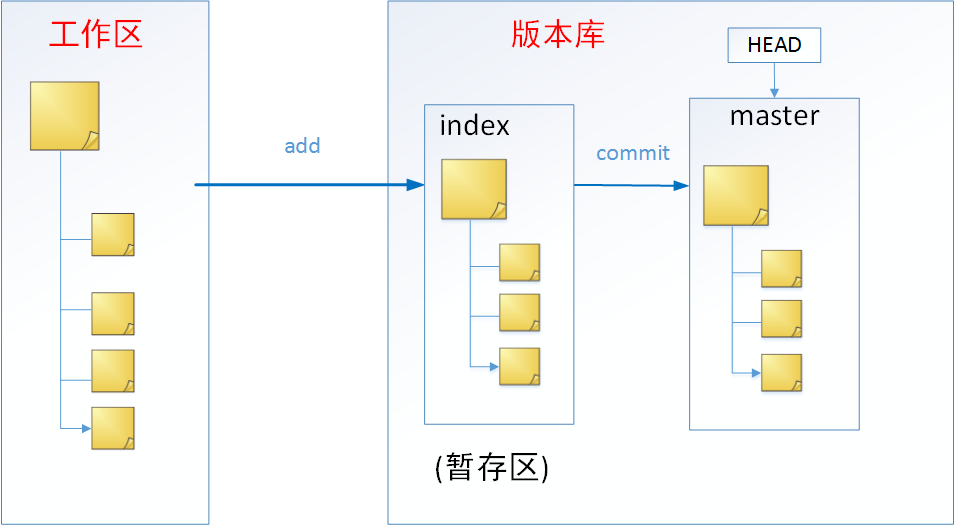
SVN 与 Git 的最主要的区别：

1. Git 是分布式版本控制系统，版本分布在各地；SVN 是集中式版本控制系统，版本库是集中放在中央服务器的。
2. Git 把内容按元数据方式存储，而SVN是按文件。
3. 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。
4. Git 的内容完整性要优于 SVN：Git 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 使用

工作区、暂存区和版本库的关系：



**工作区：**  
就是你在电脑里能看到的目录，比如我的 test 文件夹就是一个工作区  
**暂存区：**  
工作区有一个隐藏目录 .git，这就是 Git 的版本库。在版本库中标记为 "index" 的区域是暂存区（stage, index），标记为 "master" 的是 master分支所代表的目录树。图中我们可以看出此时 "HEAD" 实际是指向 master 分支的一个"游标"。

**仓库操作**

**初始化一个仓库              git init  
修改的文件添加到缓存      git add file  
缓存内容提交到仓库          git commit -m "commit instructions"**

**仓库状态**

**查看当前仓库的状态          git status  
对比工作区和缓存区          git diff  
对比工作区与版本库         git diff HEAD -- file**

**管理修改**

**丢弃工作区的修改         git checkout -- file       
丢弃缓冲区的修改      git reset HEAD file     #回到add之前的状态**

**删除文件           git rm file**

**版本切换**

**查看提交历史               git log  
查看命令历史           git reflog      #包括提交, 回退等  
回退到前面的版本           git reset HEAD~1     #1表示前一个版本**

**跳转到指定版本             git reset commitid #commitid 是提交版本号**

**Git 连接 GitHub 远程仓库**

**关联一个远程仓库**

**Git remote add origin** [**git@github.com:xgf13469/MyRepos.git**](mailto:git@github.com:xgf13469/MyRepos.git)**#ssh 传送**

**推送本地仓库             git push origin master  
克隆一个远程仓库**

**git clone** [**https://github.com/xgf13469/MyRepos.git**](https://github.com/xgf13469/MyRepos.git)**#https 传送**

**拉取远程仓库并合并 git pull**

**分支管理**

**查看分支          git brance**

**创建分支          git branch dev      # dev 是分支名**

**切换分支          git checkout dev**

**创建+切换         git checkout -b dev**

**合并分支 git merge dev #分支 dev 合并到 master**

**删除分支 git branch -d dev #已合并的分支**

**强行删除分支         git branch -D dev    #分支未合并   
查看分支图            git log –graph**

**git checkout -b dev origin/dev       #创建远程仓库的分支到本地，必须先克隆或关联一个远程分支**

**SVN**

2. **检出**

svn checkout path        # path 是服务器上的目录

简写：svn co

例如：svn checkout svn://192.168.1.1/test

svn checkout [https:](https://github.com/xgf13469/test)[//](https://192.168.1.1/test)[github.com/xgf13469//test](https://test)   #我的 github 地址 ：）

3、**添加**新的文件

svn add file

例如：svn add readme.txt    #添加 readme.txt 文件

   svn add \*.txt   #添加当前目录下所有的 .txt 文件)

4、将改动的文件**提交**到版本库

svn commit -m "LogMessage" [-N] PATH file # -m 后接本次提交说明

简写：svn ci

例如：svn commit -m "add file for my test" readme.txt

5、加锁、解锁  
svn lock -m "LockMessage" [--force] PATH file   #加锁，别人无法修改

svn unlock PATH

例如：svn lock -m "lock test file" readme.txt

6、**更新**到某个版本

svn update -r m path     #更新到版本号为 m 的版本

简写：svn up

例如： svn update  # 后面没有目录，默认将所有文件更新到最新版本。

 svn update -r 10 readme.txt #将版本库中的文件readme.txt 更新到版本10

 svn update readme.txt #更新本地文件，使得和版本库同步。

如果在提交的时候提示过期的话，是因为冲突，需要先 update，修改文件，然后清除 svn resolved，最后再提交commit 。

7、查看文件或者目录**状态**

svn status path      #目录下的文件和子目录的状态，正常状态不显示

简写：svn st

状态类型：

* ?：不在svn的控制中；
* M：内容被修改；
* C：发生冲突；
* A：预定加入到版本库；
* K：被锁定

svn status -v path      #显示文件和子目录状态

第一列保持相同，第二列显示工作版本号，第三和第四列显示最后一次修改的版本号和修改人。

8、**删除**文件

svn delete path -m "delete test fle"

简写：svn del 或svn rm

例如：svn delete readme.txt

9、查看**日志**

svn log [path]

例如：svn log       #显示所有修改记录及版本号的变化

svn log readme.txt      #显示文件 readme.txt 的修改记录及版本号

10、查看文件详细信息

svn info path

例如：svn info readme.txt

11、**比较差异**

svn diff path         #将修改的文件与基础版本比较

svn diff -r m:n path           #对版本m和版本n比较差异

简写：svn di

例如：svn diff readme.txt

svn diff -r 10:11 readme.txt

16、**标记冲突已解决**

svn resolved            #移除工作副本的目录或文件的"冲突"状态。