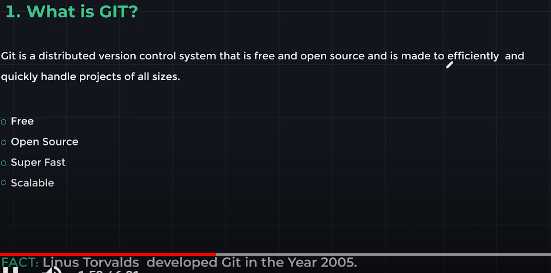
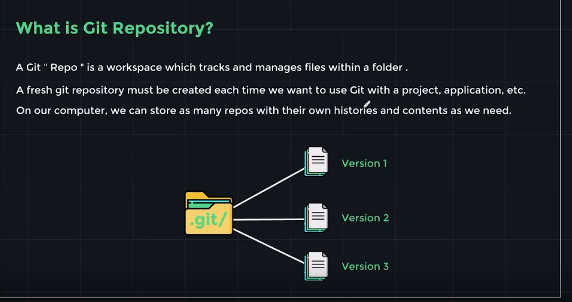
GIT

Git Basics:

D

What is Git Respository?



Git Setup Commands:

$ git init : Create an empty Git repository or reinitialize an existing one.

$ git config –global user.email <email> : Initialize the user email id for all repositories.

$ git config –global user.name <name> : Initialize the user name for all repositories.

Basic Git Commands:

$ git status: Displays the state of the working directory and the staging area.

$ git add. : It updates the current content of the working tree to the staging area.

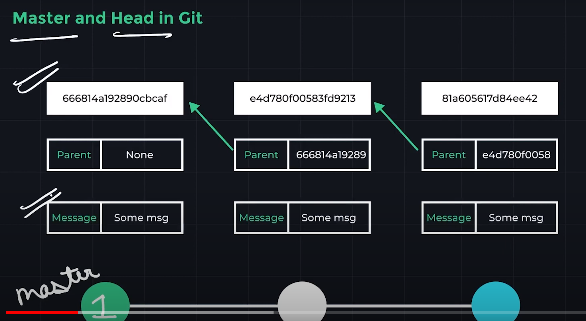
$ git commit -m “Message” : It is used commit the changes in the repository.

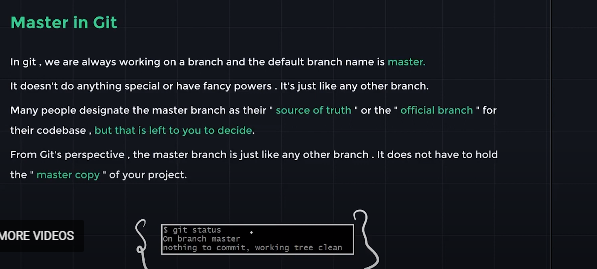
$ git log : It show commit log.

$ git log –oneline: Display log in oneline

Branching and Merging:

The Master and The Head:







Creating and Switching Branch:

Branching in Git:

* A branch is a new/separate version of the main repository.
* Consider branch as different project timelines. They allow us to set up distinct environments where we experiment or even work on several ideas at once.
* Changes made to one branch do not affect the other branches(unless we merge the changes)
* $ git branch <branch\_Name>: Creates a new branch.
* $ git branch: Shows all the available branch.
* $ git switch <branch\_Name>: To switch to an available branch.
* $ git switch -c <branch\_Name>: To create and switch the branch.
* $ git checkout <branch\_Name>: To switch/checkout the branch.

Deleting and Renaming Branch:

* $ git branch -d <branch\_Name> : To delete an independent branch.
* $ git branch -m <New\_branch\_Name> : To Rename an existing branch.

Merging in Git:

To join two or more development histories together.

$ git merge <branch\_Name>: To merge a branch.

$ git merge --abort : To abort merging two branch.

Git Stash:

Stashing in Git:

Git provides an easy way to stashing these uncommitted changes so that we can return to them later, without having to make unnecessary commits.

$ git stash is an incredibly helpful command that enables you to save modifications that you aren’t quite ready to commit. Changes can be saved so you can access them later.

By using git stash, you can roll back the changes in your working copy by stashing all pending changes(staged and unstaged).

* $ git stash: To stash all uncommitted changes.
* $ git stash pop: To remove the most recently stashed changes in your stash and re-apply them to your working copy.
* $ git stash apply: To re-apply stashed changes to your working copy without removing them from stash.
* $ git stash list: To see all the stashes.
* $ git stash clear: To remove all the stashes.
* $ git stash drop<stash\_name>: To drop/remove a particular stash from stashing list.

Revisiting Versions:

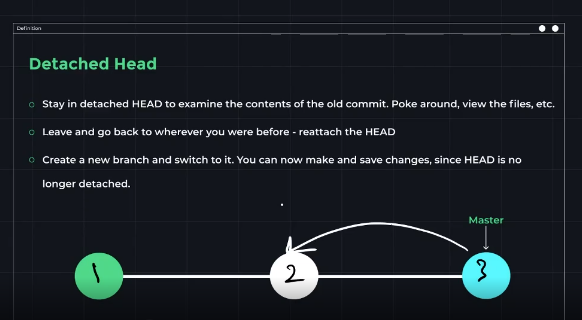
Checking out old commits:

We can use $ git checkout <commit-hash> to view a previous commit.

Remember, you can use the git log command to view commit hashes. We just need the first 7 digits of commit hash.

$ git log – oneline : To get hash value of commited file.

Detached Head:



Reattach Head:

$ git switch <branch\_Name>: To reattach head to a particular branch.

$ git switch - : To reattach head to a parent branch you were on.

Un- Modifying with Git Restore

$ git restore <file-name> : The command that helps with undoing operations.

$ git restore – staged <file\_Name> : To remove files from staging area.

Git Reset:

Suppose you’ve just made a couple of commits on the master branch, but you actually meant to make them on a separate branch instead. To undo those commits, you can use git reset.

$ git reset <commit-hash> : will reset the repo back to specific commit.

$ git reset –hard <commit> : To remove the contents and undo commits.

GITHUB

Git Clone:

* $ git clone <url\_of\_the\_Repository>
* Git will copy all the files connected to the repository to your local system after retrieving them. The whole Git history of the cloned project is also accessible to you because Git creates a new repository on your computer.

Git Remote

* $ git remote
* Before we can push anything up to Github, we need to tell Git about our remote repository on Github, we need to setup a “destination” to push up to
* In Git, we refer to these ‘destinations’ as remotes.
* Each remote is simply a URL where a hosted repository lives.
* $ git remote -v

Adding a New remote

$ git remote add <name> <url>

Perfered name for remote is origin

A remote is really two things, a url and a lable. To add a new remote, we need to provide both to git.

Git Push

$ git push <remote><branch>

This tells git to push up the branch to our origin remote.

Git Fetch

$ git fetch <remote>

* The command fetches branches and history from a specific remote repository.
* It only updates remote tracking branches.
* Git fetch origin would fetch all changes from the origin remote repository.

Git Pull

$ git pull <remote> <branch>

To pull, we specify the particular remote and branch we want to pull using git pull <remote> <branch>. Just like with git merge, It matters WHERE we run this command from, whatever branch we run it from is where the changes will be merged into.