Gitflow Workflow

By: Hashim Emad

Video: https://youtu.be/AMIwlP8YOMw

Email: hashim.emad966@gmail.com

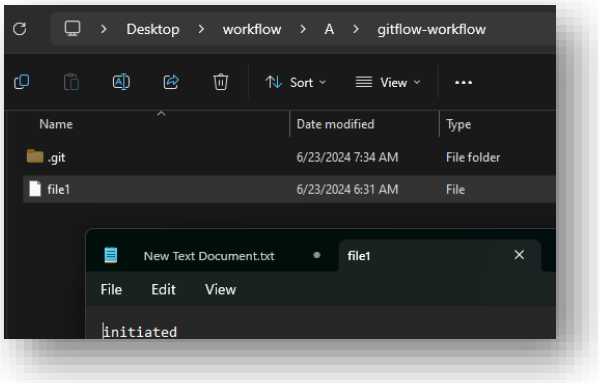
Objectives:

- Simulate a collaborative development environment using Git
- Implement a Gitflow workflow
- Use multiple collaborators
- Implementing a hotfix and a release
- Using a main, developer and a feature branch.

Introduction

This report documents the implementation of a collaborative Git workflow in a simulated development environment. The primary objective was to understand and apply Git for version control using a simple project developed with basic text files. This exercise aimed to simulate a real-world development environment and enhance practical knowledge of version control systems in a team setting.

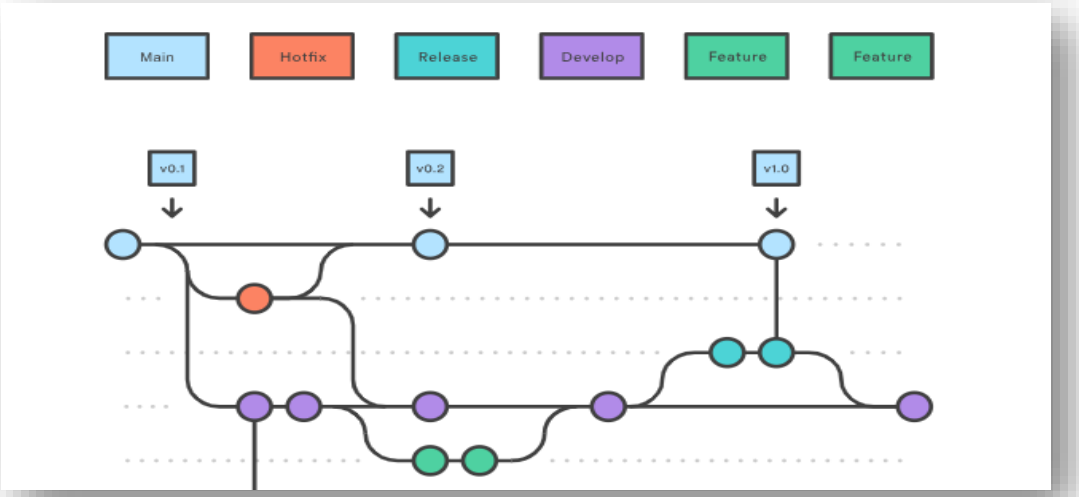
Project Description



The project consisted of simple text files to demonstrate the Git workflow. Initially, the repository contained a single file, file1.txt, with the text "initiated". Two collaborators, Collaborator A and Collaborator B, were involved in this project. Collaborator A was responsible for hotfixes and had the privilege to edit the main branch of the remote repository, while Collaborator B worked on the develop, feature, and release branches.

Workflow Diagram

Below is a diagram representing the Git workflow used in this project. This visual representation illustrates the branching and merging activities, showing how changes flow from feature development and hotfixes into the main and develop branches.

