CMSC 128 Laboratory Handout 2
Version Control with GitHub and Google Code

Version Control
- a system that *records changes* to a file or set of files over time so that specific version can be recalled later
- a program that can record multiple versions of a source file, storing information such as the creation time of each version, who made it, and a description of what was changed

Three Important Capabilities of Version Control
1. *Reversibility* – the ability to back up to a previous state if it was discovered that some modification was a mistake or a bad idea
2. *Concurrency* – the ability to have many people modifying the same collection of files knowing that conflicting modifications can be detected and resolved
3. *History* – the ability to attach historical data to the codes, such as explanatory comments about the intention behind each change to it. For a solo programmer, change histories are important to aid memory while for a multi-person project, they are an important form of communication among developers

Types of Version Control Systems
- Local Version Control Systems
  - have a simple database that kept all the changes to files under revision control

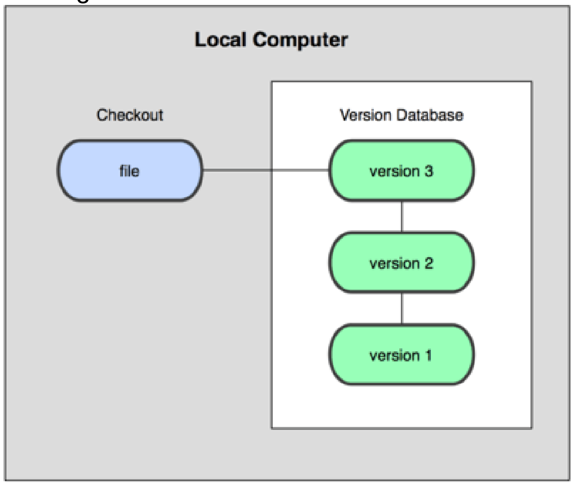

Figure 1. Local Version Control Diagram

- Centralized Version Control Systems
  - have a single server that contains all versioned files and a number of clients that check out files from that central place
  - offers many advantages like for example, everyone knows to a certain degree what everyone else on the project is doing
  - it also has some serious downsides like the single point of failure that the centralized server represents

Prepared by:
Kristine Elaine P. Bautista
Instructor 1
Institute of Computer Science, College of Arts and Sciences
University of the Philippines Los Baños

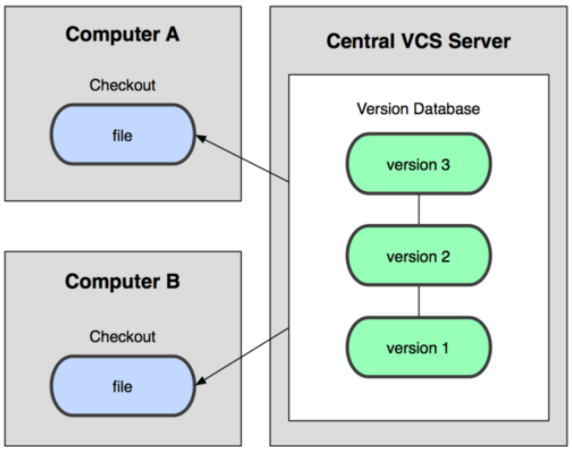

Figure 2. Centralized Version Control Diagram

- Distributed Version Control Systems
  - clients don't just check out the latest snapshot of the files, they fully mirror the repository
  - If any server dies, and these systems were collaborating via it, any of the client repositories can be copied back up to the server to restore it. Every checkout is really a full backup of all the data

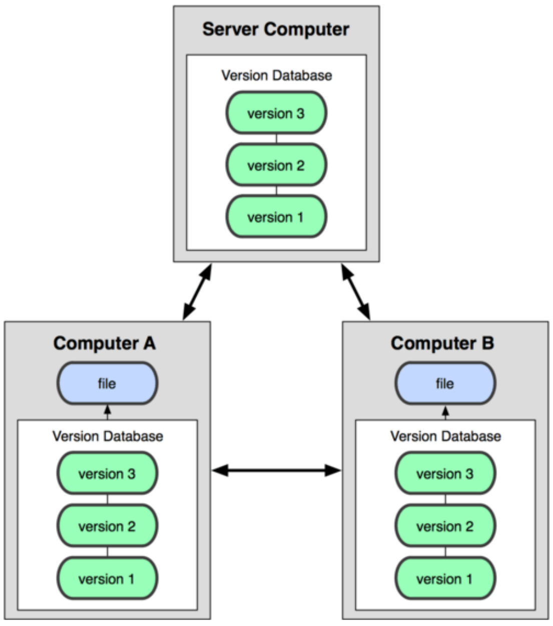

Figure 3. Distributed Version Control Diagram

Some Version Control Systems
- *CVS* – allows concurrent mutli-user development either locally or over the network. Unlike newer systems, it lacks support for atomic commits and file moving/renaming.
- *Subversion (SVN)* – a free version control system designed to be similar to CVS but without its problems (e.g., it supports atomic commits of filesets, and versioning of directories, symbolic links, meta-data, renames, copies and deletes)
- *Git* – a decentralized version control system originally invented by Linus Torvalds to support development of Linux (his kernel)
- *Mercurial* – a decentralized version control system broadly resembling Git

