**Git:**

-it is a version control System

-keep track of changing the code in repository

-we can code collaborately with different systems

Basic commands:

|  |  |
| --- | --- |
| Command | Description |
| git --version | it helps us to know the current version of git installed in the system |

What is Git:

🡪Git is a Popular Control System developed by Linus Torvalds in 2005, and has been maintained by Junio Hamano since

Used for:

🡪Track changes

🡪who makes changes to the code

🡪Collaborative Working

What GIT can do ??

🡪manage projects using Repositories

🡪Clone a project and Save as a local Copy

🡪Control and Track the repository using staging and commenting

🡪 **Branch** and **Merge** to allow for work on different parts and versions of a project

**🡪 Pull** the latest version of the project to a local copy

**🡪Push** local updates to the main project

Why git?? Over 70% of developers use git, developers can work anywhere in world ,developers can see the track of previous and present status of repositories, developers can easily revert back to the previous versions.

What is Github?

* Github is different from git
* Github uses the tools of git
* Github is the longest source code in the world owned by Microsoft in 2018.

**Using the command line git:**

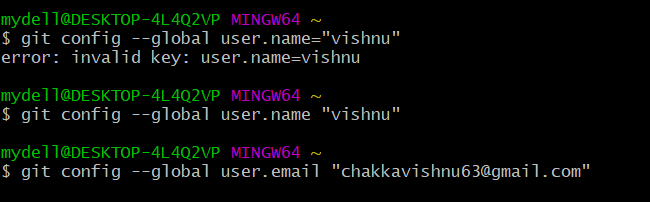
**Config**: used to configure name, e-mail, passwords for each repository

Commands:

$ git config --global user.name "vishnu"

~

$ git config --global user.email "chakkavishnu63@gmail.com"



Creating A Git Folder:

“Mkdir” used to create a folder in git

**Command:** mkdir project

“cd” changes the directory.

**Command:** cd project

**Note:** If you already have a folder/directory you would like to use for Git:

Navigate to it in command line, or open it in your file explorer, right-click and select "Git Bash here"

**Init:** it is used to initialize the repository

**Command:** git **init**

**Note:** Git now knows that it should watch the folder you initiated it on. Git creates a hidden folder to keep track of changes

Git Adding New Files:

🡪 After creating repository we can add files by creating new files and commit

🡪”git add <file\_name>” to add a file in repository

🡪 After adding a file in repository commit the operation by “git commit –m “message”

**Note:** if you didn’t commit the changes you have done till now won’t be reflected in repository.

**🡪”Git status”** is used to know the status of the repository.

**Example:**

git status

**output:**

On branch master

No commits yet

Untracked files:

  (use "git add ..." to include in what will be committed)     index.html nothing added to commit but untracked files present (use "git add" to track)

🡪to make status in a compact way, ut can be done by using “git status –short”

**Note**

Files in your Git repository folder can be in one of 2 states:

* Tracked - files that Git knows about and are added to the repository
* Untracked - files that are in your working directory, but not added to the repository

 When you first add files to an empty repository, they are all untracked. To get Git to track them, you need to stage them, or add them to the staging environment

**🡪**After Creating the file it will be in working repository and which will not be in track with branch.the below commands will be used to perform the action

“Git add <file\_name>” to link a file to branch

“git add .” or “git add –all” to link all files in the branch

🡪 To delete the branch **“git branch -d hello-world-images”**

**Committing without a state:**

Sometimes, when you make small changes, using the staging environment seems like a waste of time. It is possible to commit changes directly, skipping the staging environment. The -a option will automatically stage every changed, already tracked file.

Command:

git commit -a -m "Updated index.html with a new line"

[master 09f4acd] Updated index.html with a new line

1 file changed, 1 insertion(+)

**Warning:** Skipping the Staging Environment is not generally recommended.Skipping the stage step can sometimes make you include unwanted changes.

🡪 To get history of commands it can be done using **“git log”**

**🡪** if you are feeling difficult to remember all the options of a command.we can use **“git <command> -help”** to get the list of options and what the command can do.

🡪To get all the possible commands, it can be done by” **git help --all**”

**Git Branches**

🡪Branch is a new copy of repository

🡪for instance,if you want to update the existing code. If we modify the code in the same branch with any mistakes.the original files cannot be reverted back.So, it can be done by creating a branch.

🡪After Creating a branch work on that code and advantage the changes cannot reflect the oorginal repository And finally merge the repository with the branch

**Steps to create a branch:**

**Step 1**: Create a branch using **“git branch <branch\_name>”**

**Step 2:** verify whether the branch is created or not by checking **“git branch”**

Git branch command will displays the list of branches associated with the repository and ‘\*’ indicate the active branch

**Step 3**: switch to branch using **“git checkout <branchname>”**

**Step 4:**perform the operations on that branch

**Step 5:** get back to master branch and merge the branch with master using **“git merge <branch\_name>”**

**Procedure to upload local repository data with github:**

**Step 1:** Create a repository in github

**Step 2:**link to local repository from github by “**git remote add origin** [**https://github.com/w3schools-**](https://github.com/w3schools-) **test/hello-world.git” .**

**Step 3:**push the data from local to github by **“git push origin <branch-name>”.**

**Github Edit File:**go to the repository🡪file🡪click on edit-icon🡪perform updation🡪commit changes

**Git pull🡪**it is used to keep the files in repository up-to-date and it is combination of fetch and merge

Steps to push in github:

🡪go to repository

**🡪**click on create pull request

🡪select base as master and compare select another branch name

🡪click on create pull request

🡪merge the pull request

Github flow

The GitHub flow works like this:

* Create a new Branch
* Make changes and add Commits
* Open a Pull Request
* Review
* Deploy
* Merge

🡪Github doesn’t allow to modify someone code to their repository without access rights.