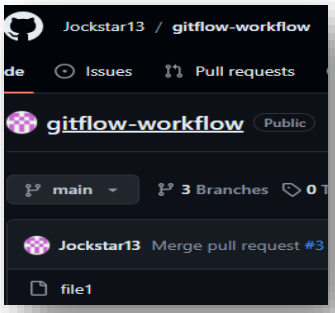


Explanation of the Workflow Diagram

1. **Initial Commit**:
    - The initial commit is made to the main branch with the file file1.txt containing the text "initiated".
2. **Hotfix Branch**:
    - A hotfix branch is created from the main branch by Collaborator A.
    - Changes are made to file1.txt to include "hotfixes by A".
    - The hotfix branch is merged back into both the main and develop branches to ensure consistency.
3. **Feature Branch**:
    - A feature branch is created from the develop branch by Collaborator B.
    - A new file file2.txt is added with the text "added sidebar feature by B".
    - The feature branch is merged back into the develop branch.
4. **Release Branch**:
    - A release branch is created from the develop branch by Collaborator B.
    - Bug fixes are made to file2.txt, adding "bug fixes by B".
    - The release branch is merged into the main branch for production release.
    - The changes from the release branch are also merged back into the develop branch to ensure it stays up-to-date.

Gitflow Workflow Implementation

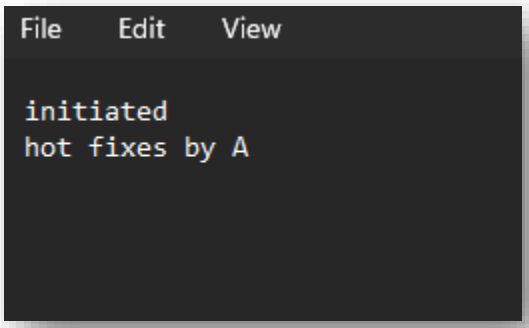
Repository Setup



The project repository was created on GitHub and cloned to local directories simulating different collaborators. The repository was initialized and the initial commit was made by Collaborator A.

Hotfix by Collaborator A

Collaborator A performed a hotfix on file1.txt in his local repository by adding a second line:

