**GIT HUB**

Video link : <https://www.youtube.com/watch?v=LIhE7L__E6M&t=5815s>

* It is mainly used for version control which means it saves the previous codes or scripts by mentioning them as versions.
* Creating git repos, merging process is also done in git hub.

**Version control system:**

* To store different versions of the codes or scripts. Previously there is no VCS the code is distributed to all and each work on their part and merge them at the end. It is difficult task.

**Centralized** **system**: Here the one person completes his code and commits it into a central server and the second person will pull the code to complete his work. They cannot share this code between them, it happens only through the central server. (This used in the past years)

**Distributed system:** This is popular and is being used currently. We can share the code with others directly without the central server.

Git is a distributed version control system.

**Flavours of Git:**

* Git hub, Git lab, enterprise github, git lab, azure devops.

**Git Hub:**

It is a Microsoft solution of git, it is combination of open source and enterprise.

Read me file should be created for every repository. It has a detailed description of the repo.

**Add file into repository: (by external person)**

**Process-1**

* Create a directory and initialize git in it. (git init)
* Create a file and add the content.
* git add .
* git commit -m “Messaage”
* git status
* git log
* Create a repository in the GITHUB and copy its path.
* git remote add origin (path of the repo)
* git remote -v (cross verification)
* git push origin master

Finally check in the GITHUB for the files.

**Process-2**

* First we have to clone the repository. Click on the code >> clone >> HTTP,SSH, Github CLI links will appear.

HTTPS: for this you need github username and password

SSH: you need public key and it should be saved in settings of github. (SSH used in automation)

* ssh-keygen -t rsa command to create public key in git bash. That key is stored in git SSH.

1. git clone (ssh link)
2. cd repository name
3. vim filename
4. git status
5. git add filename
6. git commit -m “message” filename
7. git config --global user.email "chsaikumar720@gmail.com"
8. git config --global user.name "Your Name"
9. git push

Refresh your github repository the file will be added into that repo.

**Repository:** It is a storage space for the codes. These repositories can be in any of the tools like Git hub, Git lab, etc.

We install git in our PC and using Git hub in online website.

**Configure git and git hub:**

Create a communication channel between git and github. For this we need to open git bash in PC and configure using email and password

**Git config**: a tool to configure the git it has the following

* --global: in home directory “.gitconfig” is the folder that has details of owner email, user name, etc
* --system: details of all users
* --local: project related repos information

Command: connecting git and github

git config --global user.name “ch-saikumar-720”

git config --global user.email “chsaikumar720@gmail.com”

git config --global -e (to edit the detail and cross check)

git config –list ( complete details)

**Git configure with notepad++:**

To connect notepad++ with git helps in easy editing of the files. After setting environmental variables use command “notepad++ “to open the notepad directly from git bash

Alias: use this commands

* Notepad++.bash\_profile (in bash)
* Alias npp=’notepad++.exe -multiInst -nosession’ (in notepad++ to create a shortcut)

We can make the notepad++ as default editor using command: -multiInst git config –global core.editor “notepad++.exe -multiInst -nosession”.

**Work Flow in GIT:**

Commit: It contains the details of the users who done changes with date , time, author,type, etc

Clone: exact copy of the code

Tracked/Untracked files: “.git” is the folder that tracks the entire details of the files such as users, changes, date, time, etc.

Branch:



* The code will start from the Working directory where we do the changes.
* Then it will move to the staging area or index area and then into the local repository. Finally it will be moved to the GitHub remote repository.

Example:

**Master Branch:** The branch of the project which we take from the project by cloning is called a Master Branch. The sub branches from this master branch are called feature branch.

**Origin:** the default server name or project name is called the Origin.

PUSH

commit

ADD

Staging area

Local repository

Remote repository

Working directory

Follow these commands to pull code from repos

1. cd /F
2. mkdir Directory name
3. git clone (ssh link)
4. cd repository name
5. open your My PC and in F frive you will find the repo and do the changes
6. git status
7. git add filename
8. git commit -m “message” filename
9. git push origin master(or main)

Here we pull the code into the F drive and in a folder of our PC using the SSH clone link.

Then we do the changes by opening the code

Check the status and ADD it into the staging area. Then commit into the local repository and Finally push it into the Git hub master repository.

Every Git repo has a .git file that trackes the changes and modifications of the files or content inside the repo. It is basically hidden.

