Notes on Git

Notes taken by
Nadeem Mehraj
Intern
Applied Informatics Inc

Version Control Systems

A version control system records changes to a file or set of files over time so that one can recall specific versions later. For example, during the development of software different features may be added at different times and it may happen that some feature added causes the software to malfunction and we would like to restore the software prior to that feature was added. Version control system also allows one to recover files in case the files are lost.

Local Version Control System
In a local version control system, a local computer contains the version database that stores different versions of the project. The user can access the specific version of the project from that database. One of the limitations of the local version control systems is that people cannot collaborate on projects as the computer can be used by only one user. Another limitation is that since the version database is present at only one place, failure of the computer system will result in the loss of entire projects.

Centralised Version Control Systems
In a centralised version control system, a single server contains the version database that stores different versions of the projects and different clients can access the different versions of the project from that server. This allows people to collaborate on projects. However, since the version database is still present at only one place, failure of the system will result in the loss of entire project. Also during the time the server is down, people cannot collaborate.

Distributed Version Control Systems
In a distributed version control system, the clients instead of accessing just specific versions of projects access the entire version database and therefore fully mirror the server. Thus if the server fails, it can be restored back from one of the mirrors.

Git doesn't store data as a series of changesets or differences, instead it stores data as a series of snapshots

File State
In Git, a file can be in one of three states - **committed**, **modified**, and **staged**

A file in the Git directory is **committed**, a file marked to go into the next commit is **staged**, and a file in the working directory that has been modified but not marked to go into the next commit is **modified**

Sections of Git Project
The **Git directory** is where the metadata and object database for the project is stored
The **working tree** is a single checkout of one version of the project.
The **staging area/index** is a file that stores information about what will go in the next commit
Typically there are two ways of obtaining a git repository:
Turn a local directory into a Git repository
Clone an existing Git repository

To turn a local directory into a Git repository, navigate to the directory and then type the command:

git init

This command creates a subdirectory .git under the current directory. To start version controlling, use the commands:

git add FILENAME
git commit -m "Commit String"

The first command computes a checksum(SHA1 hash) for the file FILENAME, adds the file to the Git repository and adds the checksum to the staging area while the second command computes checksum for each directory,stores tree objects in the Git repository, creates a commit object that contains a pointer to the snapshot of the content staged, the name and email address of the author, the string Commit String, and pointers to the commit or commits that came directly before this commit. To clone an existing repository located at URL, use the command:

git clone URL [DIRECTORY]

This command retrieves all the data that the server has about the repository not just the working copy and can be used to restore the server back in case of corruption of server disk. The command initializes a .git directory inside the directory DIRECTORY, gets all the data for the repository, and checks out a working copy of the latest version.

File Status

A file in the working directory can either be **tracked** or **untracked**. Tracked files, in turn, can be **unmodified**, **modified** or **staged**. Tracked files are the files that were in the last snapshot while untracked files are those that were not in the last snapshot and are not in the staging area. When a repository is first cloned, all the files are tracked and unmodified. To check the status of files in the working directory, use the command:

git status

An untracked file appears under the heading "Untracked files" in the output of git status command. A tracked and staged file appears under "Changes to be committed" heading while a tracked and modified file appears under "Changes not staged for commit" heading in the output of git status command.

Tracking files

To start tracking an untracked file named FILE, use the command:

git add FILE

This command causes FILE to be tracked and staged to be committed.

Staging Modified files

If a tracked file is modified, it will appear under "Changes not staged for commit" in the output of git status command. This means that the file is tracked and has been modified but is not staged. To stage the modified file FILE, use the command:

git add FILE

Working with Remote repositories

Managing remote repositories includes adding repositories, removing remotes, define remote branches as being tracked or not among other things. To see the short names of configured remotes, type

git remote

Using the above command with -v option shows the URLs corresponding to the remotes to be used when reading and writing data to those remotes
It is possible to have more than one remote for a single repository
To add a new remote repository as a shortname SHORTNAME and url URL, type

git remote add SHORTNAME URL

To get data from remote project with shortname REMOTE, type

git fetch REMOTE

The command gets all the data from the remote project that you don't have

To incorporate the changes that you have comitted locally in the branch BRANCH to the remote repository with shortname REMOTE, type

git push REMOTE BRANCH

To view more information about a remote with shortname REMOTE type

git remote show REMOTE

The above command lists the URL for the remote repository, which branches on the server are tracked, which remote branches you don't have, etc

To rename a remote with shortname ABC to XYZ, type

git remote rename ABC XYZ

This will rename remote tracking branches as well

To remove a remote with shortname REMOTE,type

git remote remove REMOTE or **git remote rm REMOTE**

This will remove the remote tracking branches as well

Branching

Nearly ever VCS has support for branching. However, many VCS require to create a new copy of source code directory which can take time for large projects. Branching in Git is incredibly lightweight. Branching operations and switching back and forth between branches is fast

A branch in Git is a movable pointer to one of the commits. Creating a new branch creates a new pointer to the commit one is currently on and can be done as

git branch BRANCHNAME

To keep track of what branch the user is currently on, Git keeps a special pointer called HEAD that points to the current branch.

To switch to the branch BRANCHNAME, type

git checkout BRANCHNAME

The above command causes the head pointer to point to BRANCHNAME and also causes the files in the working directory to correspond to the current commit of BRANCHNAME