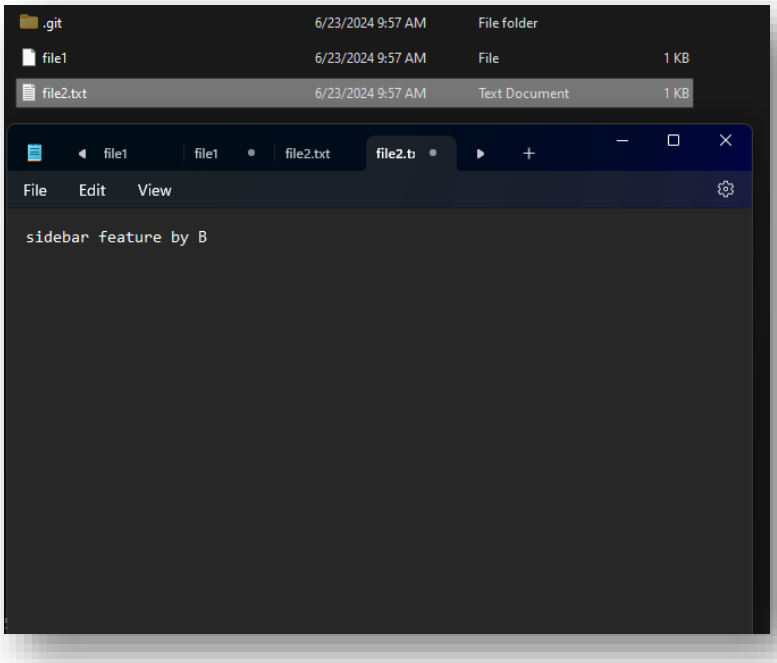


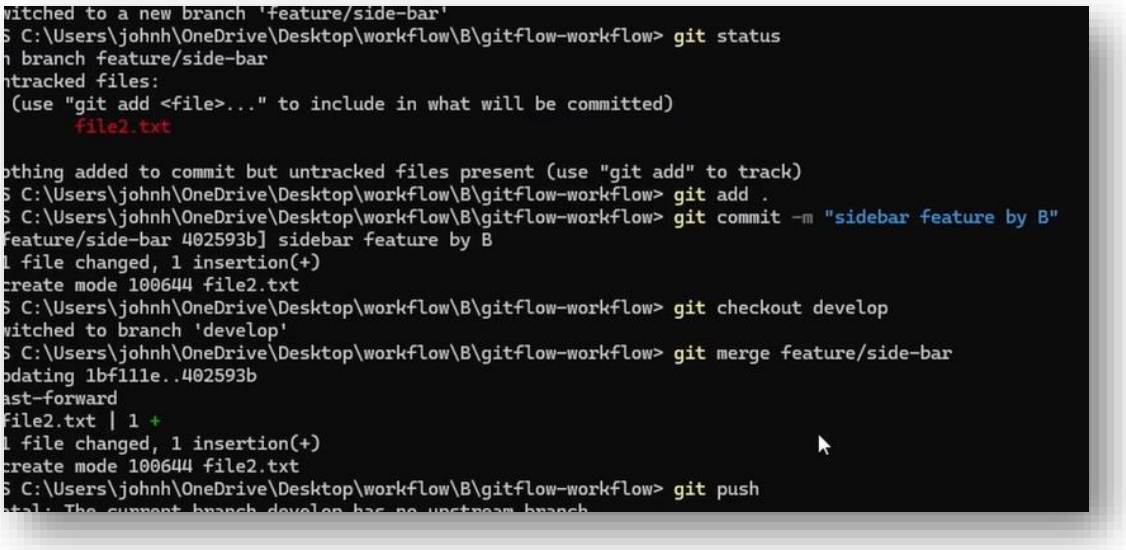
```
git checkout -b hotfix-1
echo "hotfixes by A" >> file1.txt
git add file1.txt
git commit -m "Added hotfixes by A"
git checkout main
git merge hotfix-1
git push origin main
git checkout develop
git merge hotfix-1
git push origin develop
```

The changes were committed and pushed to a hotfix branch, then merged into both the main and develop branches.

Feature Branch by Collaborator B

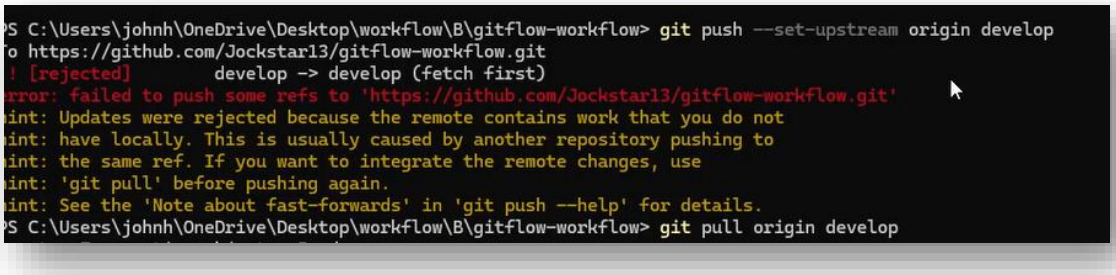
Collaborator B created a feature branch to add a new file, file2.txt, with the text "added sidebar feature by B":





Collaborator B merged feature branch to the develop branch then attempted to push the develop branch to the remote repository.

However....



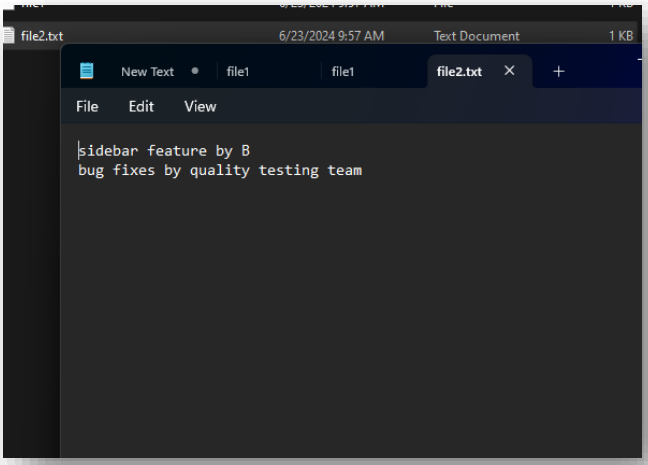
the push was unsuccessful because Collaborator B had not pulled the latest changes (the hotfix) from the remote repository.

Collaborator B pulled the updated remote repo then pushed his changes.

Release Branch by Collaborator B

After successfully updating the develop branch, Collaborator B created a release branch to make further updates to file2.txt:

```
PS C:\Users\johnh\OneDrive\Desktop\workflow\B\gitflow-workflow> git checkout develop
Already on 'develop'
Your branch is up to date with 'origin/develop'.
PS C:\Users\johnh\OneDrive\Desktop\workflow\B\gitflow-workflow> git checkout -b release/sprint1
```



Collaborator B added the bug fixes for the release.

