Git in nutshell

Understanding Git Basics:

Git, is the most commonly used version control systems. It helps you to track changes to the files. It maintains a history of all the changes done the file which can be helpful to restore the file to a specific version if case you ever have to do it.

Git also makes collaboration easier, allowing multiple people to all merged into one source.

Git runs locally and their history gets stored on your local machine. This makes the git super-fast as most of the operations are performed locally and very few needs network connection to push your local changes i.e., files and revision history to remote server/host (GitHub or Bitbucket).

Storing the files on remote host gives you a centrally located place where you can upload your changes or download (pull) changes made by others thereby letting you to collobrate with other developers.

The collection of Files along with all its history managed by Git is termed as Repository. And folder containing all the files is called the working directory or workspace. Workspace may have files or folder which are excluded(ignored) by the Git.

The root folder contains a hidden folder named “.git” which is the actual git repository and contains all the information required by the git to track the changes and also which files to be ignored by the version control and so on.

Git maintains the current state of the files in a snapshot called “commits”. Commits can contain one or more files changes. Please note Git version files and not folder.

As you make changes, commits are saved on timeline known as “Branch”. Git Repository will have at least one branch and the main branch is called “Master”.

Subsequent articles

Installing git your machine:

You can download the git from <https://git-scm.com/download>

Once the git is installed you may verify by typing the below command at the command prompt

git version

Configure Git: -

Git requires your name and email address before any real work can be done. It is best to just configure Git from the start.

git config --global user.name "Your Name"

git config --global user.email “[your.email@your-place.com](mailto:your.email@your-place.com)”

Git Initialization:

Now that we have installed the Git and done a basic configuration, it is time to create a new repository on your local machine.

Open up your favourite Terminal. It is always best practice to create a separate folder to store all your git repositories. So on my machine, i have created a folder in home directory with name “git”. You may give any name you may like. Just type the below commands.

mkdir ~/git

cd ~/git

Now let’s create a empty repository with name “git-notes” with command: -

git init git-notes

once the command runs successfully you will below message

Initialized empty Git repository in <git folder path >/.git/

Git States

With respect to the files being manged by Git, There are 3 local git stages and they are;

Stage 1: - Working Directory/Workspace

Stage 2: - Staging Area

Stage 3: - Git repository(.git folder)

Workspace will have all the files related to your application which may either be managed or not managed (.gitignore files) by Git. And .git folder will have all the changes along with revision history committed to the repository and in between we will have a staging area which is used to prepare for next commit. Files are moved to Staging area from Workspace and then committed to the git repository.

Apart from these 3 stages, there is also a remote stage where the files are pushed to the remote host (GitHub or Bitbucket) to store on the centralized place. Please note we are not going to cover the remote stage this article.

Git Commit steps:

To commit the application files to Git repository, first add all the files related to the application in your working directory (Stage 1) which in this case it will be “git-notes” folder

First Type the following command for Git status which will list all the files which are not tracked by Git, or modified by git and file stage.

git status

When you run git status for the first time after copying the files in working directory, it will list all the files under untracked files

Next we have to move the files to staging area (Stage 2) and this is achieved by following cmd

git add .

dot indicates to include all the files in staging area

to verify if it is actually in staging area run git status command again.

Now comes the last stage, i.e. committing files to repository. To commit execute following cmd

git commit -m “commit message”

There is also a way where you can combine the add and commit in one command and it is done using following command

git commit -am “commit message

Reset Changes:

At a times you may need to backout the changes you made to the files for that you have use reset and checkout command as shown below

git reset HEAD <file> - This is unstage the changes i.e. it will move the back to Stage 1 (tracked change in working directory) from stage 2 (Staging area)

One unstaging has been performed, We have to use checkout command to discard all the changes

git checkout -- <file>

Git Alias:

Git, allows us to create alias for long command

For example, the command to get the log history of all the commits is

git log --oneline --graph --decorate –all

To create an alias for above command. Execute the following command

git config --global alias.hist "log --oneline --graph --decorate --all" Where hist is a alias for “log --oneline --graph --decorate --all”

So now when you want to get the history for all the commits you can simply run git hist

Ignoring unwanted files:

Sometimes you may not want all the files in a given working directory to be committed to repository and do to just that we may first have to create a file name. gitignore and add the file name or expression to exclude the files and commit that file in git repository

For example if you don’t want to git to track changes of your log files then you can create a. gitignore file in the working directory as follows

cat > .gitignore

\*.log

Then add and commit the .gitignore file to repository.

”

To get the information on commits done on a given repository, we can use git log or git show command.

git log will show all the commit ids, author, date and commit message. And git show command will commit ids, author, date and commit message plus the diff on the files modified