Karel The Robot Assignment

Atypon Java and DevOps Cohort / June 2024

Mahmoud J. Qudah

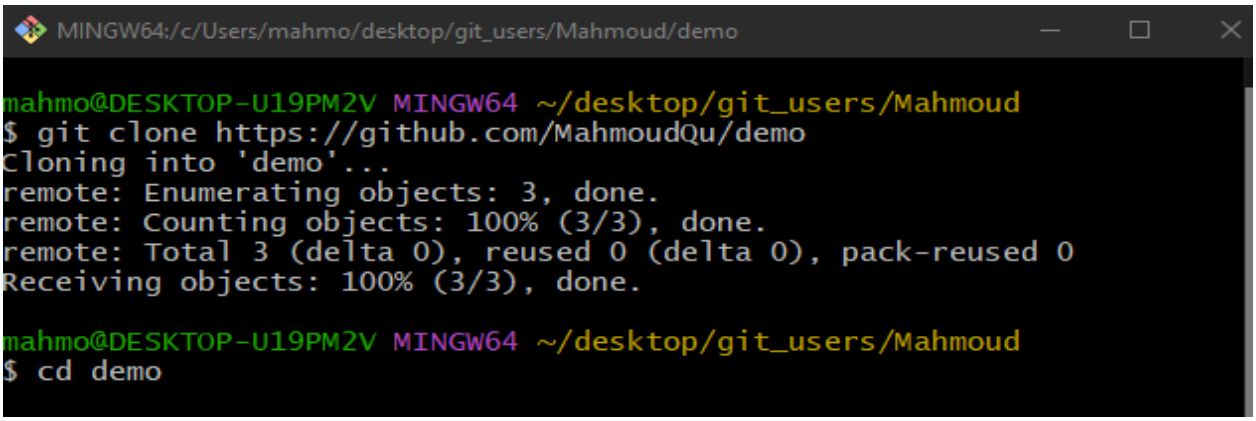
Introduction:

In this assignment, I was asked to simulate a collaborative development environment using Git and GitHub. By emulating two different users working on the same repository, I gained practical experience in using Git as a collaborative tool. This involved creating and managing branches, merging feature branches into a development branch, performing a release, and one hotfix, all while committing to the Gitflow workflow. This hands-on approach provided valuable insights into the processes and best practices of version control in a real-world development setting.

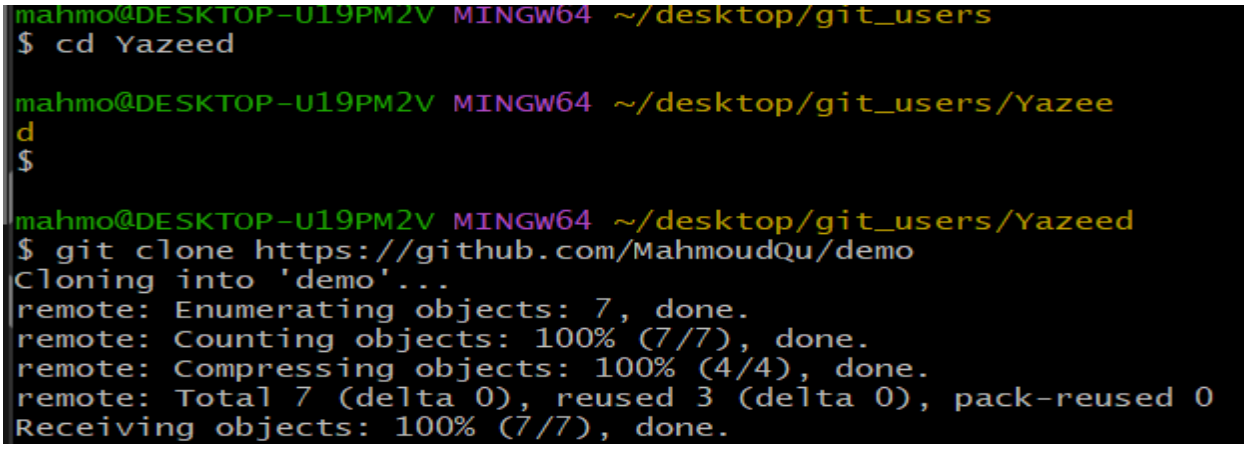
Project description:

First, I created two directories named “Mahmoud” and “Yazeed” to simulate two different users. I then cloned the Git repository into each directory, allowing each simulated user to work independently on their own feature branches.

