TS073_Shubham Pokale Setting up git

27 May 2024 11:22

A version control system (VCS) allows you to manage a collection of files and gives access to different versions of these files.

Git : Git is a distributed version control system and source code management system. It is designed to handle everything from small to very large projects with speed and efficiency. Git allows multiple developers to work on the same project without overwriting each other's changes, providing collaborative work and continuous integration and deployment.

Why git ?

- Version Control:
  - Git helps in tracking changes, allowing you to revert to previous states if something goes wrong.
- Collaboration:
  - It enables multiple developers to work on a project simultaneously without interfering with each other's work.
- Backup:
  - Your entire project history is saved in a Git repository, providing a backup of all versions.
- Branching and Merging:
  - Git's branching model allows you to experiment with new features or bug fixes independently from the main project.
- Open Source Projects:
  - Most open source projects use Git for version control. Learning Git allows you to contribute to these projects.
- Industry Standard:
  - Git is widely used in the software industry, making it an essential skill for developers.

Key terminologies while working with git :

- Git add
  - Moves changes from the working directory to the staging area. This gives you the opportunity to prepare a snapshot before committing it to the official history

- Git branch
  - This command is your general-purpose branch administration tool. It lets you create isolated development environments within a single repository.

- Git checkout
  - In addition to checking out old commits and old file revisions, git checkout is also the means to navigate existing branches. Combined with the basic Git commands, it's a way to work on a particular line of development.

- Git clean

    - Removes untracked files from the working directory. This is the logical counterpart to git reset, which (typically) only operates on tracked files.

- Git clone
    - Creates a copy of an existing Git repository. Cloning is the most common way for developers to obtain a working copy of a central repository.

- Git commit
    - Takes the staged snapshot and commits it to the project history. Combined with git add, this defines the basic workflow for all Git users.

- git fetch
    - Fetching downloads a branch from another repository, along with all of its associated commits and files. But, it doesn't try to integrate anything into your local repository. This gives you a chance to inspect changes before merging them with your project.

- git init
    - Initializes a new Git repository. If you want to place a project under revision control, this is the first command you need to learn.

- git log
    - Lets you explore the previous revisions of a project. It provides several formatting options for displaying committed snapshots.

- git merge
    - A powerful way to integrate changes from divergent branches. After forking the project history with git branch, git merge lets you put it back together again.

- git pull
    - Pulling is the automated version of git fetch. It downloads a branch from a remote repository, then immediately merges it into the current branch. This is the Git equivalent of svn update.

- git push
    - Pushing is the opposite of fetching (with a few caveats). It lets you move a local branch to another repository, which serves as a convenient way to publish contributions. This is like svn commit, but it sends a series of commits instead of a single changeset.

- git rebase
    - Rebasing lets you move branches around, which helps you avoid unnecessary merge commits. The resulting linear history is often much easier to understand and explore.

- git remote
    - A convenient tool for administering remote connections. Instead of passing the full URL to the fetch, pull, and push commands, it lets you use a more meaningful shortcut.

- git revert
    - Undoes a committed snapshot. When you discover a faulty commit, reverting is a safe and easy way to completely remove it from the code base.

- git status
    - Displays the state of the working directory and the staged snapshot. You'll want to run this in conjunction with git add and git commit to see exactly what's being included in the next snapshot

Step 1 : Initializing an empty git repository in an empty folder.

```
C:\Shubham Pokale\thinkbridge\Module 3>git init
Initialized empty Git repository in C:/Shubham Pokale/thinkbridge/Module 3/.git/
```

Step 2 : adding a text file called hello.txt in the folder

```
C:\Shubham Pokale\thinkbridge\Module 3>echo.> hello.txt
```

Step 3 : staging the changes in the staging area

```
C:\Shubham Pokale\thinkbridge\Module 3>git add .
```

Step 4 : Checking the status.

```
C:\Shubham Pokale\thinkbridge\Module 3>git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   hello.txt
```

Step 5 : committing the changes

```
C:\Shubham Pokale\thinkbridge\Module 3>git commit -m "added a new file"
[master (root-commit) 59020f5] added a new file
 1 file changed, 1 insertion(+)
 create mode 100644 hello.txt

C:\Shubham Pokale\thinkbridge\Module 3>
```

Step 6 : checking the logs for details of the commit

```
C:\Shubham Pokale\thinkbridge\Module 3>git log
commit 112d9942b200a82eb99f174e4599b7e607b10424 (HEAD -> master)
Author: Shubham Pokale <shubham.pokale2001@gmail.com>
Date:   Mon May 27 11:24:37 2024 +0530

    modified the the file hello.txt

commit 59020f58e93bb946946ee48dd7fdcc6370a624b9
Author: Shubham Pokale <shubham.pokale2001@gmail.com>
Date:   Mon May 27 11:22:11 2024 +0530

    added a new file

C:\Shubham Pokale\thinkbridge\Module 3>
```