What is git

Git is most commonly used distributed version control system that helps you:

a. To make record of versions so you can revert back when needed
b. Makes collaboration easy

Version Control System (VCS)

Suppose you are working on a project. Your daily tasks would be:

a. Create things (write code)
b. Save things (save the code that you created)
c. Edit things (code is not working, do the changes)
d. Save the things again

When saving things again and again, we often need answers to the questions like

- when I did the last change
- why I did the changes
- what were those changes etc.

Ques: Should I use git if I am the only person working on the project?

Ans: Once you call it a project, it is worth using VCS. I would love to see a pharmacist ask: "Should I store medications in an organized way or just throw them all in a drawer? Is it worth the effort?

This is where VCS comes into picture. VCS tracks things for you so that you can track the changes. Since VCS has the track, it can help us to revert those changes or merge those changes (which are features of VCS)

Ques: Why use VCS instead of maintaining the copy of folder and saving it with timestamp?

Ans: a. This is because if your project is of 200 GB and you made changes in 5 of the files, you will have to copy entire 200 GB project just for those 5 file changes

b. After a month, you may forget the name of those 5 files in which changes were made

c. Suppose you just copied those changed files and renamed them with timestamp to solve problem a and b, undoing a change in those 5 files will be hard to remember. That's why, we use VCS

Ques: What does distributed means here?

Ans: Apart from having remote repo located in a server, git provides a local repo which can be stored on the local system of people working on a same task

Ques: If git is distributed VCS, then do we have any other type of VCS too?

Ans: Yes, VCS are of two types:

a. Distributed VCS
b. Centralized VCS

Ques: Is there any other VCS tool apart from git that is available in market?

Ans: There are other VCS tools too like SVN, Perforce, ClearCase etc

Ques: What is the difference between SVN and git?

Ans: The major difference between SVN and git is its nature. SVN is centralized VCS while git is distributed VCS

Ques: What is difference between git, github and gitlab?

Ans: Although they may look like same, but git, github and gitLab are different

Git is a VCS tool which is used to track changes. Github and GitLab are service providers that manage your git repos so that people can access your code from anywhere. They offer all the functionality of VCS while adding their own features

---

Git has three states:

a. Working tree
b. Staging area
c. Repository (HEAD)

Working tree is where the actual files and folders are stored on the local machine. Any change that is made in working tree is considered 'untracked' until the changes are made to track by running 'git add' command. Staging area is an intermediate area where we can prepare changes for our next commit. Any changes made in the file in staging area will move the file to modified area. Modified files can be moved back to staged area by running "git add filename" command. Files that are committed from staging area are moved to the committed area. Repo is where all commits are stored. HEAD is the pointer to the latest commit in that branch

Although we can directly move the changes from working tree to repo without using staging area, but this is not considered as a good practice

Using git

In order to use git, make sure you have git installed on your system by running

`$ git --version`

Now, that you have git installed on your system, let's see how we can configure the git environment. Git comes with a tool called `git config` that is used to set configuration variables. These variables can be stored at 3 different places:

a. Local: It is config file in the git directory (.git/config) of repo that you are currently using
b. Global: It will be located at $home/.gitconfig location and contains values specific w.r.t the user

Passing - - global will read and write from this file

c. System: It will be located [path]/etc/gitconfig and contains values applied to every user on the system and all their repo

   Passing - - system will read and write from this file

Each level overrides values present in previous level. So local changes will override global changes will override system changes

To see all your settings and from where they are coming from, use

`$ git config - -list - -show-origin`

These config files can be used to make custom config messages, aliases and for other useful changes that we need to configure our git with

Our Identity

We can set our username and email address using command given below. It is important because every git commit uses this information

`$ git config --global user.name "<username>"`

`$ git config --global user.email "<useremail>"`

Default Editor

Another useful command that we should configure is the default editor for git. Here, notepad++ is configured to be used as default editor for git

`$ git config --global core.editor "'C:/Program Files/Notepad++/notepad++.exe' -multiInst -notabbar -nosession -noPlugin"`

Note: To start the notepad++, use command `$ start notepad++ <fileName>`

Default branch name

By default, git will always create a master branch when we create a new repo with git init command

