**Git and Git Hub**

**Git:**

Free and opensource version control system

**Version control**

The management of changes to documents, computer programs large websites and other collection of information.

We can look back at all the changes we made the code,

**Note**: Basically, the Git will save the code by versions which is useful to go back to the changes

There are some Gui software to manage git like Git Kraken, Github desktop.

But, using terminal is best because it’s easy to manage since the code we right are execute in terminal, and it is easy to edit the code in IDE too.

**Git and GitHub**:

Git is a system that manages the project by version control and tracking, whereas github is the online platform that manages git through cloud enables us to connect with other developers.

**Git Command**:

Clone -> Bring the repository that is hosted somewhere like github into your local machine, which isn’t available in you machine originally

Add -> Track the changes in the file or the git folder

Commit-> Save you file to git or update with the newer version of code

Push -> Upload git commits to remote repository like github

Pull -> Download the changes from the remote repository to your local machine(opposite of push), where the older version of those repository is already in you local machine

**Repository:**

Basically the project folder which contains all the files and codes related to the project.

**Commands:**

Git Clone:

Use: download the remote repository to you local machine

Command : git clone <clone with SSH code from remote repo like Github>

*Ex:* git clone [git@github.com:siva-varun/learnigns.git](mailto:git@github.com:siva-varun/learnigns.git)

Git Status:

*Use*: to see whether the files are ready to staged for commit .i.e., is the files are ready to commit or saved to the git branch

Shows the details about the files that are modified after the last commit

After the editing that file will be not staged to make the file ready to commit use the add command

*Command*: git status

Git add:

Use: to make the modified files to ready to perform the commit operation to the git branch

*Command*: git add .

git add <file name>

*example* : git add index.html

*note* : “.” Used to add all the file that are modified or added to the git folder to ready for commit

Git commit:

*Use*: adds or save the code to the git branch. Each commit is consider as a version

commit message is mandatory while using commit, this commit message will be extremely help full to the version or changes to the code if some problem arises and need to go back the code

Can directly use commit without adding seperatly, if there is only have modified files but no newly added file

*Command*: git commit -m <message you need>

git commit -m <msg you need> -m <optional msg description>

git commit -am <msg for commit>

*Example*: git commit -m “added a comment feature”

Note: -am -> stands for “add and message”

Git push:

Use: after the commit the code is only update locally, to upload the changes to the remote repository “push command is used”

*Command*: git push origin master

git push -u origin master

*explanation*: origin -> location of the git repository always “origin”

master -> branch that code need to be pushed (any branch name e.g. Git push origin dev. Where “dev” is a branch)

-u -> is used to set the location and branch as default for the future push command. Here “-u” sets “ origin master” as default.

Note: to use git push to github you need to authenticate that your are the owner of the repository that is present in the github

To do that:

Create an SSH key

Ssh-keygen -t rsa -b 4096 -C [youemail@email.com](mailto:youemail@email.com)

Rsa – encryption method

4096 – encryption size

You can have a name and password for you SSH key that your are generating

There will be two files create “filename” is the private key and “filename.pub” is the public. Public key is key that will be uploaded to the github

While making the push from your local machine you will send the private key, so that the github can understand that only that particular private key can able to generate the uploaded public key hence authenticate

The key will have you mail in the ending

Copy the content of the public key and add it in the github SSH section in setting

Added the SSH key to the ssh-agent so that the push request will get the key to send the request

Refer : <https://docs.github.com/en/authentication/connecting-to-github-with-ssh/generating-a-new-ssh-key-and-adding-it-to-the-ssh-agent>

Creating the git repo locally and pushing to the github :

After creating the folder and the file inside it use command “git init” make that folder into a git repository.

*Command* : git init

*Use* : makes a project folder into git repository in the local machine

Then you can use the “add” and “commit” command to commit the changes in the branch to the newly created repository.

But, you can’t push the file to the github, because there is no repository created in the name of the newly created repo in local machine.

