器拉取；

5，本地分支：

Git有本地分支

SVN无本地分支

[git详细教程]