Contents

Introduction

Git vs GitHub vs GitLab

Git – another tool developed by Linus Torvalds. Distributed version control system.
GitHub – a cloud based git repository hosting service.
GitLab – a web based DevOps life cycle tool, provides git repository manager, wiki, issue tracking and CI/CD pipeline. Based on collaborators role, different authentication levels can be assigned.

Ways to use Git

- Command line, Code editors, Git GUI clients

Git workflow

A practical example.

- We modify one or more files and this reaches a stage where we want to record it, we commit the changes into repository. Commit is like taking a snapshot of the project.
- Git has a special intermediate step – called staging area (or index). This tells what will be going in, in next commit.

  Project director is currently empty.
  a. We ***add some files*** (ex: file1, file2) and wanted to ***record*** this. So we run –
     - git add file1 file2
     - Some valid commands (git add *.txt), (git add .)#entire directory
  b. Once staged and not committed, you decided to ***remove*** some file (ex:file1). You run the command:
     - Git rm --cached file1
  c. At any point you can run a git status to see the ***current status*** of project directory.
  d. Everything looks good, so we want to ***make this permanent***. We run –
     - git commit -m "< meaning full message>"

     Note: Even if we commit, the staging area will have the same content (like even if we move product to PROD, stage will have the same latest release).
  e. We happen to modify one of the file (ex: file1), and wanted to stage and then commit. So -
     - git add file1 ##stage file.
     - git commit -m "<reason for change>"
  f. We realized we don't need file2. So we ***deleted*** the file from project folder. Then stage it with –
     - git add file2 ##note: we are using 'add' option. Git know that file is not present and has to delete from stage.
     - Following git add, we run git commit.
     - Note: if this was an ***accidental delete***, we can ***revert*** the change by git restore --staged <filename>

g. I want to rename a file (file1) to something else (somfile.cpp). So I rename it in local folder. Now for git this is two operation; delete file1 and add somefile.cpp. So the commands are:
git add file1
git add somefile.cpp
Note: Git will intelligently understand this is a file rename. You will see this detail with git status command.
SO if it is just a renaming, you can do this directly with git commands. You can use git mv file1 somefile.cpp (thus save the task of renaming file directly in the folder and executing git add twice).
Finally do a git commit "message"

Note: each commit consist of ID, Message, Date/time/Author and Complete snapshot. It stores the entire content file, but it uses some advanced compressing technique and doesn't store duplicate content.

Should we always stage before commit? Can we skip stage and commit?

Yes. You use $ git commit -am "a meaningful message". -a(all) -m(modified).

What is in my stage?

To find the content, execute $ git ls-files

Installing Git

Having sudo privilege you could the apt package to install Git.

Update packages

$ sudo apt update

Install Git

$ sudo apt install git

Git version

Check the version to ensure Git is successfully installed.
$ git --version

Working with Git

There are two different config files to work with. SYSTEM (applied over all user), GLOBAL (applied over all repositories of current user), LOCAL (applied to the current repository).

- Global configuration file is ~/.gitconfig.
- Local configuration file is project folder/.git/config.

Renjith

Install Git, GIT GUI in Debian

Configuration details include Name, email, default editor, line editor.

Global configuration

Here we will configure your Git commit username and e-mail.

```
$ git config --global user.name "YOUR NAME"
$ git config --global user.email "YOUR_EMAIL@PROVIDER.com"
$ git config --global core.editor "code – wait" ##note: setting default editor to VSCode.
$ git config --global diff.tool vscode ##vscode is will be used for diff
$ git config --global difftool.vscode.cmd "code –wait –diff $LOCAL $REMOTE"##We are providing the command to invoke the diff tool (vscode) with 'code --wait' and this is applicable to both Local and Remote repository. $LOCAL and $REMOTE are place holders that will be automatically filled by Git. Then run the git difftool command to launch diff tool.
$ git config --global --global url."https://YOUR_USER_NAME@github.com".username YOUR_USER_NAME
```

Update global config

To update the global config you run command like

View global configuration

```
$ git config --global -e ##note: opens file in default editor
```

NOTE: See the flag --global in all these command. This says we are setting properties ***not specific to a repository*** but applied to all the repositories.

Let's send some files to remote repository

Initialize a repository

```
$ git init YOUR_REPOSITORY_NAME
```

Go inside the repository you created and run a ls -la. You will see a .git folder and inside it file named config. Config file help us to know where is the origin, and where the content will be push to.

Update local config

To update the project specific config file (ex: we are updating the ***URL*** under the section [***remote*** "***origin***"]), we run below command from the project folder:

```
$ git config remote.origin.url https://github.com/YOUR_REPO_PATH.git
```

Renjith

Install Git, GIT GUI in Debian

Create a file in local repository

You can create a file or even a project inside the newly created repository.

```
$ touch README.MD
```

Check local repository status

You can check the status of your newly created repository

```
$ git status
```

Add file to local repository

Add the project/file you created into repository.

```
$ git add README.MD
```

NOTE: This won't right away add the files into repository. Instead 'git add' accumulate files for the 'commit' operation.