- Mahmoud:



- Yazeed:



For Mahmoud:

After cloning the repository, I created a “develop” branch from the main branch and pushed it to the remote repository. This develop branch serves as the base for feature branches, ensuring that new features are integrated into develop before being merged into the main branch, this maintaining a clean and stable main branch.

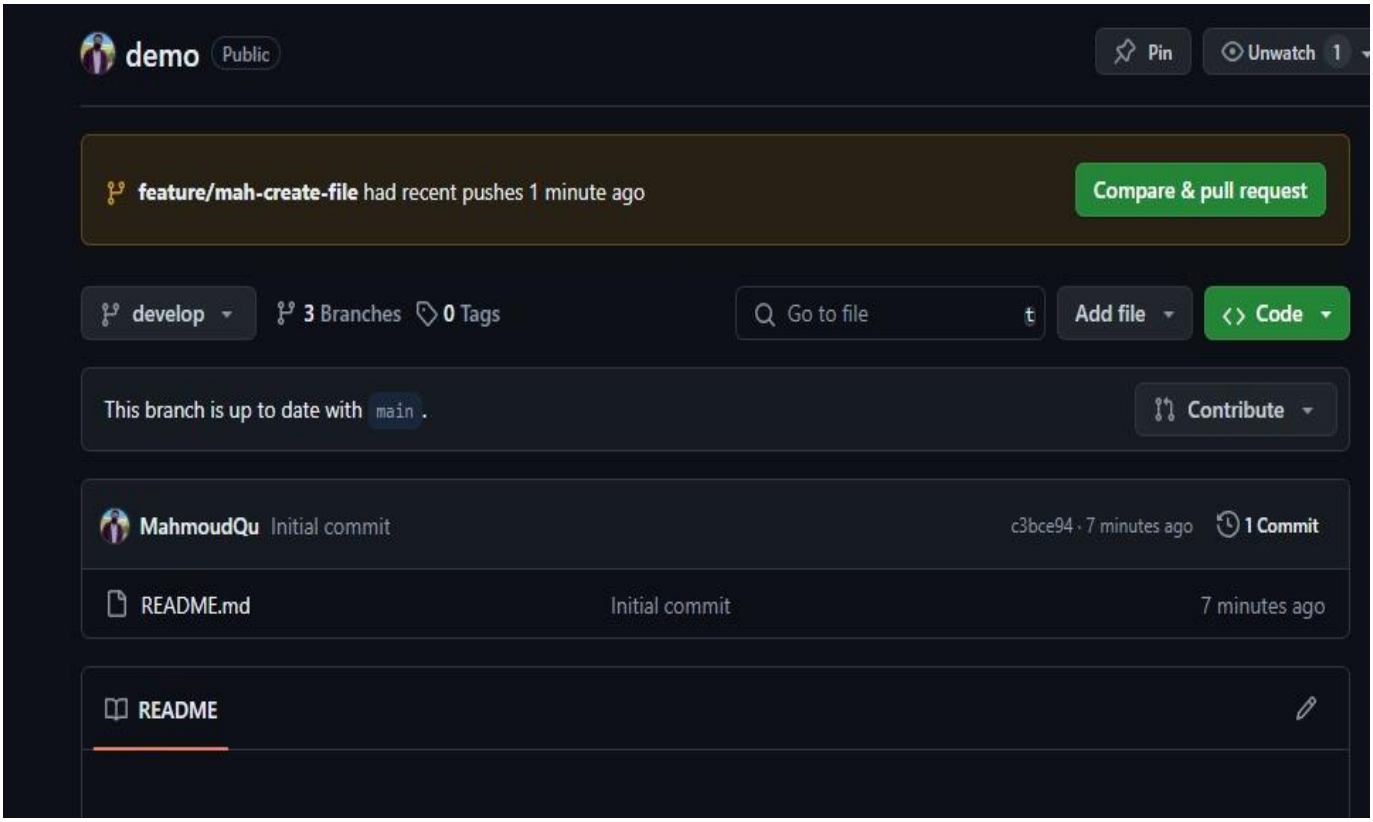
```
mahmo@DESKTOP-U19PM2V MINGW64 ~/desktop/git_users/Mahmoud/demo (main)
$ git checkout -b develop
Switched to a new branch 'develop'

mahmo@DESKTOP-U19PM2V MINGW64 ~/desktop/git_users/Mahmoud/demo (develop)
$ ls
README.md

mahmo@DESKTOP-U19PM2V MINGW64 ~/desktop/git_users/Mahmoud/demo (develop)
$ git push origin main
Everything up-to-date

mahmo@DESKTOP-U19PM2V MINGW64 ~/desktop/git_users/Mahmoud/demo (develop)
$ git push origin develop
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: Create a pull request for 'develop' on GitHub by visiting:
remote:      https://github.com/MahmoudQu/demo/pull/new/develop
remote:
To https://github.com/MahmoudQu/demo
 * [new branch]      develop -> develop
```

