Assignment 1 - 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.

Step 1 - Open a terminal window.
Step 2- Navigate to the directory where you want to initialize the Git repository using the `cd` command. For example:

cd /path/to/your/directory

Step 3 -Initialize a new Git repository using the `git init` command: for example -

git init

Step 4 - Create a simple text file in the directory. You can use any text editor you prefer. For example, you can create a file named `example.txt`:

touch example.txt

Step 5 - Add some content to the text file. You can use any text you like. For example, you can open `example.txt` in a text editor and write:

This is a simple text file for demonstration purposes.

Step 6 - Add the text file to the staging area using the `git add` command:

git add example.txt

Step 7 - Commit the changes to the repository using the `git commit` command:

git commit -m "Initial commit: Add example.txt"

In this command, `-m` is used to specify a commit message in quotes. Make sure to provide a meaningful message that describes the changes you've made in this commit.

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.

To create a new branch named 'feature' and switch to it, make changes, and commit them, follow these steps:

Step 1 - Create a new branch named 'feature' using the `git branch` command:

git branch feature

Step 2 - Switch to the 'feature' branch using the `git checkout` command:

git checkout feature

Step 3 - Make changes to the files in your working directory as needed. For example, you can open `example.txt` in a text editor and add some additional text.

Step 4 - After making changes, add the modified files to the staging area using the `git add` command:

git add example.txt

Step 5 - Commit the changes to the 'feature' branch using the `git commit` command:

git commit -m "Add additional content to example.txt in the feature branch"

Now, we've created a new branch named 'feature', switched to it, made changes, and committed them to the 'feature' branch. You can continue working on this branch, making additional changes, and committing them as needed.

Assignment 3 : 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.

To create a 'hotfix' branch to fix an issue in the main code, merge the 'hotfix' branch into 'main', ensuring that the issue is resolved, follow these steps:

Step 1 - Create a new branch named 'hotfix' from the 'main' branch using the `git checkout` command:

git checkout -b hotfix main

Step 2 - Make the necessary changes to fix the issue in your working directory.

Step 3 - After fixing the issue, add the modified files to the staging area using the `git add` command:

git add <modified_files>

Step 4 - Commit the changes to the 'hotfix' branch:

git commit -m "Fixed the issue in the hotfix branch"

Step 5 - Switch back to the 'main' branch:

git checkout main

Step 6 - Merge the 'hotfix' branch into the 'main' branch using the `git merge` command:

git merge hotfix

Step 7 - Verify that the issue is resolved in the 'main' branch.

Step 8 - Optionally, you can delete the 'hotfix' branch after merging it into 'main':

git branch -d hotfix

Now, the 'hotfix' branch has been merged into the 'main' branch, resolving the issue in the main codebase.