Clone an existing repository

Cloning is pulling a full copy of all the files from existing remote repository into local. You can do it with:

```
$ git clone "https://github.com/YOUR_USER_NAME/your_repository.git"
```

This downloads everything from the repository and creates the .git folder and its contents.

Commit the added files into local repository

```
$ git commit -m "a short message"
```

NOTE: Even commit will NOT send your files into the remote server (ex: github). The files are still in your local computer.

Push changes to remote repository

Login into Github, create a new repository to which we will be pushing our change to. Once the repository is created, copy the URL and come back to the terminal.

Add remote repository path into local repository configuration

```
$ git remote add origin YOUR_HTTPS_GITHUB_REPOSITORY_NAME
```

NOTE: What we did now is change the origin and set the URL. We did not push anything into remote.

Push changes to remote repository

To push the change to remote repository (github), create an access token or an SSH key. You need to go to settings page of Github and follow the instructions you see there.
Copy the token (DO NOT SHARE TOKEN)

Renjith

$ git push origin master "a short message"
It will now ask your user name. Once provided, it will ask password. Paste the token you created above step.

Refresh the github repository and you will see the files are in the repository.

Delete a file from project folder and local repository

$ git rm file1

Restore an earlier version of file

By default git will restore from the next env to the previous area. Ex: if the file is not in working directory, it will be restored from staging area…. And if the file is not present in staging area, it will be restored from remote repository.
$ git restore --source=HEAD~1 file1##restore from 1 previous step.

Branches

To check which branch you are on:
$ git branch ##shows which branch we are using
$ git branch -a ## lists all branches of the file.

Create new branch

Example create a branch named minor_enhancement.
$ git branch minor_enhancement

Switch to a branch

$ git checkout <branch name>

To merge changes into another branch

Example: merging minor_enhancement branch into master

- $ git checkout master
- $ git merge minor_enhancement

Additional notes:

Commit history

To see the history of commits made into the repository can be viewed with
$ git log --oneline

Renjith

Sample output:
4c3be83 (HEAD -> master) git ignore
7af0155 Update README.md
Note: Here 4c3be83, 7af0155 are HASH_VALUE

UNDO last commit

, $ git revert git revert YOUR_HASH_VALUE

File difference with last commit

$ git diff YOUR_FILE_NAME

View the commit

$ git show HEAD~<number of steps you want to go back>:<filename>
Ex:
$ git show HEAD~4:.gitignore##show 4th previous checked-in file details.
$ git show HEAD will show the last committed change.
$ git ls-tree HEAD~1 ##files are represented using blobs and directories are rep in tree.
$ git ls-tree <few characters of unique identifier>

Git Ignore

You can add files to a special file named ".gitignore". Any files/expression in this file will be ignored. Note that git ignore will not ignore the changes for any files that are already in repositotry. It will ignore files that comes after defining in gitignore. Solution is to remove the file from stage. Example :

- We accidently included 'logs' folder in git repository and committed. Later we included the logs/ in .gitignore.
- As mentioned earlier this will not ignore tracking /log folder, because git is already tracking it.
- So we will remove this folder from git stage (note: we are not removing it from our project folder, but only from staging area), with the command:
    - git rm --cached -r logs/
      --cached is only to remove from stage. -r is to recursive.
    - git ls-files ## check the files in stage.
    - git status ## check the status
    - git commit -m "to delete accidently committed file that is infact not needed".

Caching password

Incase you want to cache your password and set a timeout:

```
$ git config --global credential.helper "cache --timeout=28800"
```

Note: 28800 represent 8 hours in minute. You can provide your own minutes.

SSH logging in

You can login SSH instead of username/password. Go to the settings page in Github and follow the steps mentioned under "SSH and GPG Keys". Now go to the repository, select Code and then SSH and copy the address. Come back to terminal into the project folder and run below command -

```
$ git config remote.origin.url git@github.com:YOUR_REPOSITORY.git
```

GIT GUI

You now noticed we can do everything even without the GUI. Git GUI is an optional extension that helps most of the operations done graphically. Git Kraken is the another most popular Git gui.

Install Git GUI

```
$ sudo apt-get install git-gui
```

Go to already created repository or create one by git init command. Then run `$ git gui&`

Select Open Existing Repository (other choices are Create New Repository Clone Existing Repository)

Once the UI is opened :-
Rescan – Reloads local changes.
Stage Changed – Pick files that are locally changed and is ready to commit. Add necessary comments.
Commit – Local check-in of files. Note that at this point files won't go to remote repository.
Push – Push files from your local computer to remote repository (Github).

Standard practice

Git clone

If you are cloning new repository (or delete the local project folder and clone the project again)

```
$ git config remote.origin.url https://github.com/YOUR_REPO_PATH.git
$ git clone https://git-username[:git-password]@github.com/YOUR_REPO_PATH.git
```

Get latest code

Before you start work ensure you are taking latest code by executing below command:-
Get latest code and reset local code:

```
$ git fetch origin
$ git reset --hard origin/[your_tab/branch/commit_id/master]
```

Renjith

Retain local changes

If you want to retain local changes while need to work on latest code:

```
$ git fetch origin
$ git status
$ git pull
```

Reference: https://git-scm.com/book/en/v2