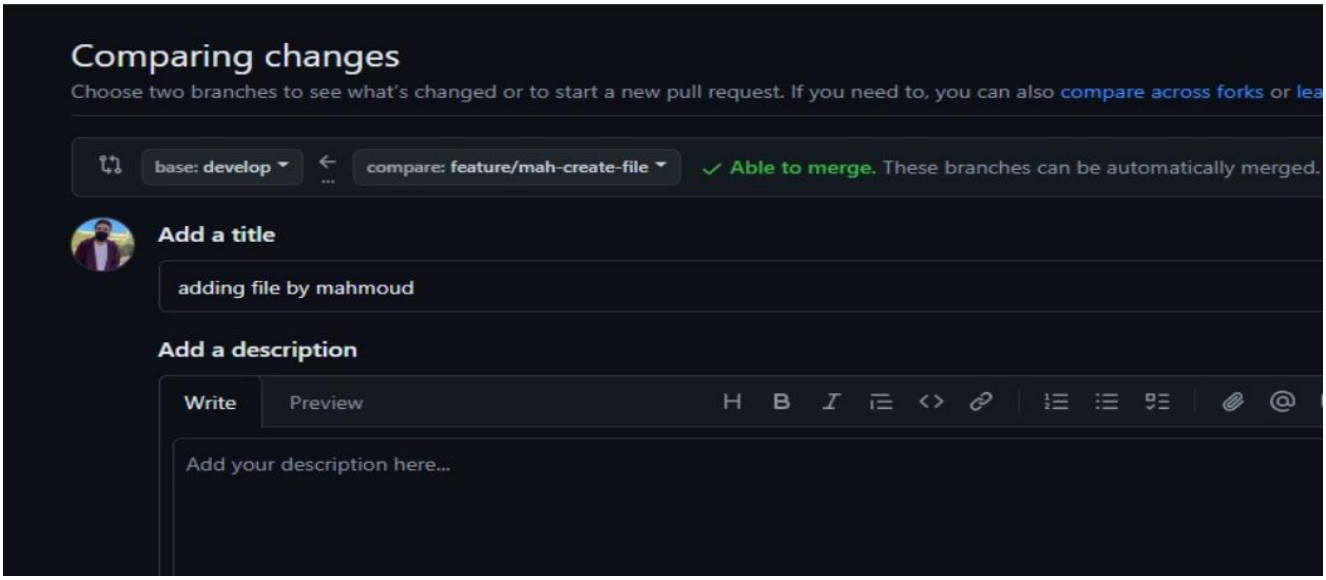
After ensuring that the develop branch was up to date on the server, I created a new branch named “feature/mah-create-file” from the develop branch. This branch was dedicated to the "adding file" feature, indicating that Mahmoud created a new file. I then pushed this feature branch to the remote develop branch, keeping the main branch clean and stable.

