Git Tutorial

What is Git/Github?

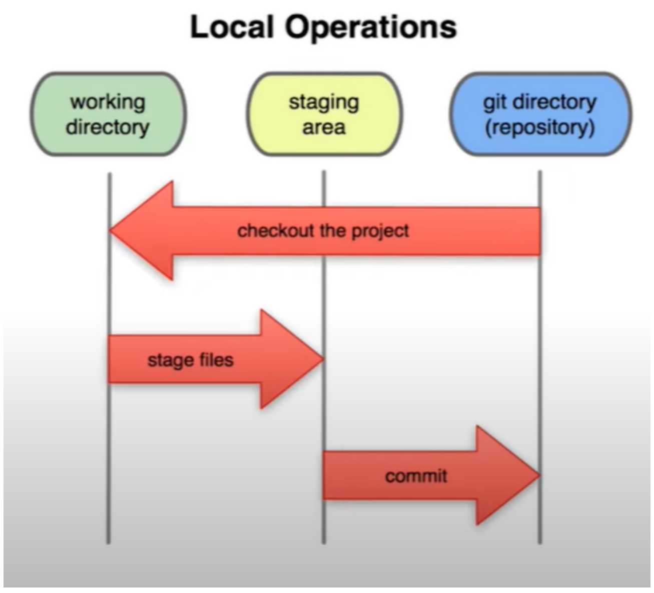
Git is a version control system. Benefits of using Git :

1. Easily recover files
2. Who introduced an issue and when
3. Roll back to a previously working state

Git installation

1. Install git from their website "Git - Downloads".
2. This will install git GUI and git bash.
3. To open the git bash terminal for any specific folder. Go to that folder and press Shift + Right click (mouse). There you can see the option for git bash.
4. Add your username and email on the git configuration using these commands:
    1. **git config --global user.name "Brijendra"**
    2. **git config --global user.email "brijcodes@gmail.com"**
5. Check if it is updated using the command :
    1. **git config --list**
    2. **git config user.name**
    3. **git config user.email**

Git: Three-Stage Architecture



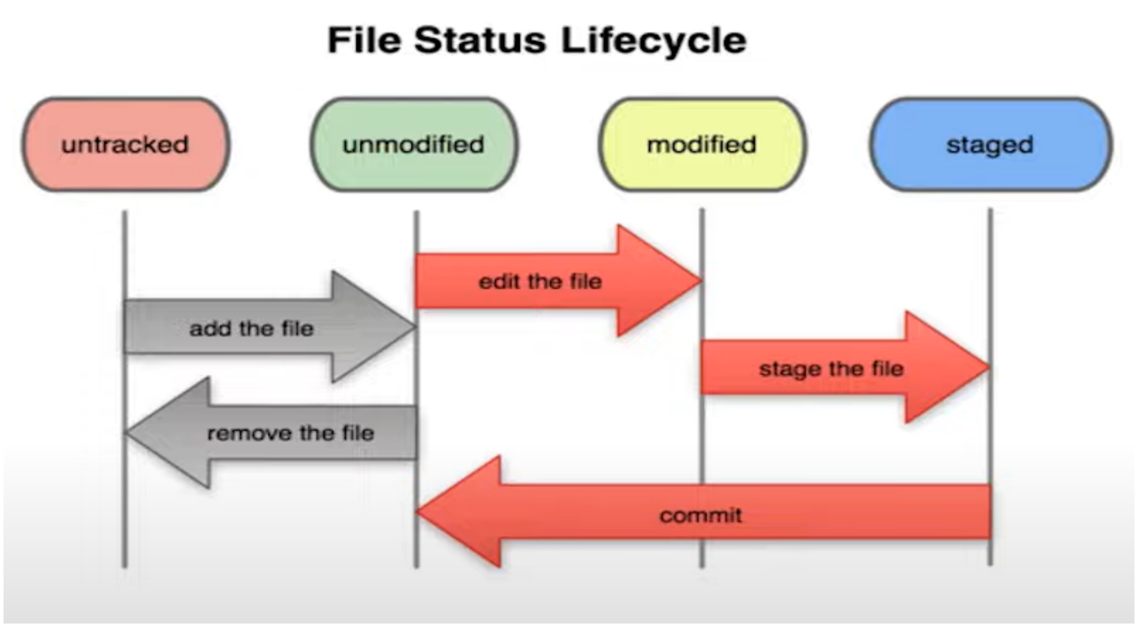
Tracking Our first Git Project

1. Add some files to any folder and open the git bash terminal.
2. Run command: **git status** (This is not a git repository yet)
3. We have to initialize a git repository there using **git init**
4. Add all the files to the staging area: **git add --a**
5. To commit those files use: **git commit -m "<message>"**
6. Check git status: **git status**
7. To check the changes use the command: **git log**
8. Do some changes in files (more than 1) and then check: the **git status**
9. Now we will only send 1 of the file to the staging area: **git add Test.txt**
10. Now check for the **git status**
11. Now we will commit the file which is in the staging area: **git commit -m "<message>"**
12. Similarly, we will do it for other files, add it to staging then commit.
13. To move a file from staging to working directory we use: **git restore --staged <filename>**

Cloning a Remote Git Repository

1. Go to the repository on GitHub and copy the URL from there.
2. To remove a folder as a git repository, the command is: rm -rf .git
3. We can create a copy/clone a repository using: **git clone <URL>**
4. Do changes in some files then add to the stage and commit them.
5. To check changes we can use: **git log,** To quit from logs press **"q"**.

