**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.**

**Sol:**

**Commands :**

git checkout main

git branch hotfix

git checkout hotfix

echo "This is a hotfix." >> file.txt

git add file.txt

git commit -m "Applied hotfix to file.txt"

git checkout main

git merge hotfix

cat file.txt

git branch -d hotfix

==========

**Step 1: Create a 'Hotfix' Branch**

Creating a 'hotfix' branch allows you to address and resolve issues in the main codebase without disrupting ongoing feature development.

**Command**:

```
git checkout main
git branch hotfix
```

**Explanation**:

- **`git checkout main`:** This command switches your working directory to the `main` branch, ensuring that the hotfix will be based on the latest state of the main code.
- **`git branch hotfix`:** This command creates a new branch named `hotfix` from the current `main` branch.

Alternatively, you can create and switch to the new branch in a single step:

```
git checkout -b hotfix main
```

**Explanation**:

- **`git checkout -b hotfix main`:** This command creates a new branch named `hotfix` and switches to it immediately, based on the `main` branch.

**Step 2: Make Changes in the 'Hotfix' Branch**

Now that you are on the `hotfix` branch, you can make the necessary changes to fix the issue. For example, you might modify an existing file to resolve a bug.

**Command**:

```
echo "This is a hotfix." >> file.txt
```

**Explanation**:

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

**Step 3: Add and Commit the Changes**

After making the necessary changes, add them to the staging area and commit them.

**Commands**:

```
git add file.txt
git commit -m "Applied hotfix to file.txt"
```

**Explanation**:

- **`git add file.txt`:** This command stages the changes in `file.txt` for the next commit.
- **`git commit -m "Applied hotfix to file.txt"`:** This command creates a new commit with the message "Applied hotfix to file.txt". The commit message should clearly describe the hotfix applied.

**Step 4: Switch Back to the 'Main' Branch**

Before merging the hotfix, switch back to the `main` branch.

**Command**:

```
git checkout main
```

**Explanation**:

- **`git checkout main`:** This command switches your working directory back to the `main` branch.

**Step 5: Merge the 'Hotfix' Branch into 'Main'**

Merge the `hotfix` branch into the `main` branch to incorporate the fix.

**Command**:

```
git merge hotfix
```

**Explanation**:

- **`git merge hotfix`:** This command merges the `hotfix` branch into the `main` branch, integrating the changes made in the hotfix.

**Step 6: Verify the Issue is Resolved**

After merging, verify that the issue is resolved by checking the changes in the `main` branch.

**Command**:

```
cat file.txt
```

**Explanation**:

- **`cat file.txt`:** This command displays the contents of `file.txt` so you can verify that the hotfix is present and the issue is resolved.

**Step 7: Optionally Delete the 'Hotfix' Branch**

Once the hotfix is successfully merged and verified, you can delete the `hotfix` branch as it is no longer needed.

**Command**:

```
git branch -d hotfix
```

**Explanation**:

- **`git branch -d hotfix`:** This command deletes the `hotfix` branch, cleaning up your branch list