GIT and GITHUB - User Guide

Git is used as version control system

Uses and information

- Free and open source
- fast and scalable
- Can get history tracking
- collaborate

github

- website that allows devs to store and manage their code using github
- folders form upload - repository

Account and set-up

- create an account in github and create repository under profile
- select readme every time to add any information about the project

1st repo- Django-Project-Python:

Making First Commit:

- edit a text or anything from README and choose commit
- Commit directly to the main branch
- While committing we can add status messages and we can select branches
- After we can commit to main branches

  Note: The text files inside doesn't work as usual, we need to do syntaxing similar to HTML

System-level operation for GIT

Requirements

- Visual Studio Code
- Windows (git Bash)

- install git from git-scm.com - setup exe and install for default setting
- install Visual Studio Code for windows - setup exe for default setting
- once downloaded we can use command git--version to check if git installed or not

Configuring Git

- Configuring includes name, profile and other repository settings
- Two type of configuration mode
  1. Global
  2. Local (inside a repo can update using any mail-id)

Commands For Configuration in GitBash

- git config --global user.name "My Name" (Name generally will be git hub account name)

- git config --global user.mail someone@email.com (mail will be the account mail we create in git hub)

- git config --list

Main Commands

Clone: cloning a repository into our local machine

Status: get code status

Add: adds modified or new files from local to remote

Commit: it is the record of change

Push: Upload local repo content to the git hub

- git clone <-repo_link->
- git status
- git add <-file_name->

    - git commit -m “Some Messaage”
    - git push origin main

copy repository from git hub to local **[remote {github}]** to [local {windows directory}]

Steps to clone the repository from git hub

a. Get inside git hub repository (Django-Projects-Python)
b. There we will get an option of code (<> Code)
c. Click on Code and copy link from https (for beginners)
d. Now to clone the repo, come to vs code terminal and enter
   Git clone link/from/the/github

Steps to get status of the code

- To get status of a repository
- Basically if we update e a file in the local directory then there will be a miss match with git hub repository
- This will be displayed when we do git status and it shows changes not staged to commit
- After editing a file we have two things to do
    - Add
    - Commit

Types of status for git

- Untracked : new files that git doesn’t track yet
- Modified : A file already present in git but changed
- Staged : if a new file is added or a file is modified, we wll be adding it then the file is staged and ready to commit
- Unmodified : Files that are unchanged
- Commit : After staged, will commit the files which will make files unchanged again

How to add project from local to local-commit

To add means to push the files to staging area

After adding to stage, we will commit the files

Commands

- ✓ Git add <-File_Name->
- ✓ Git add . (This command adds all the changed and new files to the staging area)

For commiting, we can add a meaningful message to tell what we changes we have made

Commands

- ✓ Git commit -m “changes in the current version”

Note : The changes committed will be added to local repository and in order to save it in the remote repo or git hub, we need to

Push it using command git push origin main

How to push the updated commit to github repo

- ✓ Push using command git push origin main
- ✓ It may require us to log in to git hub from vs code for first time
- ✓ In this origin means default repository or the repository we have copied earlier, we can change name as well
- ✓ Main refers to main branch we are commiting

Local directory to Git Commands

First create a local folder and navigate to that folder in vs code

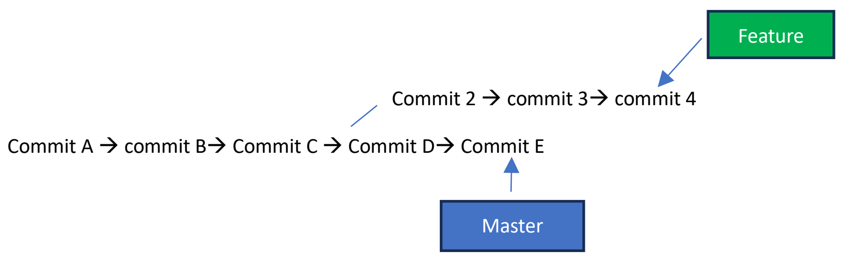
- Git init: after creating a new folder, add gitinit will initialize the folder as git repo
- After initializing the git, we can do add and commit commands
- But before adding pushing the files to the remote repo, we can have few things to do
- First, we need to create a repository in github
- After copy the https link of the repo
- Then add git remote add origin <-repo-link->
- To check if remote repository exist or not we add git remote -v

