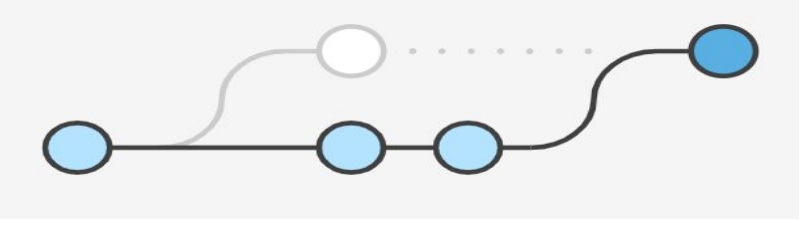


Git Basics



Submitted by:- Anupama M Shetty

Reg No:- 370CS20001

Submitted to:- Mr. Dinesh Balgi

Departement :- Computer Science And Engg

Table of content

|  |  |
| --- | --- |
| SL.NO | Title |
| 01 | Version Control System |
| 02 | What is a Git? |
| 03 | Installing and Learning Git |
| 04 | Git Workflow |
| 05 | Initial git Configurations |
| 06 | How to use Git |
| 07 | Basic Git Commands |
| 08 | Merge Conflicts |

**Version control systems**

● Version control (or revision control, or source control) is all about managing multiple versions of documents, programs, web sites, etc.

* Almost all “real” projects use some kind of version control
* Essential for team projects, but also very useful for individual projects

● Some well-known version control systems are CVS, Subversion, Mercurial, and Git

* CVS and Subversion use a “central” repository; users “check out” files, work on them, and “check them in”
* Mercurial and Git treat all repositories as equal

● Distributed systems like Mercurial and Git are newer and are gradually replacing centralised systems like CVS and Subversion.

**About Git**

* Created by Linus Torvalds, creator of Linux, in 2005
  + Came out of Linux development community
  + Designed to do version control on Linux kernel
* Goals of Git:
  + Speed - instead of copy each developer gets their own repository
  + Support for non-linear development (thousands of parallel branches)
  + Fully distributed- everyone will get the copy of the code and can clone it
  + Able to handle large projects efficiently
  + (A "git" is a cranky old man. Linus meant himself.)

**Installing/learning Git**

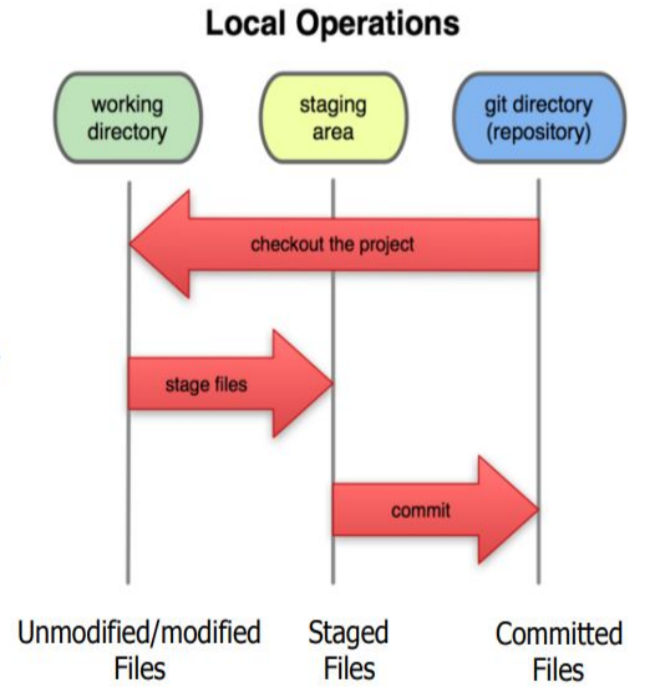
● Git website: http://git-scm.com/

* Free on-line book: http://git-scm.com/book
* Reference page for Git: http://gitref.org/index.html
* Git tutorial: http://schacon.github.com/git/gittutorial.html
* Git for Computer Scientists:
* http://eagain.net/articles/git-for-computer-scientists/

● At command line:

* (where verb = config, add, commit, etc.) – git help verb

**Local git areas**

● In your local copy on git, files can be:

* In your local repo
* (committed)
* Checked out and modified, but not

yet committed

* (working copy)
* Or, in-between, in a "staging" area
* Staged files are ready to be

committed.

