GIT(GNU Interactive Tools) - It is a code management system and version control system. GIt tracks all the changes made by different users. Information about a project is stored in the repositories. And these repositories have committed to the project. It records our project history.

Version Control System - these systems are tools which help to track changes in the file over a period of time and save them. When multiple people are working on the same code, it tracks all the changes and keeps updating them. At some point if our code stops working due to some errors then we can go back to the old code and see the changes we made which makes the processing of recording actions easy and simply traceable.

Commit - It's a record of the change in the repo. Commit has a message and stores the changes we made in that which make it easy to roll back. When a commit is created then a unique identifier is created called commit hash which allows it to easily identify.

Repository- it is basically a folder where all the changes are saved and git provides us this to record all the changes whether adding some stuff or deleting some stuff etc.

- Working directory
- Staging area: holding area for queuing up changes for next commits
- Commits (history)

These three stages are on the local system and the remote repo is in github.

Git-master
github-main

Master branch:(main) timeline which contains all the changes, branches contain commits. Git provides us a default branch called master branch.

- Git congif - -global user.name “radhika”
- Git congif - -global user.email “radhika.com_etc”
- Git congif - -global -- list
- Git push origin main
- Add username and password(PAT) or personal excess token to save file to the remote repository that's github.

Git Quick Start Commands

Command Listing, Part 1

```
pwd
mkdir projects
cd projects
pwd
```

Command Listing, Part 2

```
git version
git config --global user.name "Abe Lincoln"
git config --global user.email "mrabe@git.training"
git config --global --list
git clone github-https-url  # paste in your GitHub HTTPS clone URL
ls
cd github-demo
ls
git status
echo "Test Git Quick Start demo" >> start.txt
ls
cat start.txt
git status
git add start.txt
git status
git commit -m "Adding start text file"
git status
git push origin master
```

- The nano command is used to open and edit text files in the nano text editor, which is a simple and user-friendly command-line text editor.
  Nano file,txt
  The cntrl+o for saving
  Control +c
  Control+x
  Enter

- The **git add -A** command is used in Git, a version control system, to stage changes for a commit. The -A flag stands for "all" and tells Git to stage all changes, including new files and deletions, in the current directory and all subdirectories
- git add -A stages all changes, including new files, modified files, and deleted files. It's a short form of git add . && git add -u.
- git add . stages all changes, including new files and modifications, in the current directory and all subdirectories, but it will not stage deleted files.
- In Git, the git branch -m command is used to rename a branch. The -m option stands for "move" and tells Git to move the branch to a new name.
- The command git branch -m main is used to rename the current branch to "main". The current branch is the branch that you have checked out, and it will be renamed to "main".
- The command git init -b main local-demo is used to initialise a new Git repository and create a new branch named "main" at the same time.

- The command `git config --global init.defaultBranch main` is used to configure the default branch name for new Git repositories.

- **git push origin main:** This command is used to push changes to a remote Git repository. The "origin" parameter specifies the remote repository, and "main" specifies the branch that the changes will be pushed to. This command will upload any local commits on the current branch to the remote repository, updating the specified branch on the remote with the new commits.

- To check whether git is installed in our system or not, we need to type the command (git) in the terminal and it will show the list of items inside the git it is installed.
- **Git version** to check git in terminal.

- **Config**: In Git, the "config" command is used to manage configuration settings for a local repository. These settings include things like the user's name and email address, which are used to identify the author of commits, as well as remote repository URLs and branch tracking information.

- **Fetch**: In Git, the "fetch" command is used to download new commits and other objects from a remote repository to a local repository. The fetch command retrieves commits and other objects that are not present in the local repository, but are present in a remote repository. It updates the local copy of the remote branches, but it does not merge the changes into the local branches.

- **"git rm"**: in Git, the "git rm" command is used to r"remove" in git. The command "git rm" is used to remove files from the Git repository and also from the file system. It is used to delete files from the repository, but it also stages the removal for commit.
  git rm example.txt

- **Amend:** In git, "git commit --amend" is a command that allows you to modify the most recent commit. It allows you to make changes to the commit message or add more changes to the files that were already committed.
- **SHA (Secure Hash Algorithm)**: is a unique identifier for each commit in a repository. It is a 160-bit (40 characters) long value that is generated based on the contents of the commit. The SHA hash is unique and will change if any part of the commit changes, making it an effective way to identify and verify commits.

- **"git diff":** The basic usage of "git diff" is "git diff", which will show the differences between the working directory and the repository.

    For example, "git diff <file>" will show the differences between the working directory and the repository for the specified file, and "git diff <commit_hash> <file>" will show the differences between the specified commit and the working directory for the specified file.

