**What is GIT?**

Git is a Distributed Version Control System (DVCS).

* Each user has a local copy of the complete history of the project which is known as a repository.
* Users can work offline.
* Can easily synchronize repositories.

Git is a free and open source software project meaning that the code that implements Git publicly available. No single company owns Git, and anyone can make contributions to improve it.

**Git Bash**Git bash is not a command, it is an interface provided by the Git software.



**Git Command**

**Basic Commands:-  
pwd**:Present Working Directory. It let us know at which directory we are currently on working.

**cd**: Change Directory. It help us to switch/change directory.  
Ex: cd G: 🡪 move to G Drive

**ls**: list command use to check all the repository and file inside the current repository.

**ls –a**: to check all the hidden repository and file inside the current repository.

**mkdir**: Make Directory. Ex: mkdir Chandan

**touch:** Make files. Ex: touch index.html

**touch .gitignore**: To create gitignore file. Write any file name with its extension inside gitignore file, which would help in hiding the required file from the project repository to others.

**rm <file>:** To remove the file completely from the repository.

**clear:** Clear all the previous command in the bash terminal.

**git status:** To check whether git has already been initialized or not in the current working directory or not.  
**git status -a:** Shorthand Command of git status. (Green M) indicate staging area whereas, (Red M) indicate working area.

**git init:** To initialize the current working repository as Git repository to move it to the Staging Area.

**git add <file>:** To move the required file to the Staging Area, so that GIT will able to track it and include what will be committed. Now as I commit to this file, it will automatically add to the local repository of GIT.  
**git add -A:** To move multiples files to the Staging Area at once, so that GIT will able to track it and include what will be committed. Now as I commit to this file, it will automatically add to the local repository of GIT.

**git rm --cached <file>:** To unstage the file that moved to the Staging Area for tracking.  
**git rm --cached –r <folder>:** To unstage the files of the folder that moved to the Staging Area for tracking.

**git commit:** To commit into the file. By commiting into the file, we can notify the GIT or show to the local repository about what changes has done on it, so that I will get a unique Id for this commit and by using this id I will get to know in future about what changes / commit I’ve done this time.   
Short Command to add git commit: **git commit –m**  
Example: git commit –m ‘First commit, index file added’.

**git log:** To check the commit on the file with the commit id.  
**git log --oneline:** Another command  
**git log --oneline -3**: To get only the first 3 commit out of many commits on the single file.

**git config user.name:** to set the username of GIT for the current working repository. This will set the only for the specific repository i.e. (locally).  
Example: git config user.name “Chandan”  
**git config --global user.name**: This will set globally for all git repository i.e.(Globally). By default global user name would be consider for all git repository. Although we can override this username for the specific repository using local.  
Example: git config --global user.name “Chandan Kumar”

**git config user.email:** to set the email of GIT for the current working repository. This will set the only for the specific repository local command as well.  
Example: git config user.email “ckking189@gmail.com”  
**git config --global user.email**: This will set globally for all git repository i.e.(Globally). By default global user name would be consider for all git repository. Although we can override this email for the specific repository using local command as well.  
Example: git config --global user.email “hirechandan@gmail.com”

**git remote**: To add local repository to online in Git Lab.

**git checkout <modified file>**: To get the previous code/content in the modified file.  
Ex: git checkout index.html  
**git checkout -f**: To get the previous code/content in the multiple modified file.   
Ex: git checkout –f

**git diff**: To compare the previous code with the current modified code in the modified file. It would only show the modification before added it to the stage. It would show nothing, if the working directory and the staging area be same.  
Ex: git diff  
**git diff --staged**: To compare the previous code with the current modified code in the modified file or say, to compare the staging area with the last commit. It will show after getting back the modified file into the staging area.

**git branch:**  To check at which branch currently i’m working on.

**git branch <branch name>**: By default our repository set to ‘master’ branch. To create new custom branches we’ve to follow this command.  
 Ex: git branch anybranchName

**git checkout <branch name>**: By default our repository set to ‘master’ branch as main branch. To switch our main branch to any other custom added branches, we’ve to follow this command.  
 Ex: git branch anybranchName

**git checkout <branch name>**: To create and switch our main branch to any other custom added branches **at once**, we’ve to follow this command.  
Ex: git checkout anybranchName

**git merge <branch name>**: By default our repository set to ‘master’ branch as main branch. To merge our required custom added branches to the main branch, we’ve to follow this command.  
 Ex: git merge anybranchName

**Note:**

* **Folder** (Windows) = **Directory** (Linux) = **Repository** (GIT)  
  A repository **contains all of your project's files and each file's revision history**. You can discuss and manage your project's work within the repository.
* **Commit** means to update any file of the repository. Its status show as “modified <file>” by checking repository status using git status. In shor commit = update / modify.
* To again move the modified file back to the staging area, then we can use this short command **git add .**  instead of git add <file> to get back file to the staging area.