Concepts of Version Control

- A *file under version control* is one that is registered in the version control system
- *Repository* – stores both the file's present state and its change history. It also contains other information, such as log entries that describe the changes made to each file.
- *Work file* – the copy of a version-controlled file that you actually edit
- *Commit (or check in)* – records the changes in the repository, along with a descriptive log entry
- *Working tree* – directory tree of work files

Merging

- each user may modify a work file at any time
- The system lets you merge your work file, which may contain changes that have not been committed, with the latest changes that others have committed.

Locking

- work files are normally read only.
- To edit a file, the user asks the version control system to make it writable by locking it. Only one user can lock a given file at any given time.

Types of Log Files

- Version Control Log
  - log maintained by the version control system
  - each time a change is committed, a log entry for the change is filled out
- Change Log
  - provides a chronological record of all changes to a large portion of a program (typically one directory and its subdirectories)

Git

- a distributed revision control and source code management (SCM) system with an emphasis on speed
- initially designed and developed by Linus Torvalds for Linux kernel development in 2005

Apache Subversion (SVN)

- a software versioning and revision control system
- used by developers to maintain current and historical versions of files such as source code, web pages and documentation

Github

- a web-based hosting service for software development projects that use the *Git* revision control system

Basic Commands

- *git init* – initializes a new Git repository
- *git status* – check the status of your repository and see which files are inside it, which changes still need to be committed, and which branch of the repository you're currently working on
- *git add <filename>* - tells Git to start tracking changes to the new file
- *git commit -m "<your commit message>"* – stores staged changes to the repository
- *git log* – lists *all the changes* that were committed in the order they were committed from the most recent commit
- *git remote add <name of remote repository> <repository URL>* - add a remote repository on GitHub
- *git push -u <name of remote repository> <local branch name>* - tells the Git to make commits visible online (on GitHub)
- *git pull <name of local repository> <local branch name>* - get the most up-to-date version of the repository
- *git diff <commit>* - see what is different from the current repository and the last commit
- *git diff --staged* – see changes that were just staged
- *git reset <pathname of file to be removed>* - remove the file from the stage
- *git checkout -- <target>* - change the files back to how they were at the last commit
- *git branch <branch name>* - create a branch
- *git checkout <branch name>* - switch branches
- *git rm '<filename>'* - delete file from disk and stage removal of the files
- *git merge <branch name>* - merge changes from the branch to the master branch
- *git branch -d <branch name>* - delete a branch

Google Code

- Google's site for developer tools, APIs and technical resources
- contains documentation on using Google developer tools and APIs
- also features a variety of developer products and tools built specifically for developers like *Project Hosting* which gives users version control for open source code
- runs project hosting service that provides revision control offering Subversion, Mercurial and Git, an issue tracker and a wiki for documentation

Some helpful version control tools

- Github: https://github.com/
- SourceTree: http://www.sourcetreeapp.com/
  - a free Mercurial and Git Client for Windows and Mac that provides a graphical interface for Hg and Git repositories
- TortoiseSVN: http://tortoisesvn.net/
  - a Subversion client implemented as a Microsoft Windows shell extension. It helps programmers manage different versions of the source code of their system.

References:

- *Git – About Version Control*. (2008) Retrieved December 9, 2013, from: http://git-scm.com/book/en/Getting-Started-About-Version-Control.
- *Version Control – GNU Emacs Manual*. (2013) Retrieved December 9, 2013, from: http://www.gnu.org/software/emacs/manual/html_node/emacs/Version-Control.html.
- *Code School – Try Git.* (n.d.) Retrieved January 5, 2014, from: http://try.github.io
- *Orsini, L. (2013). GitHub For Beginners: Don't Get Scared, Get Started.* Retrieved January 5, 2014, from: http://readwrite.com/2013/09/30/understanding-github-a-journey-for-beginners-part-1#awesm=~os2QEtY3GKqfNB.
- *Google Code.* (n.d.) Retrieved January 13, 2014, from: https://code.google.com/