- Branch: In the repo, several team may be working on several freatures,
- Say one team works on Feature 1 and another team works on Feature 2, these are called as branches
- Each team will take their branches
- So, to check which branch we are we uss git branch
- In the git hub policy, the master branch is changed to main branch, we can either create a new main branch or rename master to main
- To change name git branch -M “name” command can be used
- Then we will push the projects to main using git push origin main
- Also say we are working on a project for long time instead of push it origin main each time we can add command
  Git push -u origin main This will remember that all pushes are done to the main branch and then we can just git push

Project – version control workflow

Create git hub repository >> clone to local >> changes >> add >> commit >> push

Git Branches


Feature
Commit 2 → commit 3→ commit 4
Commit A → commit B→ Commit C → Commit D→ Commit E
Master

- Generally, branches are created to separate a main code with extra features
- It also allows multiple devs to create multiple branches on multiple features they are working on
- We can also merge them into form a single project with all features
- Advantage is devs don't have to wait form other devs to complete, one can clone his own branch and later merge it to the main

Commands for branches

- Git branch : to check branch
- Git branch -M main : to rename a branch
- Git checkout -b <-new branch name-> : to create new branch
- Git branch -d <-branch name -> : to delete a branch
- Git diff <-branch name -> : to compare commits, branches, files and more
- Git merge <-branch name -> : to merge two branches into one
- git pull origin main : To pull changes to main branch from sub branches

Pull Requests to Merge Branches

- It lets you tell others about changes you've pushed to a branch in a repository on git hub
- This is helpful because generally the main branch will ben handled by a senior dev, and in the new branch, a junior may add
    - Add new feature and if we do pull request, the main branch leader can review and add comments and accept or reject
- Now the PR Request that we have done will be reflected in remote directory but will not appear in local directory
- In order to pull it to local directory we use git pull origin main

Resolving Merge Conflicts

- ✓ This event occurs when a git is unable to automatically resolve differences in code between two commits
- ✓ This occurs when we try to merge two branches using command git merge version2.0
- ✓ Because of this vs code will throw an error to resolve the conflict

Undoing Changes

Sometimes by mistake we may add a feature or commit a feature but we want it to reset it we have following commands for that

Commands

- ✓ Case 1 : git reset <-file_name-> : Staged reset (resets to original before commit)
- ✓ Case 2 : git reset HEAD ~1 : (This helps to reset after commit for one commit)
- ✓ Case 3 : git reset <-commit hash-> (multiple commits back)

  Git reset –hard <-commit hash -> (This is used because even after reset commit to previous the changes will be shown in vs

  Code as modified. But if we don't want that we can use – hard command before reset to make changes to vs code)

Fork

Fork is basically a rough copy of an exisiting repository

ALL GIT And GIT HUB Commands

Git config --global user.name “My_Name”	Set user name in git bash
Git config –global user.mail “github mail”	Set mail same as that of github account
Git config –list	Lists all the latest configuration
Git clone <-Repo_Link->	Clones repository from git hub
Git status	Check status of file changes
Git add . or git add <-file->	Adds modified file to staging area
Git commit -m “message for commiting”	Commits the final changes to local repo
Git push origin <-branch_name-> or main	Pushes the committed changes to remote repo
Git init	Makes a not git local repo to a git repo
Git remote add origin <- new-remote-repo-link->	Add new local repo to remote repo
Git remote -v	To check if remote repo exists
Git push -u origin <-branch_name->	Save to which branch we need to push everytime
Git branch	Check current working Branch
Git checkout -b <-New Branch Name->	Create new branch and enter the new branch
Git branch -M “rename”	Renames the current branch
Git branch -d <-branch_name->	Deletes the named branch
Git diff <-branch_name->	Compares two branches
Git merge <-Branch_Name->	Merges two branches into one (to a main branch)
Git pull origin main	After pull requests are accepted, this adds to local dir
Git reset HEAD~1	Resets the current to one previous commit
Git reset <-file_name->	Resets file to default during staged change
Git reset <-commit hash->	Resets to a past commit using the commits hash
Git reset –hard <-commit hash->	Hard resets to mentioned commit (no need to add or commit again to the main branch)