*To solve that*:

create the new empty repository with the same name in the github, then copy the unique SSH git code for that repository which will be like “ [git@github.com:siva-varun/new-repo.git](mailto:git@github.com:siva-varun/new-repo.git)”

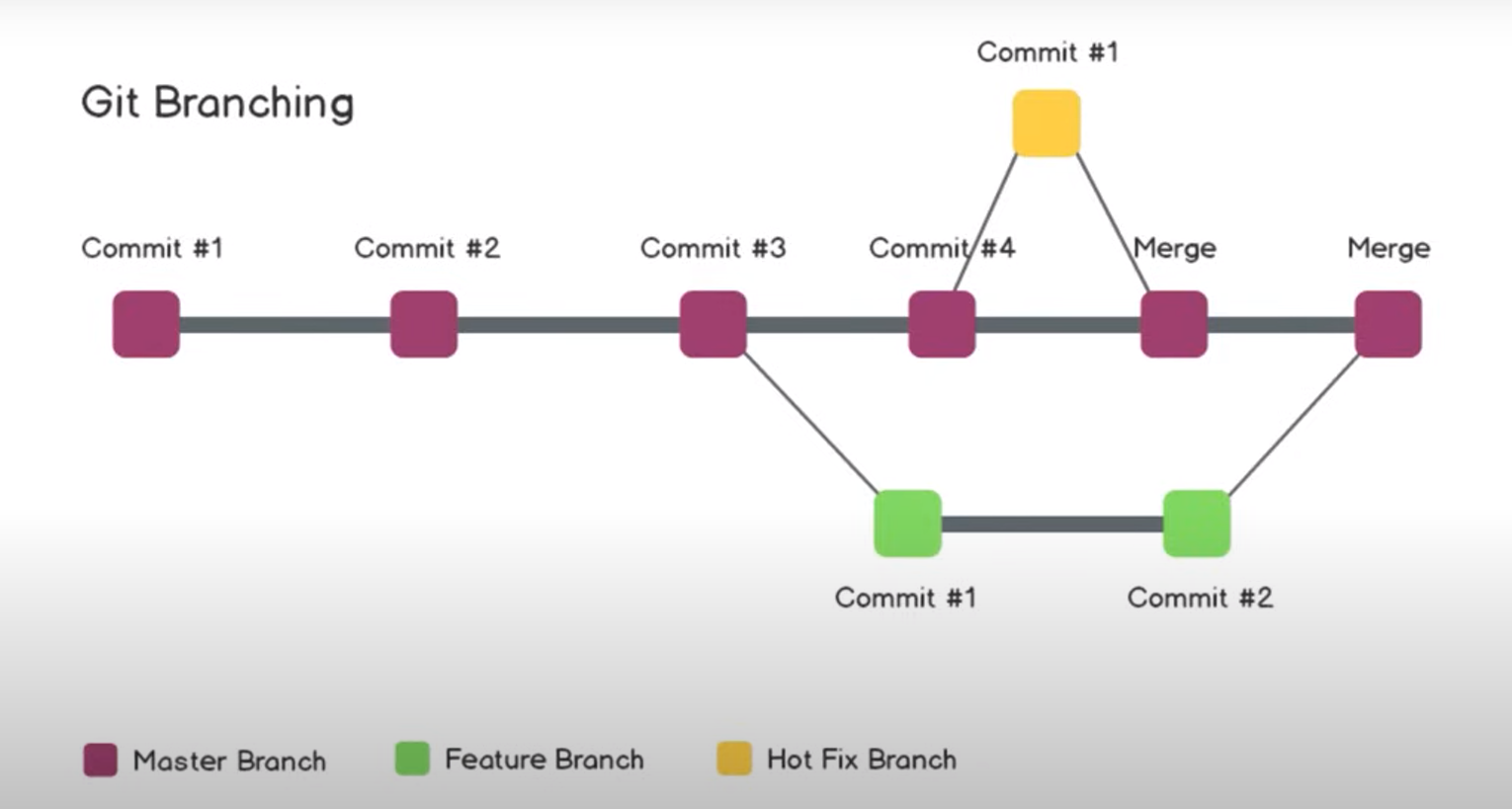
in the local machine run “git remote add origin” command to link the github repo with the local repo

*command*: git remote add origin <SSH git code>

*example*: git remote add origin [git@github.com:siva-varun/new-repo.git](mailto:git@github.com:siva-varun/new-repo.git)

now you can push the local repo to the github by using “git push” command

**Git Branching:**



Branching is used to create the entirely new feature or make a change that don’t want to affect the main branch or the master branch of the code.

