Name : Vaibhav kumar gupta

Date : 14-02-2025

Git and Github:-

1. Setting Up Git

Before using Git, you must configure your user details using `git config`. This helps track who made changes in a repository.

```
git config --global user.name "Your Name"
git config --global user.email "youremail@example.com"
```

2. Initializing a Repository

Creating a new Git repository to track changes in your project.

```
git init
```

3. Branching in Git

Branches allow developers to work on features independently without affecting the main branch.

- **Create a branch:** `git branch feature-branch`
- **Switch to a branch:** `git checkout feature-branch`
- **Create & switch:** `git checkout -b new-feature`
- **View all branches:** `git branch`

4. Committing Changes

After making changes, you must **stage** (`git add .`) and **commit** (`git commit -m "message"`) them to save progress.

```
git add .
git commit -m "Added new feature"
```

5. Pushing Changes

Push local commits to a remote GitHub repository.

```
git push  origin feature-branch
```

6. Pull Requests (PRs)

A **pull request** is a GitHub feature that lets you propose changes before merging them into the main branch.

Steps to create a PR:

1. Pu changes to GitHub.
2. Go to the repository and click **New Pull Request**.
3. Select the branches and submit the PR.
4. A reviewer checks the PR before merging.

7. Merging Branches

Combining a feature branch into the main branch.

```
git checkout main
git merge feature-branch
```

To delete a branch after merging:

```
git branch -d feature-branch
```

8. Forking a Repository

Forking is creating a personal copy of another user's repository to modify it without affecting the original project.

Steps:

1. Click **Fork** on GitHub.
2. Clone the forked repository to your local machine:

```
git clone https://github.com/yourusername/forked-repo.git
```

9. Resolving Merge Conflicts

Conflicts occur when changes from different branches cla . Git marks these conflicts in the file.

Fixing a conflict:

1. Open the conflicted file and resolve differences manually.
2. Stage and commit the resolved file:

```
git add resolved_file
git commit -m "Resolved merge conflict"
```

10. Undoing Changes

Unstage a file:

```
git reset filename
```

Undo the last commit but keep changes:

```
git reset --soft HEAD~1
```

Undo and discard changes:

```
git reset --hard HEAD
```

Revert a specific commit without losing history:

```
git revert <commit-ha >
```

11. Cloning a Repository

Download a GitHub repository to your local machine.

```
git clone https://github.com/user/repository.git
```

12. Stashing Changes

Temporarily save uncommitted changes without committing them.

```
git stash
```

Retrieve stashed changes:

```
git stash  pop
```

13. Checking Git History

View all previous commits in a repository.

```
git log
git log --oneline
```

14. Rolling Back Commits

Go back to a previous commit state.

Undo last commit (keep changes):

```
git reset --soft HEAD~1
```

Undo and remove changes:

```
git reset --hard HEAD~1
```

15. Using .gitignore

The `.gitignore` file prevents tracking unnecessary files (e.g., logs, system files, credentials).

Example `.gitignore`:

```
node_modules/
.env
*.log
```