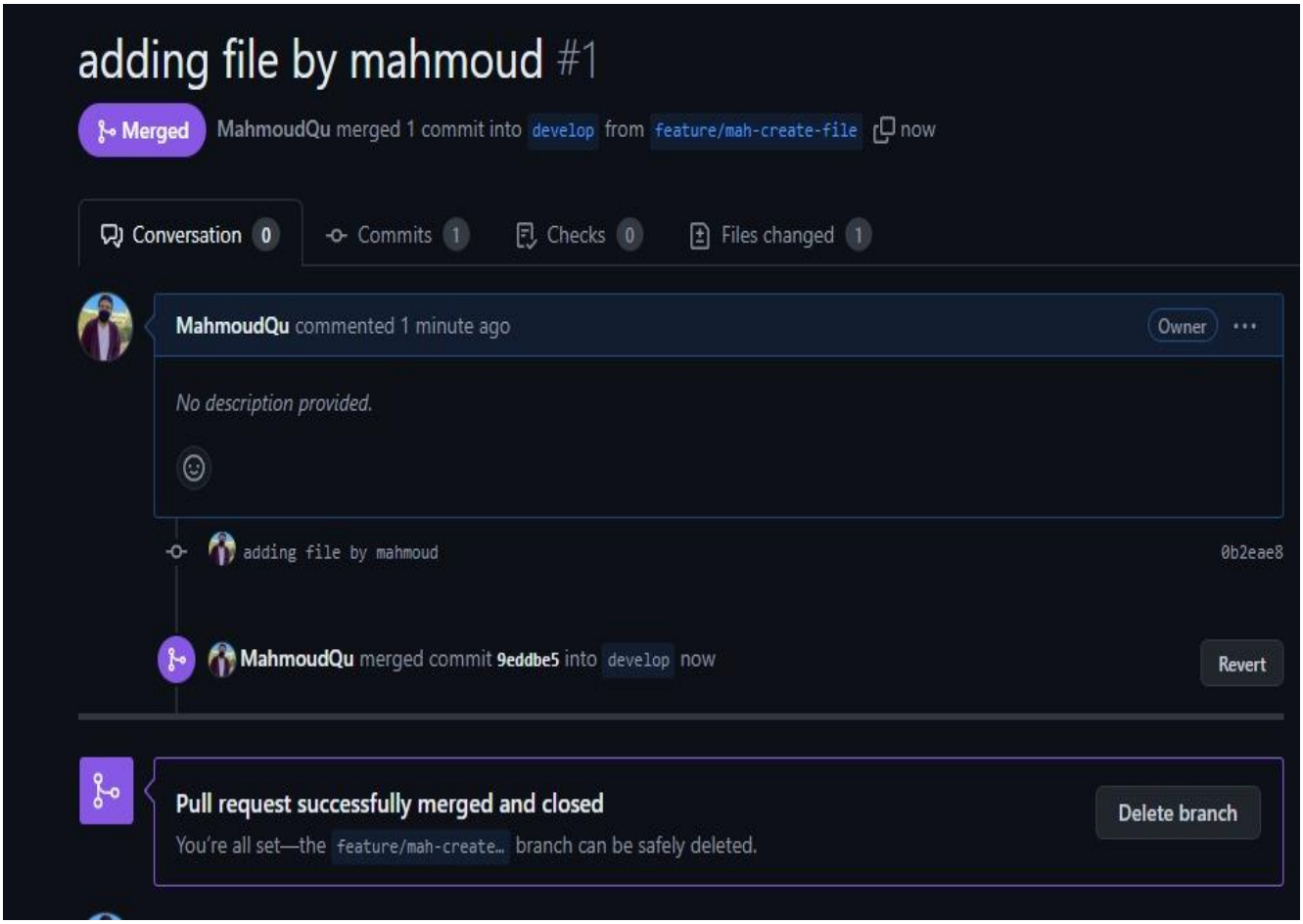
```
mahmo@DESKTOP-U19PM2V MINGW64 ~/desktop/git_users/Mahmoud/demo (develop)
$ git checkout -b feature/mah-create-file
Switched to a new branch 'feature/mah-create-file'

mahmo@DESKTOP-U19PM2V MINGW64 ~/desktop/git_users/Mahmoud/demo (feature/m
ah-create-file)
$ vi fileM.txt

mahmo@DESKTOP-U19PM2V MINGW64 ~/desktop/git_users/Mahmoud/demo (feature/m
ah-create-file)
$ git add fileM.txt
warning: in the working copy of 'fileM.txt', LF will be replaced by CRLF
the next time Git touches it

mahmo@DESKTOP-U19PM2V MINGW64 ~/desktop/git_users/Mahmoud/demo (feature/m
ah-create-file)
$ git commit -m "adding file by mahmoud"
[feature/mah-create-file 0697d1a] adding file by mahmoud
 1 file changed, 1 insertion(+)
 create mode 100644 fileM.txt
```

```
mahmo@DESKTOP-U19PM2V MINGW64 ~/desktop/git_users/Mahmoud/demo (feature/mah-create-file)
$ git push origin develop feature/mah-create-file
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 320 bytes | 320.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: Create a pull request for 'feature/mah-create-file' on GitHub by visiting:
remote:      https://github.com/MahmoudQu/demo/pull/new/feature/mah-create-file
remote:
To https://github.com/MahmoudQu/demo
 * [new branch]      feature/mah-create-file -> feature/mah-create-file
```

