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.

To initialize a new Git repository, add a simple text file, and make the first commit, follow these steps:

Open Terminal/Command Prompt: First, open your terminal or command prompt.

Navigate to Your Desired Directory: Use the “cd” command to navigate to the directory where you want to initialize the new Git repository. If you want to create a new directory for this purpose, you can use the “mkdir” command followed by “cd” to move into the newly created directory.

```
mkdir my_new_repo
cd my_new_repo
```

Initialize the Git Repository: Run the following command to initialize a new Git repository in the current directory.

```
git init
```

Create a Simple Text File: You can use a text editor or a command line tool to create a simple text file. For example, using echo to create a file named example.txt with some content.

```
echo "This is a simple text file." >
sample.txt
```

Add the File to the Staging Area: Use the git add command to add the new file to the staging area.

```
git add sample.txt
```

Make the First Commit: Use the git commit command to commit the file to the repository. You can provide a commit message with the -m option.

```
git commit -m "Initial commit with sample.txt"
```

The total code is

```
mkdir my_new_repo

cd my_new_repo

git init

echo "This is a simple text file." > sample.txt

git add sample.txt

git commit -m "Initial commit with sample.txt"
```

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.

Now that you have a Git repository with a single commit, let's create a new branch and make some changes:

Create the 'feature' branch: Use the following command to create a new branch named "feature" and switch to it simultaneously:

```
git checkout -b feature
```

The -b flag tells git checkout to create a new branch named "feature" and then immediately switch to that branch.

Modify the 'test.txt' file: Open "test.txt" in your text editor and add some changes. For example, you could add a new line of text or modify the existing content.

Stage the changes: Once you've made your changes, run the following command to tell Git you want to include those changes in the next commit:

```
git add test.txt
```

Commit the changes: Finally, commit the changes you made to the "feature" branch with a descriptive message:

```
git commit -m "Added new content to test.txt in feature
branch"
```

This creates a new commit specifically on the "feature" branch, capturing the modified version of "test.txt".

Now you have a new branch named "feature" with its own independent history of commits. You can continue working on this branch and make further changes without affecting the main branch (usually called "master").

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, fix an issue, and merge the 'hotfix' branch into 'main', follow these steps:

Open Terminal/Command Prompt: Ensure you are in your repository directory and on the main branch.

Create and Switch to the 'hotfix' Branch: Use the git checkout -b command to create and switch to the 'hotfix' branch in one step.

```
git checkout -b hotfix
```

Make the Hotfix: Edit the necessary file(s) to fix the issue. For this example, let's add a fix to the example.txt file.

```
echo "Hotfix: Corrected an issue in the main code." >> example.txt
```

Add the Changes to the Staging Area: Use the git add command to add the modified file to the staging area.

```
git add example.txt
```

Commit the Changes: Use the git commit command to commit your changes with an appropriate commit message.

```
git commit -m "Applied hotfix to correct an issue in the main
code"
```

Switch Back to the 'main' Branch: Use the git checkout command to switch back to the 'main' branch. Alternatively, you can use git switch.

```
git checkout main
```

Merge the 'hotfix' Branch into 'main': Use the git merge command to merge the 'hotfix' branch into 'main'.

```
git merge hotfix
```

Resolve Any Merge Conflicts: If there are any merge conflicts, resolve them manually in your text editor, add the resolved files using git add, and then commit the merge.

Delete the 'hotfix' Branch (optional): After merging, you can delete the 'hotfix' branch if it is no longer needed.

```
git branch -d hotfix
```

Total code is :

```
git checkout -b hotfix

echo "Hotfix: Corrected an issue in the main code." >>
example.txt

git add example.txt

git commit -m "Applied hotfix to correct an issue in the main
code"

git checkout main

git merge hotfix

git branch -d hotfix
```