🡪by using “**fork”** we can get a copy of the developer data to our github account .This can be done by visiting their repository and click on fork button.

**Note:** Git doesn’t have fork method

**Cloning a repository:**

🡪cloning a repository involves having all the files,status,logs,etc.

🡪to get the repository of github to the local git it can be done by using **“git clone <link>**”

Note:to specify the specific color to be cloned, it can be done by “**git clone <link> myfolder”**

**🡪** finally, a copy of repository will be in local git

## Configuring Remotes

🡪basically we can’t make any changes in their own reposities

🡪 So we can make modifications from our repository by following the below steps

-change the remote name to avoid naming conflicts and this can be done by command “git remote rename orgin <desired-name>(upstream)

--git remote –v to check whether the remote name is updated or not.

--After that clone the forked repository with “git remote add origin “link”

--finally, you will get two remotes

-upstream for read-only

-origin for modification

🡪so, the above is the process to modify the other person repository and add them to your reposioty

**Pull request**:

🡪you can pull request to admin

🡪 to do this we can move local repository to github by”git push origin “ in git command-line

🡪and go to github and create a pull request in your repository

🡪base as their repo and compare with your repository

🡪specify the appropriate message and send the pull request

🡪if admin feels that it’s feasible and he will merge the pull request and confirms the merge request

Git Revert:

🡪 git revert is used to revert back to the previous commits

🡪 for this go to log by “**git log –oneline**”

🡪 copy the hash code of revert back version

🡪”**git revert <hash-code>** “ . this will revert back to previous version and it’s behind push and tree is clean.

🡪git push to move ahead.

🡪 this will changes to the desired version.

🡪 for latest commit can be done by git revert HEAD –no-edit

-–no-editto to skip the commit message editor (getting the default revert message):

**🡪** we can go back to any specify the number of commits by **“git revert HEAD~X”**(where X represents a number like 5 indicates fifth from last”)

**Git Reset**: git reset is used to revert back to last commits discarding the last commits

**Syntax:** git reset <Hash-code>

🡪 to perform reset we need to check the log and and choose desired hashcode and implement the operation.

**Note:** This **is a bit** dangerous task as the last committed data will be lost working in our local repository is fine but, working in the team it may affects the others work

**Git ammend:** git amend is used when we want to make the last commit corrections

**Syntax:** git commit -ammend -m “Updating the last commit message”.

**Git Ignore:**  gitignore is used when we are not interested to share logfiles, tempfies,hidden files etc.

**Syntax:** create a .gitignore file by “**touch .gitignore**” and add all the patterns like “\*.log”ignores all the log files.

and there are many patterns to perform various operations.

Note: In this case, we use a single .gitignore which applies to the entire repository. It is also possible to have additional .gitignore files in subdirectories. These only apply to files or folders within that directory.

**Git SSH:**

🡪SSh is secured shell network which is used for remote transfer, file transfer, file system management.

🡪SSH is a kind of authorization to transfer the data over the internet.

🡪SSH keys are used to initiate a secure "handshake". When generating a set of keys, you will generate a "public" and "private" key.

**🡪** public key is shared with remote party and private key stays wth out.

Steps to generate SSH Key pair:

🡪Generate a ssh key using **ssh-keygen -t rsa -b 4096 -C "test@w3schools.com"**

🡪You will be prompted with the following through this creation:

Enter file in which to save the key (/c/Users/user/.ssh/id\_rsa):

Select a file location, or press "Enter" to use the default file location.

Enter passphrase (empty for no passphrase):

Enter same passphrase again:

Entering a secure passphrase will create an additional layer of security. Preventing anyone who gains access to the computer to use that key without the passphrase. However, it will require you to supply the passphrase anytime the SSH key is used.

Now we add this SSH key pair to the SSH-Agent (using the file location from above):

**Example:**

ssh-add /Users/user/.ssh/id\_rsa

Enter passphrase for /Users/user/.ssh/id\_rsa:

Identity added: /Users/user/.ssh/id\_rsa (test@w3schools.com)

You will be prompted to supply the passphrase, if you added one.

Now the SSH key pair is ready to use.

**ADD SSH KEY:**

* After Creating the key open the file and copy the public key of it.
* Go to GITHUB and Head on to Settings and select SSH And GPG Keys And click on New SSH Key
* Paste the public key in key section and Specify a title and click on **“add SSH Key”** buttuon

Add A new remote

🡪First, get the SSH address from our repository on GitHub:

🡪 go to clone and copy the ssh link

🡪and go to git and run the command “**git remote add ssh-origin ‘<link>’** ”

🡪And finally set url using **“git remote set-url origin ‘<github-link>’ ”**