Once the feature of change is completed without bug or error, it can be merged with the master branch, which makes the complete project with features

It is very useful where multiple people are working in the same project. Branch helps the developers from interfering from one other.

It also helps to maintain the core code of the project while there is also the feature development.

The branches other than “master branch” can be deleted after merging with the master branch, but some of the branch can be kept permanent to reduce the interfering with the master branch.

Can be used to fix the bugs, the branch can be named by the ticket number for better identifying purpose

Git branch:

*Use*: to view the all the branch – the working branch is indicated by “\*” sign and also used to delete the branches

*Command*: git branch

git branch -d <branch want to be deleted>

*example*: git branch -d comment-feature

Git checkout:

*Use*: used to create a new branch

Also used to switch between the branches

Always use commit to commit the changes before switching the branches

*Command* : git checkout -b <new-branch-name>

git checkout master

*Example*: git checkout -b comment-feature

git checkout master

Git diff:

*Use*: can show the difference between two branches,

Helpful to compare the code before merging

*Command*: git diff <feature-branch>

*Example*: git diff comment-feature

*Note*: your current branch must be the branch that you want to compare with.

Git merge:

*Use*: to merge the feature brand with master branch or other main branches

The common practice is to push the feature branch to the github and making the pull request to merch with the main branch.

It is better to merge the code in github or other remote machine which contain GUI, better for approval and comment tracking

To push the feature branch the command will be

:” git push origin <feature-branch>”

*Command*: git merge <feature-branch>

*Example*: git merge comment-feature

*Note*: in above example the feature branch will be merged to your current branch

**Pull request or PR:**

Pulling the feature branch into the master branch(Merging the feature branch with main branch), it has the feature like add approval requirement before merging and comment features for the pull request.

Once you pushed your new branch there will be the option emerge in the github for creating the pull request. You can use that to create and track the pull request.

It is always advisable to make the pull request in the remote machine which has GUI for better results like approval and commenting

Git pull:

*Use*: download or pull the remote repo to you local machine, where you have the repo in local machine but not updated as the remote repo

*Command*: git pull

*Note*: the current branch will get updated as the remote branch

**Merge conflict:**

Merge conflict occurs when there are multiple changes like deletion of few line and adding few lines, so the git will confuse which change to take and which changes to drop.

It occurs when the master gets updated by other and when you merge your branch there may occur some overlapping

While performing the merge which has merge conflict the code won’t completely merge but will create a intermediate merged file that contains changes of the conflict.

We can edit the intermediate file and then commit with the master or the main branch.

**Undoing or Going Back in Git**

We can go back to the previous version of the code like before staging, or before single or few committing

Git log:

*Use* : used to view all the commit changes

Mostly used to find which commit to go back if we decide to undo or go back to previous changes

*Command*: git log

Git Reset:

*Use*: helps to go back to the older version or previous version or before staging

*Command*: git reset <name of the file to unstage>

git reset HEAD~1

git reset <hash of commit from git log command>

git reset –-hard <hash of commit>

example: git reset redme.md

git reset 154153484adf4684321616846sd12

git reset –-hard 154153484adf4684321616846sd12

note: --hard parameter completely resets the file before the commit that the hash has mentioned or given

while using without –-hard only unstage(the file in ide is not changed but also not added to the commit) the file before the commit that hash has mentioned or given

**Forking**

Used to copy the full repository with all the branches and commit from some other user to user own as repository.  
 it is copied to your own account you have the full access to that particular repo because it’s in your account as a copy

You can also make a PR to the original repository from you copied repo, it will require approval based on the condition set by the original repo owners