After pushing, now the remote repo has 3 branches, main, develop and feature/mah-create-file, and there is a pull request here from the feature branch, after reviewing this pull request and deciding by the one who is responsible for this, lets assume it is confirmed







The feature branch from Mahmoud is Successfully merged with develop branch!

Challenges and Solutions:

Challenge 1: Accidental Pushes to the Main Branch

I had trouble because I accidentally pushed changes to the main branch instead of the develop branch. This happened because I didn't specify the branch when pushing changes, which could make the main branch unstable.

Solution:

To fix this, I used this command to make sure the changes went to the right branch:

git push origin develop

As shown in the images above, this command explicitly specifies both the remote repository (origin) and the target branch (develop), preventing accidental pushes to the main branch. This approach helped maintain the integrity of the main branch by ensuring that all new features and changes were first integrated into the develop branch.

Challenge 2: Keeping Branches Updated

Another challenge was keeping the local branches updated with the latest changes from the remote repository, especially when multiple collaborators were working on the same project. This required frequent synchronization to avoid conflicts and ensure smooth integration.

Solution: I regularly used the git pull command to fetch and merge changes from the remote repository into my local branches. For example: **git pull origin develop,**
this command fetched the latest updates from the develop branch and merged them into my local develop branch, ensuring that I was working with the most recent codebase.

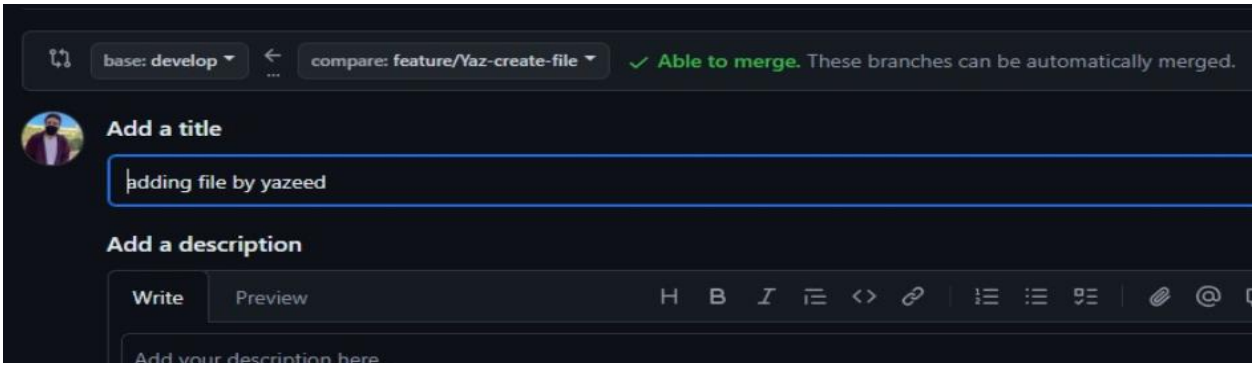
For Yazeed:

