 **Quick Tour Part-1 - Shivam Yadav, IT, Fr.CRCE, Mumbai**

1. **Introduction**

If you are not so familiar with the so called GIT CLI(Command Line Interface) and don’t like those haunting black screens then SourceTree is a better option for you guys which provides a rich GUI for implementing git operations without writing a single Line of Command and you get a top up of Graphical Representation of all your commit history and other things.

So this tutorial does not cover SourceTree and we will be covering it surely in our upcoming blogs.

Git is a complete word with no abbreviation as you all might be wondering about GIT as an acronym which it is not and it is defined as the

“*Distributed version control system* “

Before coming to the CLI we need to look at some regularly used terminologies of Git.

Version Control - It enables teams to manage a collection of files in an agile way.

Commits - A commit is a snapshot of the entire project.

Git Repository - It contains the project history as commits.

1. **Installation**

Going to this link , u will get a download option for your OS specific version , just try to install the latest stable version and you are already after installing it.

We are going to use Git Bash CLI for our tutorial which is a great command line interface as it supports Windows and Linux Commands , thereby allowing the multiple varieties of users to implement git.

1. **Syntax**

git [command] [--flags] [arguments]

Note :

1. flags – options or switches

2. command – command name

3. arguments – are the additional information given In order to use a command for particular use.

Example :

git status

The above command is used to check the status of the commit history of the directory or a git repository.

1. **Commands**

Commands help in achieving our task with the help of it we implement Git operations. And there are many commands in GIT . If we want to read any command’s details or summary we can use –

git help [command name]

or simply

git

Reading Help -

1. -f or –flag : Changes the command behavior
2. |: It is used to denote “*OR*”
3. [optional]
4. <placeholder>
5. (grouping)
6. -- : Disambiguates the command
7. … : Multiple occurrences

Now here are some commands which are generally used once when u setup Git to your workplace.

# To update Git Version

git update-git-for-windows

# To set username and email

The git config command is used to set a author for repository level , system level or global level.

git config [-- local | -- global | -- system] <key> [<value>]

1. system : This flag applies to every repository for all users on computer
2. global : It applies for every repository that you use in your computer.
3. No flag or local : Applies only to current repository

Following two steps are followed to set author :

git config -- global user.name “John Doe”

git config -- global user.email “john.doe@abc.com”

# Reading username and email

git config user.name

git config user.email

# Setting GIT’s default editor

Git config -- global core.editor vscode

**5. Locations of GIT**

**Working Tree -> Staging Area -> Local Repository - - - - - > Remote Repository**

Project Repository (Local Computer) GitHub,Bitbucket

Fig 1

1. Working Directory : A single commit’s directories and files

2. Staging Area : (Index) Files that are planned for next commit

3. Local Repository : Contains the commits of the project

4. Remote Repository : It is located in data centers, clouds. It acts a source of truth .

**6. Creating Local Repositories**

cd <name\_of\_the\_directory\_whom\_you\_want\_to\_create\_a\_local\_repository>

git init

Note : The git init command initializes an empty local repository (although you may be having some project files or folders to it , as they are in folders but not in local repository because for that we need to add and commit to have them in our desired local repository) with a hidden file named .git . The .git file has your above mentioned staging area and local repository.

U can find .git file in Git bash Command Line Interface(CLI) by command .

ls -a

**7. Commit to a Local Repository**

Steps to be followed for commit are (Assuming that you have created your local repository by the help of above given command) –

Step 1 : Check status of new files and untracked or modified files.

git status

This will provide the working tree and staging area details.

Now lets say we have file1.txt and demo file in our directory which is supposed to be there in local repo and it is showing in red accent to add and commit in order to have them in our local repo.

So we need to first add them to the local repo.

(Note : Untracked or Modified files are the files which were added to the local directory after the git init command was implemented.)

Step 2 : Add the untracked files and modified files

git add file1.txt

If we want to add all untracked files at one go we can use

git add .

Now after adding we get a reminder in green accent which says to commit so and so files to your local repo.