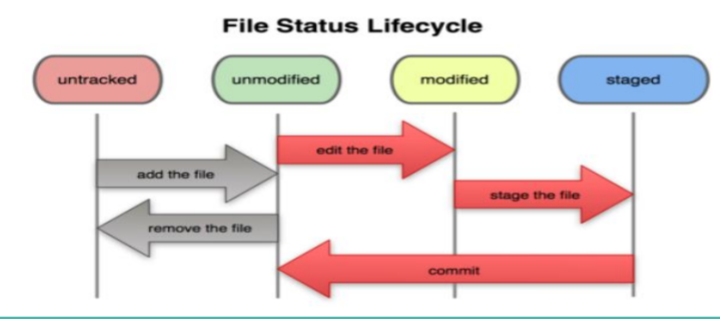
* A commit saves a snapshot of

all staged state.

**Basic Git workflow**

**● Stage** files, adding snapshots of them to your staging area.

**● Commit**, which takes the files in the staging area and stores that snapshot permanently to your Git directory.



**Initial Git configuration**

**● Initialize the local directory as a Git repository.**

○ git init -b main

**● Clone remote Repository**

○ git clone https://external-host.com/ *extuser*/*repo.git*

**● Set the name and email for Git to use when you commit:**

*○ git config --global user.name "Bugs Bunny"*

○ git config --global user.email [bugs@gmail.com](mailto:bugs@gmail.com)

○ You can call git config –list to verify these are set.

**Using your repositories**

● Get the files from your repository before starting

○ Make a local repository as a clone of master

■ git clone /opt/git/csc480ASpring15.git

■ See all the contents of the folder

● ls -a to see the .git folder.

● Make changes

○ See what changed

■ git diff <File Path>

○ Stage changes

■ git add –all (or particular files)

■ git diff –cached

■ Still only in your repository

● Verify status

○ git status

● Put changes back up into repository

○ Commit your staged changes in your repository

■ git commit -m "the reason for the change"

○ Update the repository:

■ git push origin

● See what is on the repository

○ git remote

● Get what is on repository

○ git pull

○ If it says to resolve manually, just vi that file and see the head which is yours

**Git commands**

|  |  |
| --- | --- |
| Command | Description |
| git clone url [dir] | copy a Git repository so you can add to it |
| git add file | adds file contents to the staging area |
| git commit | records a snapshot of the staging area |
| git status | view the status of your files in the working directory and staging area |
| git diff | shows diff of what is staged and what is modified but unstaged |
| git help | get help info about a particular command |
| git pull | fetch from a remote repo and try to merge into the current branch |
| git push | push your new branches and data to a remote repository |
| other | init,reset,branch, checkout, merge, log,tag |

**Add and commit a file**

● The first time we ask a file to be tracked, and every time before we

commit a file, we must add it to the staging area:

○ git add Hello.java

○ Takes a snapshot of these files, adds them to the staging area.

○ In older VCS, "add" means "start tracking this file." In Git, "add" means "add to

staging area" so it will be part of the next commit.

● To move staged changes into the repo, we commit:

○ git commit –m "Fixing bug #22"

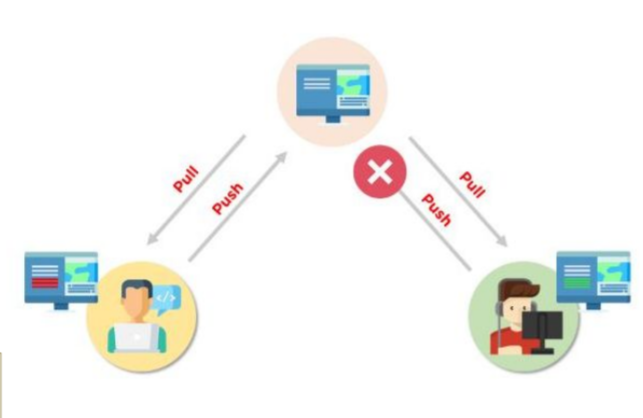
● To undo changes on a file before you have committed it:

○ git reset HEAD -- filename (unstages the file)

○ git checkout -- filename (undo your changes)

○ All these commands are acting on your local version of repo.

**Merge conflicts**

****A merge conflict is an event that

takes place when Git is unable

to automatically resolve

differences in code between two

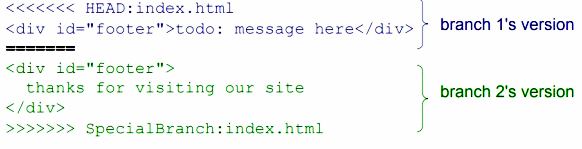
commits. Git can merge the

changes automatically only if

the commits are on different

lines or branches.

**Resolving Merge conflicts**

****

**Git GUI Tools**

● You can find various GUI tools for git which are very useful in daily

life: https://git-scm.com/downloads/guis