**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.**

Ans.**1. Choose a Directory:**

* Think of a location on your computer where you'd like to keep your project files. This could be your Documents folder, a project-specific folder on your Desktop, or anywhere that makes sense for you.

**2. Open Terminal (or Command Prompt):**

* Navigate to the directory you chose in step 1 using the cd command in your terminal. For example, if your directory is on the Desktop named "my-project", use:

cd Desktop/my-project

**3. Initialize the Git Repository:**

* Run the following command in your terminal:

git init

This creates a new hidden folder named .git in your directory. This folder stores all the version control information for your project.

**4. Create a Text File:**

* You can use any text editor to create a new file inside your project directory. Let's call it "test.txt". Add some content to the file, like a simple sentence.

**5. Add the File to Git:**

* Use the following command in your terminal to tell Git to start tracking the "test.txt" file:

git add test.txt

This "stages" the file for your first commit.

**6. Commit the Changes:**

* Finally, run this command to create a snapshot of the current state of your project, including the "test.txt" file:

git commit -m "First commit - Added test.txt"

Replace "First commit - Added test.txt" with a more descriptive message about your changes. This message will be helpful when reviewing the history of your project later.

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

Ans.  **Verify Current Branch:**

* Make sure you're on the branch where you want to create the 'feature' branch. You can check the current branch using the following command:
* git branch

This will list all the local branches in your repository. The active branch will have an asterisk (\*) next to its name.

 **Create and Switch Branch:**

* To create a new branch named 'feature' and switch to it simultaneously, use this command:
* git checkout -b feature

The -b flag tells Git to create a new branch named 'feature' if it doesn't already exist. Otherwise, it will simply switch you to the existing 'feature' branch.

 **Make Changes:**

* Now that you're in the 'feature' branch, edit your files (like "test.txt" from the previous assignment) and make some changes specific to the feature you're working on.

 **Stage and Commit Changes:**

* Once you've made your changes, use the familiar Git commands to stage and commit them:
* git add <modified\_files> # Replace with actual modified files
* git commit -m "Meaningful message about your changes in feature 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.**

1. Ans. **Switch to Main Branch:**
   * Make sure you're on the 'main' branch where the issue exists. Use the following command to switch:
   * git checkout main
2. **Create Hotfix Branch:**
   * Create a new branch named 'hotfix' based on the current state of 'main':
   * git checkout -b hotfix
3. **Fix the Issue:**
   * Edit the relevant files to fix the bug or issue in your code.
4. **Commit the Fix:**
   * Stage the changes you made to fix the issue:
   * git add <modified\_files> # Replace with actual modified files
   * Commit the staged changes with a clear message describing the hotfix:
   * git commit -m "Hotfix: Fixed <issue description>"
5. **Verify the Fix (Optional):**
   * It's recommended to test your code locally to ensure the hotfix resolves the issue before merging.
6. **Merge Hotfix into Main:**
   * Switch back to the 'main' branch:
   * git checkout main
   * Merge the 'hotfix' branch into 'main'. There are two common ways to do this:

**Merge:**

git merge hotfix

This command integrates the commits from the 'hotfix' branch into the 'main' branch. If there are no conflicts (changes to the same lines of code), the merge will happen smoothly.

1. **Resolve Conflicts (if any):**
   * If there are conflicts during the merge (code changes in the same lines from both branches), Git will stop and ask you to resolve them manually. Edit the conflicting files to choose the desired changes and then use git add to stage the resolved conflicts. Finally, run git commit again to complete the merge process.
2. **Push Changes (Optional):**
   * If you're using Git version control on a platform like GitHub, you can push your changes to the remote repository after a successful merge:
   * git push origin main # Replace 'origin' with your remote repository name

This process creates a dedicated 'hotfix' branch to isolate the issue fix, allows you to verify the fix, and then integrates the fix back into the 'main' branch to resolve the issue in production.