GIT

Designed by Linus Torvalds in 2005 for development of the Linux kernel, with other kernel developers contributing to its initial development.

GIT is a Command Line Tool

Why Command Line?

NOTE: every Git directory on every computer is a full-fledged repository with complete history and full version-tracking abilities, independent of network access or a central server

Repository contains files, history and config managed by GIT

Miscellaneous:

Name GIT?

The man page describes Git as "the stupid content tracker". The read-me file of the source code elaborates further:

"git" can mean anything, depending on your mood.

* Random three-letter combination that is pronounceable, and not actually used by any common UNIX command.
* "Global information tracker": you're in a good mood, and it actually works for you.
* "Goddamn idiotic truckload of sh\*t": when it breaks.

The source code for Git refers to the program as, "the information manager from hell."

Git means “unpleasant person” in British English.

Installation and configuration

Installation: Separate document has been created. Follow it

Configuration:

We need to configure GIT with some basic information such as username and email. This is done as follows.

>> git config - -global user.name “Salman Ali”

>> git config - -global user.email “SalmanAli@gmail.com”

>> git config - -global - -list

NOTE:

1. We can run these config commands any where in our system. These are global commands. These are irrespective of any repository
2. Settings can be global level or user level. Here we have done at global level
3. GIT will create a config file(simple text file) in our user directory that is called “.gitconfig”
4. You can print to see the contents of config file using cat command or open file in text editor(like notepad) in the file explorer. I am using cat command

>> cat ~/.gitconfig

This is git’s user-based config file

Working locally:

Commands

1. Help

Getting general help:

>> git help

Getting help about a specific Git command:

Syntax: git help command

>> git help config

In Linux: this will open man page in the same terminal

In Windows: this will open a webpage in default browser

First Commit

1. Start with README.md file. Md means markdown
2. Do the process 2 times and third time do short method of direct commit using “git commit -am “message”

NOTE: Use the -a parameter with the git commit command to directly commit newly modified tracked files. Warning: Only do this for small changes. Tracked files are files that have been previously added to Git (committed or staged).

Unstage and restore

1. Do the process 2 times again with new index.html along with Readme.

* First time do regular
* sec time show “git add .” to add all the modified files to staging.
* Third time take only readme, add to staging and show how to get back from staging using “git restore –staged file-name” and how to undo the changes done in working directory using “git restore file-name” to the last committed version

History

1. To get history of commits for a repo use “git log”
2. To get one liner use “git log - -oneline”, “git log –oneline –graph”

Back-out already committed files (with git and w/o git)

1. Take example of log file. Commit it and use “git rm file-name” and then “git commit”. Like commit is necessary after creating/modifying a file, similarly, commit is necessary after deleting a file to affect the Git repo

Moving files: (with git and w/o git)

1. Use “git mv index.html web/” to demonstrate moving already committed file to subfolder or any other folder

Ignoring files in our project

1. Demo using “.gitignore” file and a log file
2. Use \*.log inside .gitignore file
3. Use web/temp.txt inside .gitignore file in second line to prove that any file in any location inside project can be ignored by using absolute path
4. Delete ignored file w/o git and prove that git is not tracking those ignored files.

GOING REMOTE

1. Create github account with e-mail and generate ssh keys in your system and add those keys in your github account

ssh-keygen -t rsa -C “your-email”

ssh -T [git@github.com](mailto:git@github.com)

NOTE: ssh keys are tied to your computer

1. First push

After matching default branches, Create repo in github and issue the following commands

git remote add origin https://github.com/SalmanAli-W/First-Git-demo.git

git remote -v

git push -u origin master

-u option is required for the first time, -u will establish an upstream link between our local repo and remote repo

1. After few commits with branch as master and then want to change to main,

git branch -m main

1. git init -b main sec\_demo
2. git config - -global init.defaultBranch main
3. git clone git@github.com:SalmanAli-W/First-Git-demo.git

to get files from github to local git

1. git pull origin master

Doing pull before push is good practice. First we should get the latest code from remote repo and then push our changes

How to Access Github repo between systems

1. Github repo should be public
2. Generate SSH keys in “Target machine” and add these keys into the github account
3. Create empty local git repo
4. Done once, add github to known hosts using “git remote add …” command
5. Use “git clone ssh-clone-code-from-github-repo” found in code drop down in the github repo which you want to clone