We can overwrite this setting by using command `$ git config --global init.defaultBranch <branchName>`

How to get config values?

a. If you want to check your configuration settings, you can use command:

   `$ git config --list`

b. You can also use command `$ git config <key>` to get specific value

If you are confused about the value, you can always use the command `$ git config –show-origin` to get the resource of the value

Getting a git repo

We can obtain a git repo in two ways:

a. Making a local directory that is currently under VCS into git repo
b. Cloning an existing git repo from elsewhere

A. Initializing a repo in an existing directory

If a directory is not under version control and you want to start controlling it with git, then go to that project's directory and type the command given below:

`$ git init`

This will create a new subdirectory named .git that contains all necessary files. This command also creates a new main branch

B. Cloning an existing repo:

If you want to get a copy of an existing git repo, then use the command below

`$ git clone <repo-link>`

This will create a directory named gitInternal, initialize a .git subdirectory inside it and pull down all the data available in that repo and checks out a working copy of the latest version

Ques: Is it possible to change the name of directory getting created during clone to something else?

Ans: Yes, you can change the name of the cloned repo by passing the name of the directory in which you want to clone the repo using command given below

```
$ git clone <repo-that-needs-to-be-cloned> <directoryName>
```

Note:

a. If directoryName not already exists, then git will create an empty directory and then clone the required repo
b. Git will throw error if the directoryName already contains a .git file

Ques: Is there a way to get the URL of the repo that you used to clone the repo?

Ans: Yes, by running the command `$ git config --get remote.origin.url`. Note: This command will return none if the repo was created using git init command

There are two major protocols that are used to clone a repo. One is HTTPS and another one is SSH.

Recording changes to the repo

Once a git repo is created, the next thing that you want to do is start making changes into the repo each time the project reaches a state you want to record. In git, each file in repo can be in one of the three states:

a. Tracked
b. Untracked
c. Ignored

Untracked basically means that git see a file that is not available in the last snapshot (commit)

To determine which file are in which state, use the command `$ git status`

Clean working directory means none of your tracked files are modified and there is no untracked file. This command also informs you the branch that you are working on

Let's suppose now you have added a few files here. You can see your untracked files by running the `$ git status`

Note: If your repo contains an empty directory, git status will not track that directory. This is because git is a content tracker and empty directories are not content

Note: If you don't want to get all the details, command `$ git status - - short` or `$ git status -s` can be useful. It uses flags to show status of the files inside the repo. Short status flags are:

?? - untracked files

M – Modified files

A- Files added to stage

D – Deleted files

There are two columns to the output — the lefthand column indicates the status of the staging area and the right-hand column indicates the status of the working tree

Tracking new files

To begin tracking the files, run the command `$ git add <filename>`

To add more than one file at the same time, use `$ git add - - all` or `$ git add .`

Note: `$ git add -A` is shorthand command for `git add - - all`

At this stage, we can do two things:

a. Rollback the staged changes
b. Modify the staged changes
c. Commit the staged changes

If you commit at this point, the version of file at the time you ran the command git add will be added in the subsequential historical snapshot.

If you want to roll back the changes, simply run command `$ git restore - -staged <fileName>`

This is the difference between **git add .** and **git add ***. Read more about them here

Ques: What does git add –u command does?

Ans: git add –u or git add –update command is used to stage all the modified, deleted or renamed files that are already been tracked by git. Note: This command will not stage any untracked file.

For ex: There were a few files in the repo which were already being tracked by git

Now, we added a new file named file1 and modified file project that was already being tracked by git. When running git add –u command, the file project was added to the staging area, but file1 still remain untracked by git

Ignoring files (.gitignore file)

More about gitignore can be found here and here

There will be few files and directories that we do not want to be part of git commit. It can be log files, temp files etc. In order to ignore those files or directories, create a file named **.gitignore** in the root repo and add the name of the files or directories that you want to ignore. Let's create a project.log file that we do not want to track

In order to ignore the project.log file from being tracked, create .gitignore file and add project.log file to it. Run git status again. You will see that project.log file will no longer be part of git status output