# Lecture 2

**Software configuration management (SCM)**软件配置管理的四个方面

- Change control（没说）
- Version control
- Building
- Relsasing（没说）

## Version Control System(VCS)

VCS is a software system that keeps track of the changes made to a set of files so that you can recall a specific version.

1. collaborate on a project with multiple other developers, merging changes and resolving conflicts.
2. revert changes.
3. go back in time to a specific version (tags can be your friend).

**Subversion(SVN)**

- Creating local copy(切换分支)

  `svn checkout <address_to_remote> <name_of_local_dir>`

  `git checkout <branch-name>`

- Committing loacl changes(commit评论)

  `svn commit -m "msg"`

  `git cmoomit -m "msg"`

- Updating local copy(更新本地文件)

  `svn up`

  `git pull upstream master`

- Telling svn/git about a new file to track(增加文件追踪)

  `svn add <file-name>`

  `git add <file-name>`

- 其他的svn命令

  `svn st(status)`: shows the status of files in the current svn directory. 是在提交前查看本地文本和版本库里面的文件的区别。
  `svn rm`: removes a file from the set of tracked files (will be removed on the remote server as well)
  `svn mv`: moves a file from one directory to another (or renames if in same directory)
  `svn diff`: diff between two revisions, or diff a file to see uncommitted local changes

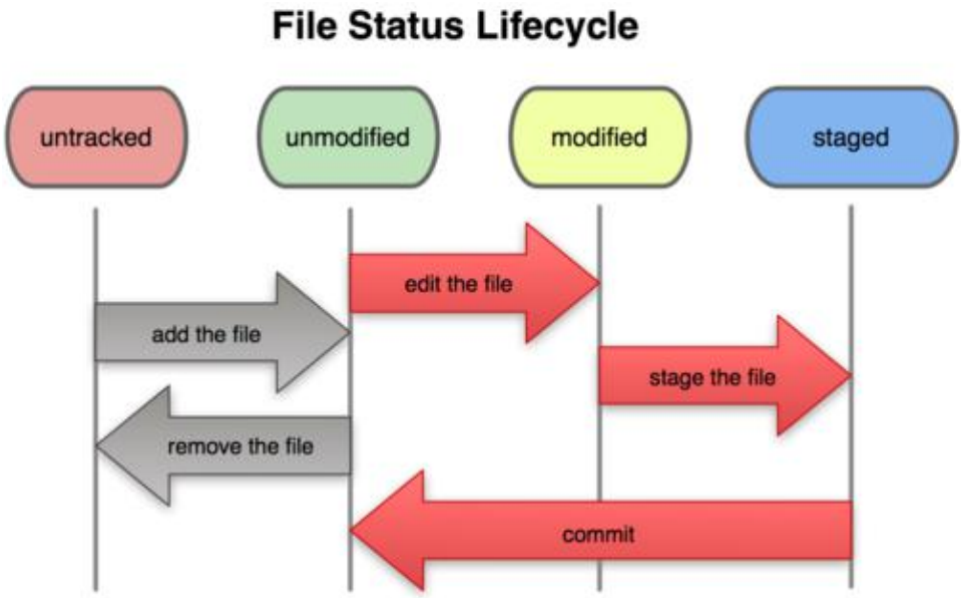
  `svn:ignore -R *.class`: ignore ".class" files

svn Directory layout: 1. **Trunk**(Master): most up to date current dev. 2. **Branches**: releases, bug fixes, experimental. Do not branch to: support a different hardware, or support a different customer. 3. **Tag**: Mark a state of the code (release for e.g.)

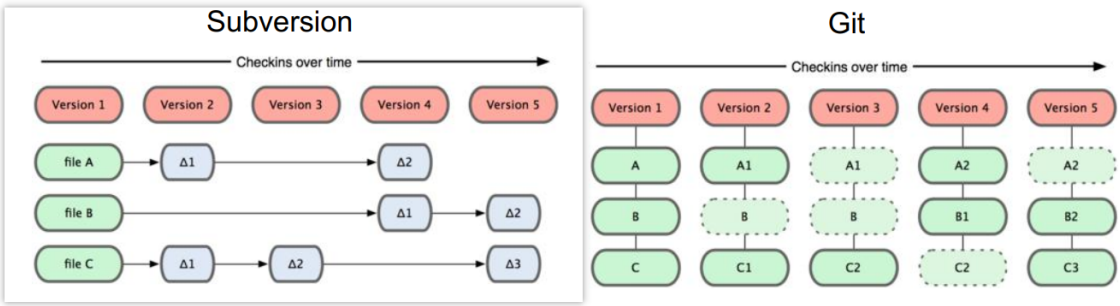
**GIT**

Git generates a **unique SHA-1 hash – 40** character string of hex digits, for every commit. Refer to commits by this ID rather
than a version number. Often we only **see the first 7 characters**.

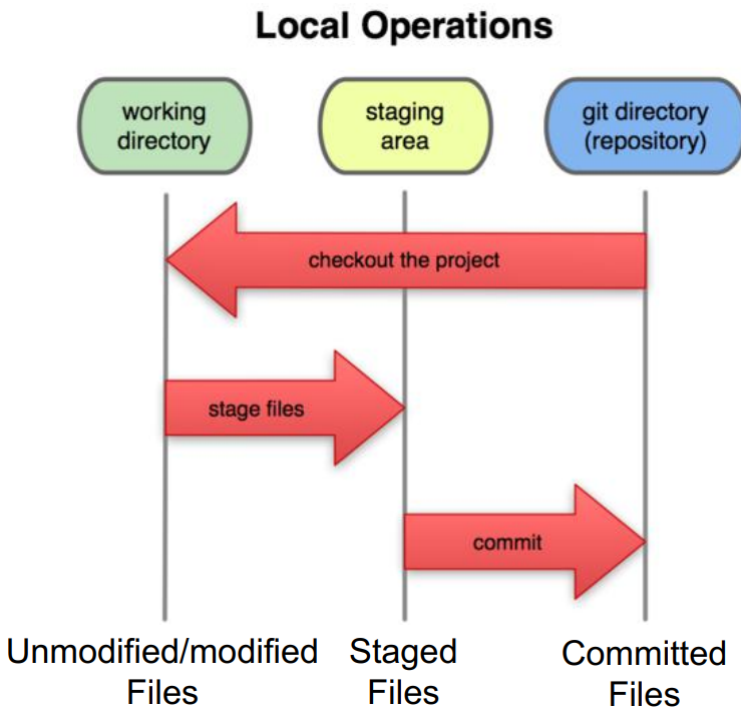
**Git file lifecycle**:




**Git takes snapshots(快照)** : subversion关心文件内容的具体差异。第一次保存了完整的数据，往后每次保存的都不是完整的数据，只会记录基于之前的版本和现在两者的变化信息，对于此外没有变化的都不会去记录。git是记录和组装一系列快照流的微型系统，关心文件数据的整体是否发生变化，每次commit的时候保存一次快照，而每个快照都包含了完整的数据。

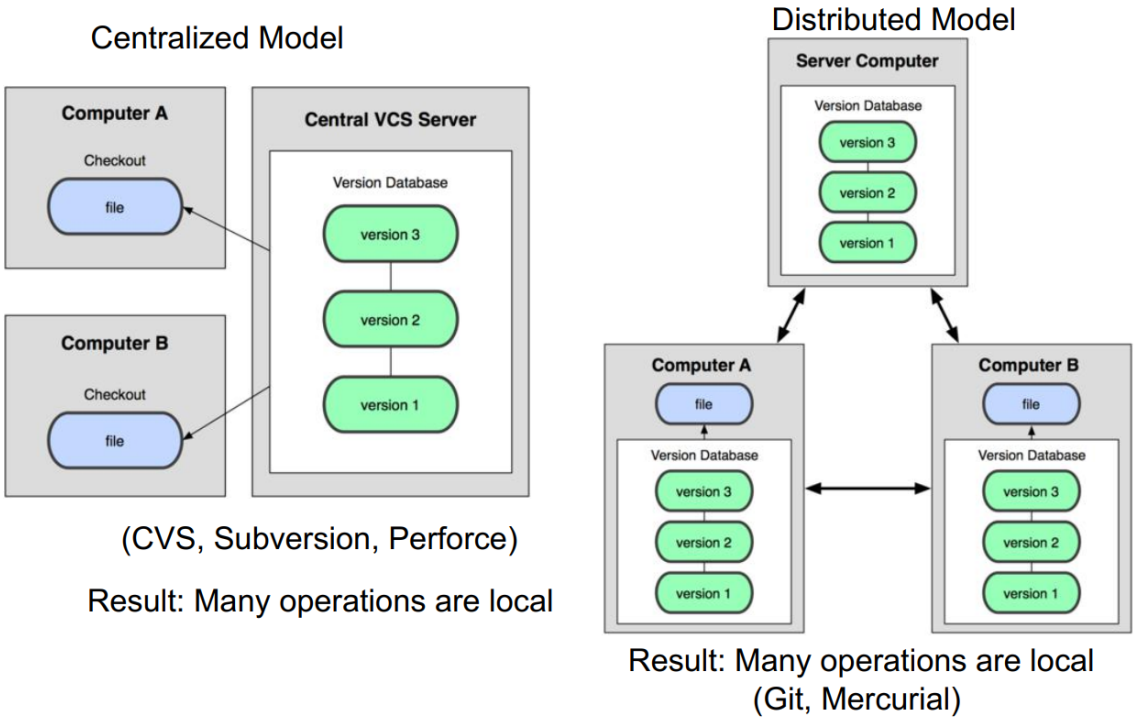



**Local Git**: Three areas




**Basic WorkFlow**: 1. Modify files; 2. Stage files, adding snapshots of them to your staging area; 3. Do a commit, which takes the files as they are in the staging area and stores that snapshot permanently to your Git directory.

**SVN vs. GIT**: SVN是中央储存库的方法，只有主储存库才有完整的文件历史；用户checkout当前版本的副本。GIT是分布式储存库的方法，每一个checkout都是一个完整的储存库，有完整的文件历史；有着更大的冗余和速度；分支和合并存储库的使用更加频繁。




---

**Parallel Work**: 1. locking: 要强制等另一个人完成; 1. merging: 两个人改同一行；代码不冲突但是合并导致bug

# Building Management

**Building**应该是自动的

**Daily build and smoke test**(冒烟测试其实就是每日build建立后，对系统的基本功能进行简单的测试。这种测试强调程序的主要功能进行的验证，而不会对具体功能进行更深入的测试。)

- Ways to break the build process
    - Check in bad code
    - Forget to include file in makefile
    - Move a library
- Every day (night) build the latest version of product and run simple test suite

**Branches**的一些事:

- 传统的版本控制尽可能避免使用长期存在的(long-lived)分支，现代版本控制鼓励使用短期存在的(short-lived)分支。
- 使用分支的好原因: 1. 在特定的版本中修复bug. 2. 体验版本. 3. 政治斗争？？？
- 使用分支的坏原因: 1. 支持不同的硬件平台。2. 支持不同的客户

**Testing:**

- **Smoke Test(冒烟测试)**: 确保在进行更改后系统仍在运行
- **Unit Test(单元测试)**:确保一个模块在进行更改后被破坏
- **Regression Test(回归测试)**: 确保做了其他提升后，代码不会变得更糟