Step 3 : Commit to the local repo

git commit -m “file1.txt added ”

( Note : -m : It basically denotes message and “message” : The quotes contains the message of commit which will help other team mates or collaborators or the author himself for the history and reference. )

Now lets say we have done couple of commits to our local repo and we want to have a look at them and take some insights from them .

Step 4 : Displaying log / history of commits

git log

This will generate a history of commits by user.name and user.email(author) with the timestamp.

If we want to have a condensed form of information of all of the commits with just their branch name and (-m) message associated.

git log – oneline

**8. Remote Repository**

By referring the Fig 1 at pg.no. 4, we have reached till local repository , where we have added all our files to it but here arises a concern of backup as everything is on the local computer and the system failure, etc. can cause data destruction/loss so the option of Remote Repository (*Bare Repository – a local repo which is not used by anyone*) helps us in backing up and becoming a source of truth for the local repo files. It also enables us for developing projects and implementing task remotely with great bunch of friends or colleagues.

By Remote Repository I mean, we try to use the services of hosting organizations such as Bitbucket and GitHub where they allow us to store our files safe and secure at the cloud with an options of open sourcing i.e. making our files public thereby increasing the reach of our content(codes, ideas, projects and other types of contents are welcomed here) and this helps in contributing to the society of talented innovators, tech and non -tech enthusiasts. And we can even create our own private repos using the hosting options such as GitHub, Bitbucket, etc.

Now lets see how to make a remote repository,

So we have two paths from here to go ahead ,

Path 1. -> If we do not have a local repository

Path 2. -> If we have our local repository

So for path 1 we can either make one local repository by following pages( 4 to 6) or we can clone a repository .

Lets see first what is Cloning and some other terminologies related to Remote Repositories,

Cloning – Process of creating a local copy of a pre-existing remote repository and the local copy created with these is called as a Clone.

Origin – Remote Repository URL(with a .git extension)

How to clone ? To our local working directory

git clone “URL\_of\_the\_github\_or\_bitbucket\_repo.git”

(Note: Here we aren’t modifying the pre-existing files present in existing remote repo , that is done through pulling the code or files from it and then adding and committing to it, all these are coming in the part 2 of this quick tour )

Branch – All commits belongs to a branch (master is the default branch)

**PUSH – It adds commits for a branch to remote repository ,used for backups of local commits and enables synchronization with the team of collaborators or developers.**

Now for path 2 , As we have a local repo and If we want to have that in our remote repo ,we need to add it to the remote repository (Pushing the local commits).

Follow the steps below in order to create a remote repo of your local git repository.

Step 1: Creating a repository on GitHub

You just need to visit the GitHub official website and login to it then there will be an option to create directory .

After this you just need to copy the remote repository URL which you have just created from the option named Clone repository (just copy the link given in it).

Step 2 : Go to your local git repository and run following commands in order.

git remote add origin <URL\_of\_remote\_repo\_you\_created)

To check the details of the remote repo which you just added…

git remote -v

Step 3 : Push the recent commits to the remote repo (to origin)

Syntax for Push Command

git push [-u] [<repository>] [<branch>]

Here, <repository> : shortcut name (i.e. origin) or URL

-u : This is basically used to enable track option (--set—upstream) .Used for tracking the local and remote master branches.[for synchronization]

git push -u origin master

(Note : If you get errors after doing this regarding , unable to merge etc.

master -> master then follow these steps ….

1. Clone the newly created repo at github ..

git clone “Remote\_Repo\_URL”

2. After that cd to the newly cloned Remote Repository (local copy of remote repo just created in the directory where u cloned it) so just cd to that directory.

cd <Clone\_Repo>

3. After this add your complete set of files which you need to have in remote repo.(i.e. paste files to be pushed to the remote repo in this clone\_repo)

4. Now follow the steps below ….

git status

git add . (or git add <filename>)

git commit -m “commit message to the files you want to commit”

git push -u origin master

After this your local repo will be successfully backed up to the remote repository. And you can check logs, commit history , status of the pushed files and then untracked files which you wish to add can be pushed again in the above mentioned steps. i.e. first you will need to add them (git add) then (git commit) after that you can push them to your remote repo. (origin)

(Reference : Coursera)