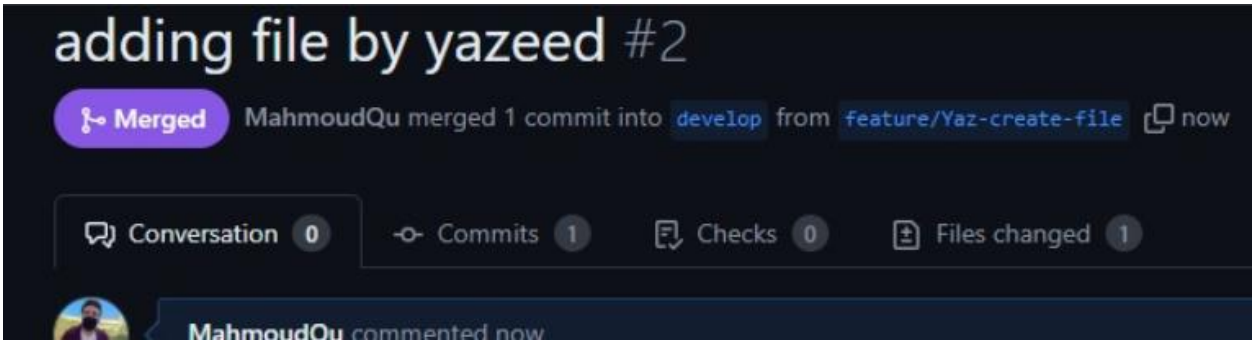
Applying the same steps as for Mahmoud, after cloning the repo and the switching the develop branch, now Yazeed can see the added feature by Mahmoud, which is “fileM.txt”:

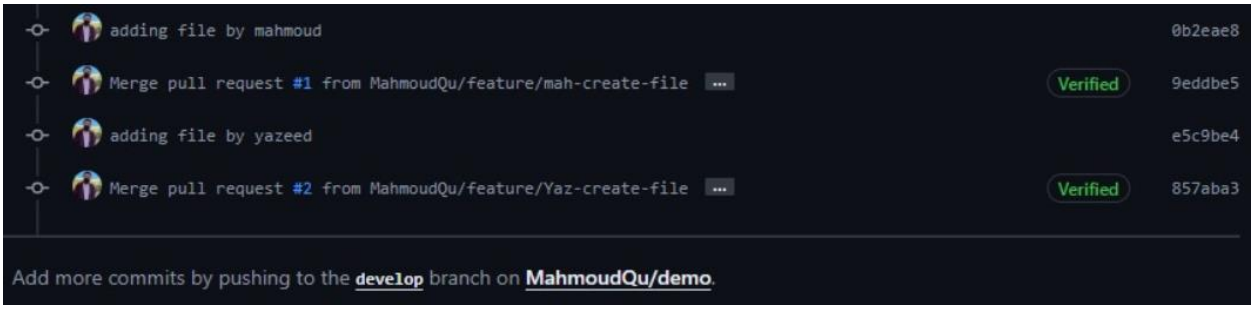
```
mahmo@DESKTOP-U19PM2V MINGW64 ~/desktop/git_users/Yazeed/demo (main)
$ ls
README.md

mahmo@DESKTOP-U19PM2V MINGW64 ~/desktop/git_users/Yazeed/demo (main)
$ git checkout develop
Switched to a new branch 'develop'
branch 'develop' set up to track 'origin/develop'.

mahmo@DESKTOP-U19PM2V MINGW64 ~/desktop/git_users/Yazeed/demo (develop
$ ls
README.md  fileM.txt

mahmo@DESKTOP-U19PM2V MINGW64 ~/desktop/git_users/Yazeed/demo (develop
$ git branch
* develop
  main
```