File Status Lifecycle



.gitignore: Ignoring Files in Git

1. We can ignore files in git using: **.gitignore**.
2. Create a file error.log using: **touch error.log**
3. Create another file using this command: **touch .gitignore**
4. Now if you will check: **git status,** it shows 2 files modified.
5. If you write **"error.log"** into the **.gitignore** file and then run **git status,** it will only show 1 file modified. It will ignore any changes made to the **error.log** file.
6. To ignore multiple files with the same extension we can write "***.<file_extension**>" in the **.gitignore** file.
7. To ignore the directory we can add "**<directory_name/**" in the **.gitignore** file.

Git Diff: Showing Changes Between Commits/Staging Area and Working Directory

1. **git diff** shows the difference between file in working directory and file in staging area.
2. **git diff --staged** will show the difference between last committed and staged area.

Git : Skipping the staging area

1. **Git commit -a -m "<message>"** This command will move all the tracked file directly to commit without staging area.

Moving and Renaming Files in Git

1. **git rm <filename>** will delete the file and it will not go to staging area. Directly go to commit section.
2. **git mv <file1> <renamefile1>** this command will rename the file.
3. Remove the tracking of the file using **"git rm –cached <filename>"**

Git Log: Viewing and Changing commits in Git

1. **git log -p** shows the latest commits with differences.
2. **git log --stat** shows the brief about latest commits.

3. **git log --pretty = oneline** shows the commits in one line.
4. **git log --pretty = short** shows the commits in short.
5. **git log --pretty = full** shows the more details about commits.
6. **git log --since=2.days/months/years** shows the commits in last 2 days/months/years.
7. git-scm.com/docs/git-log we can find format options to check commit on this website. Eg. **git log --pretty=format:%h–%an**
8. In vim editor we press **"I"** to insert anything and after that press **"ESC"** then write **":wq"** and hit Enter to save and exit.
9. We can change commit but that would be a complex task.

Unstaging and Unmodifying Files in Git

1. **"git restore --staged <file>"** to send file to staging area to working directory.
2. If we want to revert back to file's last commit we can use : **git checkout -- <filename>**
3. If we want to change all files to last commit we use : **git checkout -f**

GitHub: Working with Remote Repositories

1. **git remote add origin** git@github.com**:Brijendra07/GitPractice.git** this command will create a remote connect to repository "GitPractice" as origin.
2. We can check this by command : **git remote** this will give output **origin.**
3. We have to create a new SSH key and add it to SSH-agent on github.
4. Complete process is available at https://docs.github.com/en/authentication/connecting-to-github-with-ssh/generating-a-new-ssh-key-and-adding-it-to-the-ssh-agent
5. Now we can run the code **git push -u origin master** to push the code.
6. Command to remove the remote connection : **git remote remove origin**
7. To push the code to main branch :

   git remote add origin git@github.com**:Brijendra07/GitPrac.git**
   git branch -M main
   git push -u origin main

8. **git branch -M <branch_name>** Using this command we can switch between branches or create new branch.
9. We can only push the code to the branch we are currently at.
10. **git push -u origin main** will push the code to main branch.

Setting Alias In Git

1. Create short forms for commands Eg. **git config --global alias.st status**
2. Now we can use **git st** for **git status.**
3. **git config --global alias.last 'log -p -1'** after running this command if we want to check last log we can directly use git last.

Git: Creating & Switching Branches in Git

1. **git checkout -b develop** This command will create a new branch "develop" and take us there.
2. **git checkout <branchname>** This command will switch between branches.
3. **git branch** will show all the branches.

Git: Branching and Merging a Production Grade Project

1. **git merge <branchname>** This command will merge the branch to main branch.

Resolving Merge Conflicts

1. **<<<** This sign is conflict resolution markers.
2. To delete a branch we can use **"git branch -d <branchname>"**
3. **git branch --merged** shows already merged branches.
4. **git branch --no-merged** shows branches not merged yet.

Pushing Git branches to remote repositories

1. Setup a remote connection to your github repository : **git remote add origin git@github.com:Brijendra07/gitprac.git**
2. Change branch name master to main using : **git branch -M main**
3. Push your code to main branch : **git push -u origin main**
4. Push any branch to github using **: git push -u origin <branchname>**
5. To delete branch at github : **git push -d origin <branchname>**