**Pro Git**

**Chapter 1 – Getting Started**

Any type of file can use Git.

* **VCS** – Version Control System
* **CVCS** – Centralized Version Control System (CVS, Subversion, Perforce)
* **DVCS** – Distributed Version Control System (Git, Mercurial, Bazaar, Darcs, Bitkeeper). Fully mirror the repository, every clone is a full backup.

Any unchanged files in a commit are saved as links to the previous snapshot. Git only adds data, so it is difficult to lose once committed.

Git can be used offline, since the entire project is there.

Impossible to change the contents of any file without Git knowing about it.

Uses the SHA-1 hexadecimal 40-character hash for integrity (checksum).

* **Committed** – Data is safely stored in your local database.
* **Modified** – File has been changed, but not committed yet.
* **Staged** – Modified file has been marked to go into the next commit.

Three main sections of a Git project:

* **Git directory (repository)** – Most important part, this is where Git stores the objects and metadata for your project. It is what is copied when you clone a repository from another computer. This is where commits (snapshots) are stored permanently.
* **Working tree (directory)** – Single checkout of one version of the project. This is where you modify files.
* **Staging area** – A file, generally contained in your Git directory, that stores information about what will go into your next commit. Staging area is also known as the index. You stage only the files you want to be part of your next commit.

Git config file is located in the user’s “home” directory. Or type “**git config –list**”, “**git help config**”, then “**q**” to exit.

**Chapter 2 – Git Basics**

**git init** – Creates a new repository in the current directory.

**git clone [location]** – Pulls down all data for a repository.

**git status** – Shows which files are in which state.

**git status -s** – Shows abbreviated version. New files added to staging area = “A”, modified files = “M”, and untracked files = “??”. Left column is staging area, right column is working tree.

**git add [path and/or filename]** – Add that file or directory to be tracked.

To STAGE a modified file, you need to run “**git add**” again. If you go back and edit again, it will show one version of the file as staged and one version as unstaged. You’ll need to “**git add**” it again.

The “**.gitignore**” file will ignore certain types of files. Go to <https://github.com/github/gitignore> for examples.

**git diff** – Shows the patch (the lines that have been added and removed since the last commit), but not yet staged.

**git diff –staged** – Shows the changes that have been staged (“**git diff –cached**” is the same).

**git commit** – Opens default editor (Notepad++) to write a brief message, then commits all staged files.

**git commit -m “Enter message here”** – bypasses the editor.

**git commit -a -m “Message”** – Will bypass “add” for everything modified, be careful with this, it might include unwanted changes.

**git rm [filename]** – Stages a file for removal from Git, and deletes it from the directory (does NOT go to Windows Recycle Bin). Just deleting the file won’t remove the file from Git.

**git log** – Shows the history of commits, in reverse chronological order.

**git log -p** or **git log –patch** – Shows only the patches.

**git log -2** – Shows only the last two commits.

**git log –stat** – Shows abbreviated statistics.

**git log –graph** – Shows the branch and merge history.

**git log –pretty**

* **=oneline** – prints each commit on a single line.
* **=short, =full, =fuller** – Shows progressively more information.
* **=format [options]** – Custom format.

**git log –since** or **git log –after** – Only commits from a certain date.

**git log –before** or **git log –until** – Only commits from before a certain date.

Author is the one who originally wrote the patch, committer is the one who applied it, so everyone gets credit.

**git commit –amend** – If you do a commit and forget to add some files. Message is optional, the snapshot will overwrite your previous commit.

**git reset HEAD [filename]** – Unstages a file. Can also be dangerous, especially with **–hard** flag.

**git checkout – [filename]** – Restores a modified file to last commit. This is dangerous, because all changes are now gone. Only things that have been committed can be recovered.

git remote –