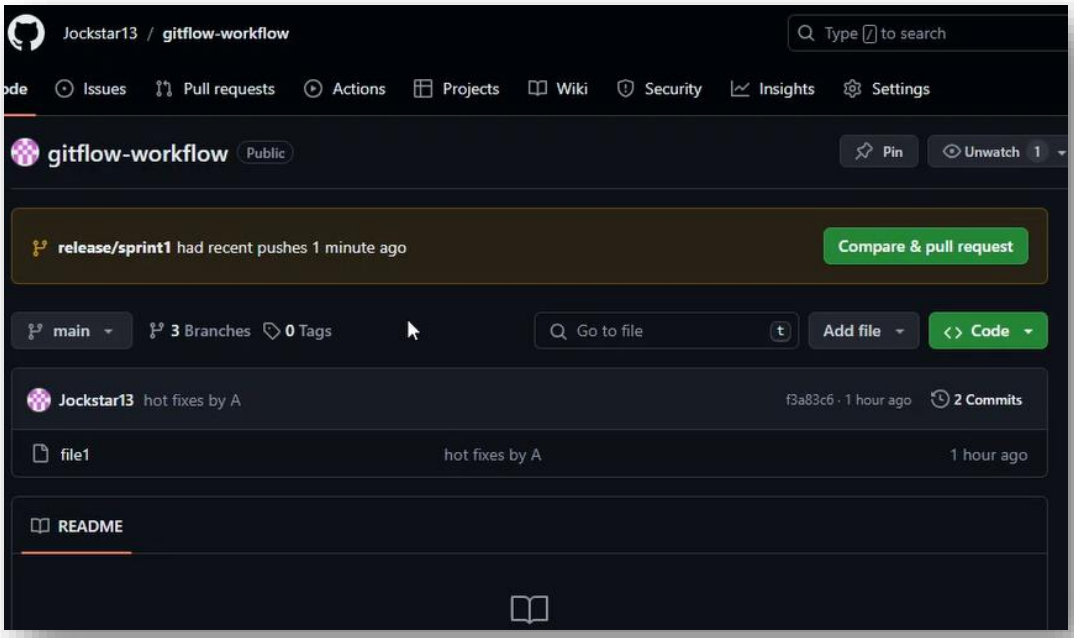
```
Switched to a new branch 'release/sprint1'
PS C:\Users\johnh\OneDrive\Desktop\workflow\B\gitflow-workflow> git status
On branch release/sprint1
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   file2.txt

no changes added to commit (use "git add" and/or "git commit -a")
PS C:\Users\johnh\OneDrive\Desktop\workflow\B\gitflow-workflow> git commit -am "bug fixes"
[release/sprint1 abb2c68] bug fixes
 1 file changed, 2 insertions(+), 1 deletion(-)
PS C:\Users\johnh\OneDrive\Desktop\workflow\B\gitflow-workflow> git push --set-upstream origin release/sprint1
```

Then collaberator B pushed the changes to the remote repo.

Merging Changes into Main Branch

In Github Collaborator A reviewed the pull request and merged the changes from the release branch into the main branch.

Collaborator A then updated the develop branch to include the release changes:



Challenges and Solutions

Challenges

1. **Branch Management**: Keeping track of multiple branches simultaneously was initially challenging.
2. **Merge Conflicts**: During the integration of features, merge conflicts occurred when Collaborator B did not pull the latest changes before pushing.

Solutions

1. **Branch Naming Conventions**: Using clear and consistent branch naming conventions helped in managing multiple branches.
2. **Regular Pulls**: Regularly pulling updates from the remote repository helped minimize merge conflicts.

Learning Experience and Insights

Implementing Git in a simulated team environment provided valuable insights into the workflow's efficiency and organizational benefits. Key takeaways include:

- **Structured Workflow**: Git offers a clear structure for managing feature development, releases, and hotfixes.
- **Collaboration**: It facilitates collaboration by allowing multiple team members to work on different features simultaneously without interfering with each other's progress.
- **Version Control**: The separation of main and development branches ensures that production code remains stable.

Conclusion

The Gitflow workflow significantly enhances version control and team collaboration in software development. This project demonstrated the practical application of Git, highlighting its benefits and challenges. The experience gained will be invaluable for applying Git in real-world scenarios, ensuring efficient and organized development processes.