Git (Day 07)

Assignment 1: Initialize a new Git repository in a directory of your choice. Add a simple text file to the repository and make the first commit.

Solution :-

1. git init :- Initializes an empty Git repository in the current directory.
2. git add :- Adds the specified file(s) to the staging area.
3. git commit :- Creates a commit snapshot of the files in the staging area with a descriptive message. It to specify the commit message directly on the command line.

Assignment 2: Branch Creation and Switching
Create a new branch named 'feature' and switch to it. Make changes in the 'feature' branch and commit them.

Solution :-
git checkout -b feature
This command utilizes the git checkout command with the -b flag for a shortcut.

1. git checkout :- It is the primary command which is used for switching branches in Git.
2. -b :- This flag tells git checkout to create new branch if it does not exist and then switch to it.
3. feature :- This specifies the name you want to give to the new branch.

Assignment 3: Feature Branches and Hotfixes
Create a 'hotfix' branch to fix an issue in the main code. Merge the 'hotfix' branch into 'main' ensuring that the issue is resolved.

Solution :-

1. Create the hotfix branch and switch to it:

git checkout -b hotfix-branch-name main

Explanation:

- **git checkout -b**: This creates a new branch named **hotfix-branch-name** based on the current branch (**main**) and switches to it.

2. Fix the issue in the 'hotfix' branch:

- Edit your code to address the identified issue in the main codebase.

3. Stage and commit the fix:

git add <modified_files> # Replace with actual modified files
git commit -m "Fix for the identified issue"

Explanation:

- **git add**: Add the modified files containing the fix to the staging area. Replace **<modified_files>** with the actual file names you changed.
- **git commit**: Create a commit with a descriptive message explaining the fix.

4. Verify the fix (optional):

- You can perform unit tests, integration tests, or manual testing to ensure the fix resolves the issue effectively.

5. Merge the 'hotfix' branch into 'main':

git checkout main
git merge hotfix-branch-name

Explanation:

- **git checkout main**: Switch back to the **main** branch.
- **git merge hotfix-branch-name**: Merge the changes from the **hotfix-branch-name** branch into the **main** branch.

6. Resolve merge conflicts (if any):

- If there are conflicts during the merge (meaning the same lines of code were modified in both branches), Git will pause and ask you to manually resolve the conflicts. Edit the conflicting files to choose the appropriate changes and then use **git add** to stage the resolved files. Finally, run **git commit** with a message describing the conflict resolution.

7. Push the changes (if using a remote repository):

git push origin main

Explanation:

- This command (assuming you're using GitHub or a similar platform) pushes the changes made to the **main** branch on your local machine to the remote repository. Replace **origin** with the actual name of your remote repository if it's different.