    You can also use "git diff --staged" to see the changes between the files currently staged for commit and the last commit or "git diff <branch_name> <branch_name>" to see the differences between two branches

- Git calculates differences, or changes, between commits by comparing the contents of the files in each commit. The term "HEAD" in Git refers to the most recent commit on the current branch.

- **git-show:**In git, "git show" is a command that displays the details of a specific commit. It shows the commit message, the author, the date, and the changes made in the commit. You can specify a commit hash or reference to view the details of a specific commit.

- **Git stash:**In git, "git stash" is a command that allows you to temporarily save changes that you have made to your working directory, but you are not ready to commit yet. It allows you to switch branches or pull in updates from a remote repository without losing your current changes.

- **revert:**To revert means to undo or return to a previous state or version. In the context of Git, it can refer to undoing changes made in a specific commit using the "git revert" command, or returning the entire repository to a previous state using the "git reset" command. Both commands allow you to undo commits and change the repository's history.

- **Git log:**The log is displayed in reverse chronological order, with the most recent commit at the top.
    For example, "git log <file>" will display the commit history for the specified file, and

"git log <commit_hash>" will display the commit history starting from the specified commit.

- **Reset**: In Git, the "reset" command is used to undo commits and bring the local repository back to a previous state.
- Terminal allows us to manipulate file structures using commands.
- If git is not installed then we can run a command and install it in linux with (sudo apt-get install git). Advanced Packaging Tool (APT)
- Red color indicates the deleted lines and Green color indicates the added lines in the project history.
- **(git init)** this is the first command we need to get in order to create a repository to hold our changes.
- As this ,git file is hidden so we can't see it with the ls command but can be seen with the **ls -a**. All the files which starts with dot(.) are hidden files.
- **(ls .git)** is used to check all the content in the .git hidden file.
- **Untracked files:** they show up when we enter the git status command. These files are those files which are not saved yet.
- **(git add .)**: used to add all the untraced files .We can mention the particular file name at the place of dot(.) for saving that particular file but not all the files.
  Also git add file.txt
- **(git commit -m "file added"):** for adding the file permanently to the history of the project. Here m simply stands for message.
- **(git log):** used to see all the history of the project
- **(rm -rf file.txt):** to delete a file or folder
- **(git reset commit_id):** it will restore the above files.
- **(mv file new_name)** to rename a folder or file
- **Vi file.txt:** vi (short for "vi iMproved") is a text editor that is commonly used to create and edit files
  When you first open a file in vi, you will be in "command mode". In this mode, you can navigate through the file using the arrow keys, but you cannot make changes to the file. To enter "insert mode" and start making changes, press the "i" key. To save your changes and exit vi, press the "Esc" key to return to command mode, and then type ":wq" and press "Enter".

- **(git stash):** is used to save and keep aside the things.
- **(git stash pop):** to get back the things we kept aside.

To push your commits containing your code in .js files in a new branch 'assignment-1' on a remote repository, you can use the following steps:

1. Make sure you are on the correct local branch by running `` `git checkout master` `` or the branch you want to base the new branch on.
2. Create a new branch using `` `git branch assignment-1` ``
3. Switch to the new branch using `` `git checkout assignment-1` ``
4. Add your .js files using `` `git add path/to/your/js/files/*.js` `` or `` `git add .` ``
5. Commit your changes using `` `git commit -m "Adding my code for assignment-1"` ``
6. Push your new branch to the remote repository using `` `git push -u origin assignment-1` ``

CLONING(COPYING A REPO):

- Git clone url(starting with http

  - Default branch in git is as the main not master branch.
  - Pwd(print working directory.)is the command to check the current working directory.

commands
Git “initial commit”
Git checkout -b name

Ls
Ls -a
Git init - create a repo
Ls .git
git status - to check all the changes which are made which need to be recorded or not.
rm -rf filename - to delete the file
git add filename - to add changes in our history (put in staging area)
git add . - put all the changes in staging area
git commit -m “any relevant message”
git restored - -staged filename - To remove a file from the staging area if we don't want to track this change and save it
git log - to check all the commits
git reset commit hash ID- it will give how our work look at that particular commit and delete all the commits after that

git log - show commits
git stash - to put things backstage after git status and git add .
git stash pop - to take things from backstage to staging area
git stash clear - to delete things from stash area
git remote add origin url - to attach github repo to our project on local
git push origin master/main- to push changes there in the url remote
git remote -v
git push origin main(to which branch we want to push)

- We can fork things in our account and then clone to our local

Git clone url- to have project in our system
remote add upstream- from where we forked that url