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.

Sol:

Step 1: Create a New Branch Named 'Feature'

Branches in Git allow you to diverge from the main codebase to work on separate features, fixes, or experiments. Creating a branch named `feature` will help isolate changes related to a new feature.

Command:

```
git branch feature
```

Explanation:

- **`git branch feature`:** This command creates a new branch named `feature` based on the current branch (usually `main` or `master`). The new branch is a copy of the current branch at the point of creation.

Step 2: Switch to the 'Feature' Branch

Switching to the `feature` branch allows you to start working on it. After switching, any changes you make will be applied to the `feature` branch.

Command:

```
git checkout feature
```

Explanation:

- **`git checkout feature`:** This command switches your working directory to the `feature` branch. It updates the files in your working directory to match the state of the `feature` branch.
- you can create and switch to a new branch in a single step using:

```
git checkout -b feature
```

Explanation:

- **`git checkout -b feature`:** This command creates a new branch named `feature` and then switches to it immediately.

Step 3: Make Changes in the 'Feature' Branch

Now that you are on the **`feature`** branch, you can make changes specific to the new feature. For example, you can modify an existing file or create a new one.

Command:

```
echo "This is a new feature." >> file.txt
```

Explanation:

- **`echo "This is a new feature." >> file.txt`:** This command appends the text "This is a new feature." to the end of **`file.txt`.** The >> operator appends the output to the file rather than overwriting it.

Step 4: Add the Changes to the Staging Area

Before committing the changes, you need to add them to the staging area.

Command:

```
git add file.txt
```

Explanation:

- **`git add file.txt`:** This command stages the changes in **`file.txt`** for the next commit. You can also add multiple files or use **`git add.`** to stage all changes in the directory.

Step 5: Commit the Changes

Commit the staged changes to the **`feature`** branch. A commit captures the state of the staged changes and records them in the branch's history.

Command:

```
git commit -m "Added a new feature to file.txt"
```

Explanation:

- **`git commit -m "Added a new feature to file.txt"`**: This command creates a new commit with the message "Added a new feature to file.txt". The commit message should clearly describe the changes made.