**Clone:**

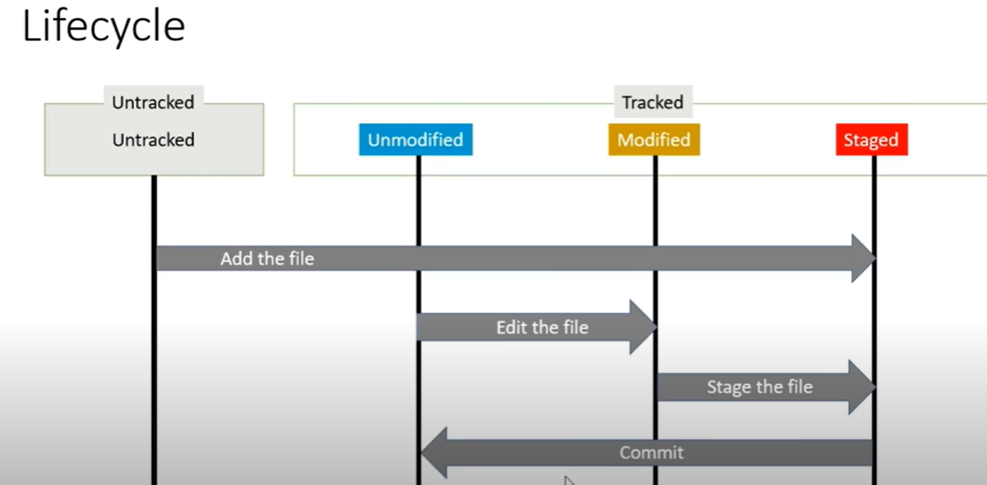
We can clone the repository through HTTPS and SSH. To clone through SSH we need to create a public key to our PC using commands and store it in the git hub account then only we can access this feature.

**Creating new project and pushing it into a new repository:**

before adding the files or data into repository from new project we should have a .git file and this is created by using the command git init.

* create a folder in local computer with the codes and supporting files.
* Create a new repository in the GITHUB.
* Open git bash
* git init ( to create .git file)
* git add . (adds all the files in the folder)
* git status (to check the status)
* git commit -m ‘the files are added’
* git remote add origin (paste the repo link)
* git push -u origin master

git status is the command used to know the status of our project.

**Git Status life Cycle:**

After Add command the file moves to staged state.

After commit command it goes to unmodified state and where we can modify it if necessary. Then it moves to modified state.

After again we should to use add command to move it to staged.

Finally we can push the file to Remote repo. (use git status command)

* Git diff is a command that gives the content what is modified after the file is in staged. Any changes done will be visible using the git diff command.
* Git diff –staged is a command used to check the files that are in staged state.

**Git commit:**

* Git commit -m “message” (this is inline comment commit)
* Git commit -v (additional info is added)
* Git commit (opens vim to write the message)

**Git Log:**

* Gives the history of the project.
* This command gives the all the commits done in the project.
* git log -p this command gives the diff between the commits.

**Delete the directory in git:**

* Rm -rf filename

--------------------------------------------------------------------------------------------------------------------------

**GIT and GITHUB: (Abhishek Verammalu)**

Basically, Git is used for two applications

**1.Sharing:**

The developers, testing, application teams, etc.in a project divide their work and sometimes they need to share their code and for this purpose we need a single repository to store and access the code by anyone who have access.

**2. Versioning:**

If a developer developed an application and added extra features and sent to production then if they go unused by users. Then the team has to remove the extra features. This versioning helps in that by coming back to the previous version code where the extra features are not there.

**Centralized Versioning:**

Dev1 Dev2

Central repo

Here the code is pushed into the central repo by dev1 and the dev2 pulls the code but there cannot be any code sharing between them without the central repo. The centralized version control system is maintained by an system admin, if it goes down then everything will stops.

**Distributed Versioning:**

Dev1 Dev2

Central repo

Here there will be a central repo but we can create many copies of the central repo and can share them with other devs.

Fork: The entire code in the source of the git repo is created into a copy of the original. Even if any one of it got lost we can have any copy.

**GIT:**

It is an open source we can download and create a repo.

We need to install git in our local system and use git bash. Create repos in local system.

* git init : to initialize a repository.(first step) It creates a .git file inside the repo. The .git file will track the actions performed in that particular repository.

Inside .git file we have

Refs, objects, hooks, config and HEAD

**Objects:** everything in git is an object so to track them we have this.

**Hooks:** To prevent unwanted actions and commits in git we use this.

**Config:** it stores the passwords, security things.

The files that are created in the git are untraced till we use the ADD command for the files.

* git diff: the changes will be visible.
* git log: After the changes made we can verify them and know the commits using the command.

All the commits and commit messages will be visible and each version will have a version ID.

* git reset –hard version ID: The previous version will be updated and overridden. Can be verify using the cat filename. Switching from one version to another.

All these above are in your local desktop. We can switch to multiple versions. Create a git repository, logs, track files, diff, status. This comes under GIT. We cannot share it with others.

In Git we use only git add and git commit to create versions.

**GITHUB or Bitbucket or self hosted git:**

The git hub concept comes into picture when we want to share our code with peers or keep it open source for others. This makes the git hub a distributed system.

**Git Branching:**

Branching means taking the copy of the master or main and executing the changes, test and finally if we got good results then we can merge the feature branch to existing master branch. We can create any no. of feature branches.