**Git global setup fr GitHub:-**

**To get remote access of Github using Command line:**

* **git remote add origin** [**git@github.com:WebDevChandan/gitTutorial.git**](mailto:git@github.com:WebDevChandan/gitTutorial.git)  
  -- origin is the name of the URL i.e. Remote
* git remote or git remote -v  
  --To get the info about all connected remote. Ex: origin
* (Optional - Step) Now we’ve to get the read/write access of our gitHub account using SSH Keys from the main account Setting of GitHub.  
  In order to generate new SSH key using command line for our github account, follow this document:  
  <https://docs.github.com/en/authentication/connecting-to-github-with-ssh/generating-a-new-ssh-key-and-adding-it-to-the-ssh-agent>  
    
  Now in order to add our newly generated SSH key to your Github account, follow this document:  
  <https://docs.github.com/en/authentication/connecting-to-github-with-ssh/adding-a-new-ssh-key-to-your-github-account>
* **git push -u origin master**  
  -- To push, the repository to the github
* **git push**-- To push, the updated local repository to the github
* **git pull** then, **git push** (In other case)-- To push, the updated local repository to the github

**Example of pushing local repository to Public at Github:**

Let’s take an example of Text To Speech repository consisting index.html, CSS & JS files and folders respectively inside in it.

**Important Note** **Before pushing repo**:-

* By default, branch of Git is ‘master’ branch whereas by default, branch of github repository is ‘main’ branch. So, make sure to keep both repository have the same branch.
* Make sure directory location is at the root folder in order to push all the files to Github.
* Make sure you’re connected with phone’s hotspot not with college wifi otherwise it would create a connectivity problem. Because college wifi uses a proxy server to access the internet.
* SSH Key for the required repository:
* *https://github.com/WebDevChandan/text-speech-converter.git*

Steps:-

1. git init
2. git add –A
3. git commit –m ‘initial commit’
4. Make sure both branch of git environment and github should same by using (*git branch*). But my currently my default branch of git environment is ‘master’ branch where github has ‘main’ branch.  
   So, to switch the default branch (\*master) of git environment to (\*main) branch. We’ll use (**git branch -M main**)
5. git remote add origin [*https://github.com/WebDevChandan/text-speech-converter.git*](https://github.com/WebDevChandan/text-speech-converter.git)
6. git push –u origin main

**#Error Solutions**: (if got the following **errors**):-

1. fatal: unable to access 'https://github.com/WebDevChandan/text-speech-converter.git/': Could not resolve host: github.com  
    **Solution:**This error message seems to indicate a connectivity problem. If having any antivirus software installed other than the default antivirus that ships with Windows might cause this error.

Additionally, if are you using a proxy server to access the internet, might cause this error. Git requires special configuration to work behind proxy servers.

1. ! [rejected] main -> main (fetch first) error: failed to push some refs to 'https://github.com/WebDevChandan/crudop  
     
   **Solution:**Write this Code:- (Make sure branch should be same)  
     
   git fetch origin main:tmp

git rebase tmp

git push origin HEAD:main

git branch -D tmp

Note:

* In order to check all the commits of the respective branches, simply use url like this in the github dashboard:   
  <https://github.com/WebDevChandan/gitTutorial/commits/master>

- - origin/commit/branchname

* To get the clone of any repository across GitHub:  
  git HTTPSURL AnyFolderName  
  Ex: git clone https://github.com/MainakRepositor/TodoApp.git CloneRepo

**Git global setup for GitLab**

git config --global user.name "Chandan Kumar"

git config --global user.email "ckking189@gmail.com"

**Create a new repository**

git clone https://gitlab.com/WebDevChandan/googlekeepclone.git

cd googlekeepclone

git switch -c main

touch README.md

git add README.md

git commit -m "add README"

git push -u origin main

**Push an existing folder**

cd existing\_folder

git init --initial-branch=main

git remote add origin https://gitlab.com/WebDevChandan/googlekeepclone.git

git add .

git commit -m "Initial commit"

git push -u origin main

**Push an existing Git repository**

cd existing\_repo

git remote rename origin old-origin

git remote add origin https://gitlab.com/WebDevChandan/googlekeepclone.git

git push -u origin --all

git push -u origin --tags