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 01:

Initialize a new Git repository: Navigate to the directory where you want to create the repository using the cd command, then initialize a new Git repository using git init.

```
cd /path/to/your/directory git init
```

Step 02:

Add a simple text file: Create a new text file in the directory. For example, you can create a file named example.txt using a text editor or by using command-line tools like touch:

```
touch example.txt
```

Step 03:

Add content to the text file: Open example.txt in a text editor and add some text. Add the file to the repository: Use git add to stage the file for commit.

```
git add example.txt
```

Step 04:

Make the first commit: Commit the staged changes using git commit.

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

Now you've initialized a new Git repository, added a simple text file, and made the first 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.

Step 01:

Create a new branch named 'feature' and switch to it:

```
git checkout -b feature
```

Step 02:

This command creates a new branch named 'feature' and switches to it. Alternatively, you can use separate commands to create the branch and switch to it:

```
git branch feature # Creates the branch
```

```
git checkout feature # Switches to the 'feature' branch
```

Step 03:

Make changes in the 'feature' branch

Step 04:

Make the desired changes to your files. For example, you can edit the example.txt file. Stage and commit the changes:

After making the changes, stage them using git add and then commit them:

```
git add example.txt # Stage the changes
```

```
git commit -m "Add feature in the 'feature' branch"
```

Now you've created a new branch named 'feature', made changes in it, and committed those changes.

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.

Step 01:

Create a 'hotfix' branch:

```
git checkout -b hotfix
```

This command creates a new branch named 'hotfix' and switches to it.

Fix the issue in the 'hotfix' branch:

Make the necessary changes to fix the issue in your codebase. This could involve editing existing files or adding new files.

Stage and commit the changes.

Step 02:

After fixing the issue, stage the changes and commit them:

```
git add <file(s) changed>
```

```
git commit -m "Fix issue in hotfix branch"
```

Step 03:

Switch back to the 'main' branch:

```
git checkout main
```

Step 04:

Merge the 'hotfix' branch into 'main':

```
git merge hotfix
```

This command merges the changes from the 'hotfix' branch into the 'main' branch. If there are no conflicts, Git will automatically perform the merge.

Resolve any merge conflicts (if applicable):

If there are conflicts during the merge, Git will pause the process and ask you to resolve them manually. After resolving the conflicts, you need to stage the changes and complete the merge by committing the merge result.

Verify that the issue is resolved:

After merging the 'hotfix' branch into 'main', test your code to ensure that the issue has been successfully resolved.

Now the 'hotfix' branch has been merged into 'main', and the issue should be resolved in the main codebase.