Git

A free and open source
distributed version control system

What does Git do?

Git allows group of people to work on the same documents at the same time, and without stepping on each other's toes. This is called distributed **V**ersion **C**ontrol and source code management **S**ystem (VCS).

Git handles from small to large projects with speed and efficiency. Overall, Git is a free software distributed under the terms of the GNU General Public License version 2.

Benefits of Version Control System

VCS has following benefits:

1. It allows developers to work simultaneously,
2. It does not allow overwriting each other's changes and,
3. It maintains a history of every version.

In following slides, we will learn about the advantages and some of the basic commands of Git. Let's learn it.

Advantages of Git

Here is the list of advantages that is why Git becomes so popular. Because,

1. it is Free and Open Source,
2. it is Fast and small,
3. it provides Implicit backup,
4. it is Secured and,
5. it has Easier branching.

Basic Git commands

Now, after getting through advantages, let's go through some of its commands that we use generally. Here is the list:

1. git init
2. git status
3. git add
4. git commit
5. git log
6. git remote
7. git push
8. git pull
9. git diff
10. git reset
11. git checkout
12. git branch and
13. git merge

git init

Use to initialize an empty Git repository located in your local system.

git status

Use to see the current status of the project.

Often, this command is used to see the progress of the project and to check how is things going on. Lest, something has happened erroneously or unknowingly and we have no knowledge about this. So, this command gives us the update of the repository like what file(s) has been added, removed or edited.

git add <file name>

Use to add file(s) to the staging area to start tracking changes made to these files. At staging area, we can add or remove files before storing them in our repository.

We can specify particular file format to stage or add all files at once to the staging area by using last two commands.

Other usage of this commands is listed below:

git add '*.txt'

or,

git add .

git commit

This command is used to store the staged changes in our repository. Here, you can add a message describing what you've changed.

Usage:

```
git commit -m "type your message"
```

git log

This command is used to see a report of changes committed to the repository.

Usage:

```
git log
git log --oneline
git log --oneline –graph
```

and much more.

git remote add <option>

This command is used to push our local repository to a remote repository on GitHub server.

For this, we have to first create a new empty GitHub repository. Its path may look like - “https://github.com/<repo folder name>/<repo name>.git

For example:

Let us say, <repo folder name> is “gitrepo” and <repo name> is “my_repo”, then the complete command would be as below:

```
$ git remote add origin https://github.com/gitrepo/my_repo.git
```

git push

The push command tells Git where to put our commits when we need to do so.

Here, the name of our remote is ‘origin’ and the default local branch name is ‘mybranch’. The -u tells Git to remember the parameters, so that next time we can simply run ‘git push’ and Git will know what to do.

Usage:

git push -u origin mybranch

On success, following message will appear:

“Branch master set up to track remote branch master from origin.”

git pull

The pull command helps us to update our local repo with the changes made by other people on our GitHub repository. So, to check for changes and pull down any new changes, we can use this command.

Usage:

git pull

git diff head

Now, we want to see the difference between our last and current commits.

Usage:

git diff head

git diff --staged

This command shows differences between two files at staging area.

git reset <file name>

To unstage our last commit, we use ‘reset’ command.

git checkout --<branch name>

Files can be changed back to how they were at the last commit by using this command.

Usage:

git checkout <branch name>

git branch

To create a copy of the master branch, we use this command. Then when we are done with the new changes, we can merge this copied branch back into its main ‘master’ branch.

Usage:

git branch clean_up

Branch switching

Now, we have two local branches: a main branch named ‘master’ and our new branch named ‘clean_up’. We can switch branches using following command.

Usage:

$ git checkout clean_up

[Removing unwanted files]
$ git rm <file name>

Since, we have merged the changes into the master branch. Now, we do not need files in our local branch. Therefore, we can remove these files. This not only remove the actual files from disk, but also stage the removal of the files for us. For this, we use following command:

$ git rm *.txt

We can use ‘wildcard’ character i.e. *, to include all relevant files.

[Committing the changes]
$ git commit -m <message>

Since, we have removed all the files and now need to commit these changes into the master branch. So, the command line will look like –

Usage:

$ git commit -m “Remove all the cats”

[Checking out]
$ git checkout master

We need to switch back to ‘master’ branch to merge changes done in other branch i.e. clean_up.

[Merging]
$ git merge <branch name>

Now, we need to merge changes into the “master” branch. Since, we are already in “master” branch. Therefore, we just need to merge “clean_up” branch into “master” branch. Here is the merging command:

Usage:

$ git merge clean_up

[Cleaning up the things]
$ git branch -d <branch name>

In our last attempt, we have merged the branch successfully. Now, we are done with the “clean_up” branch we do not need this anymore. So, the command to delete a branch is given below:

$ git branch -d clean_up

[Final thing to do]
$ git push

Now, everything has been cleaned up. Final step is to push everything on to our remote repository. Use this command and we are done. That’s it.

Happy Learning!