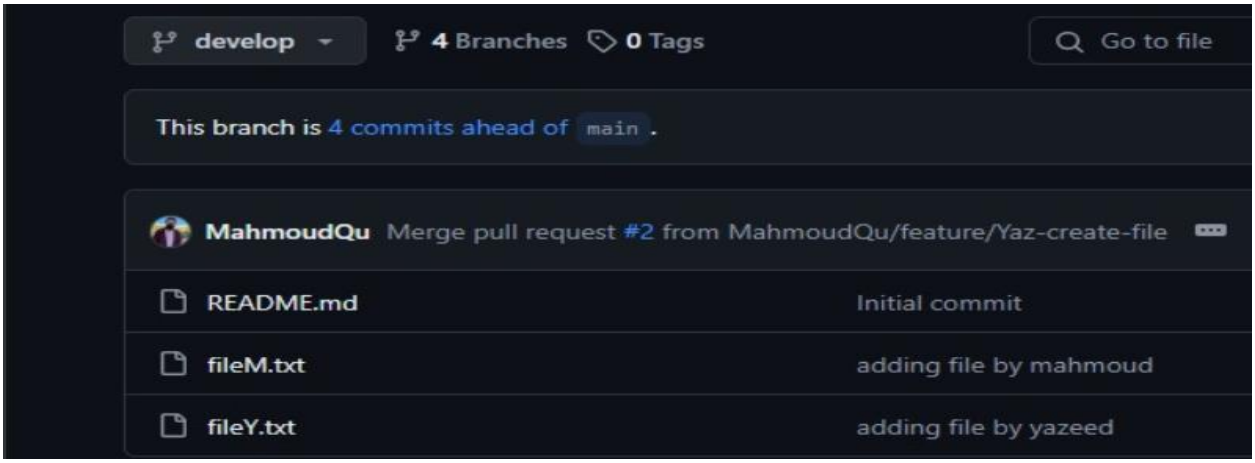
Now, I cut a new branch from develop named “feature/yaz-create-file” to add “fileY.txt”, and then pushed this branch into the server to the develop branch,

```
mahmo@DESKTOP-U19PM2V MINGW64 ~/desktop/git_users/Yazeed/demo (feature
reate-file)
$ git push origin develop feature/Yaz-create-file
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 346 bytes | 346.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: Create a pull request for 'feature/Yaz-create-file' on GitHub
iting:
remote:      https://github.com/MahmoudQu/demo/pull/new/feature/Yaz-cr
ile
remote:
To https://github.com/MahmoudQu/demo
 * [new branch]      feature/Yaz-create-file -> feature/Yaz-create-fil
```





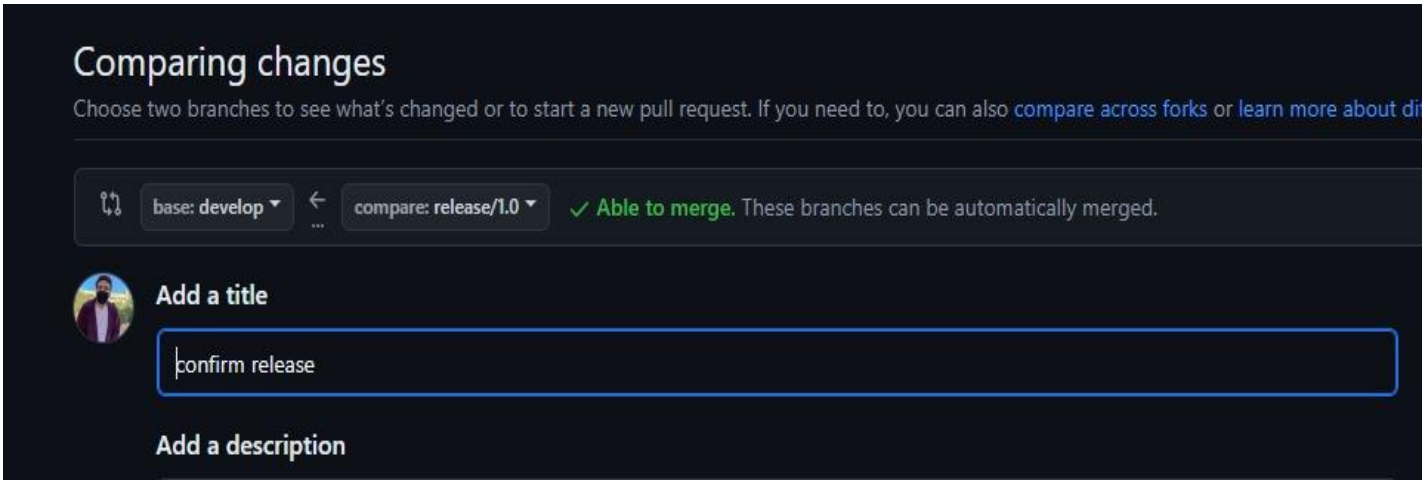


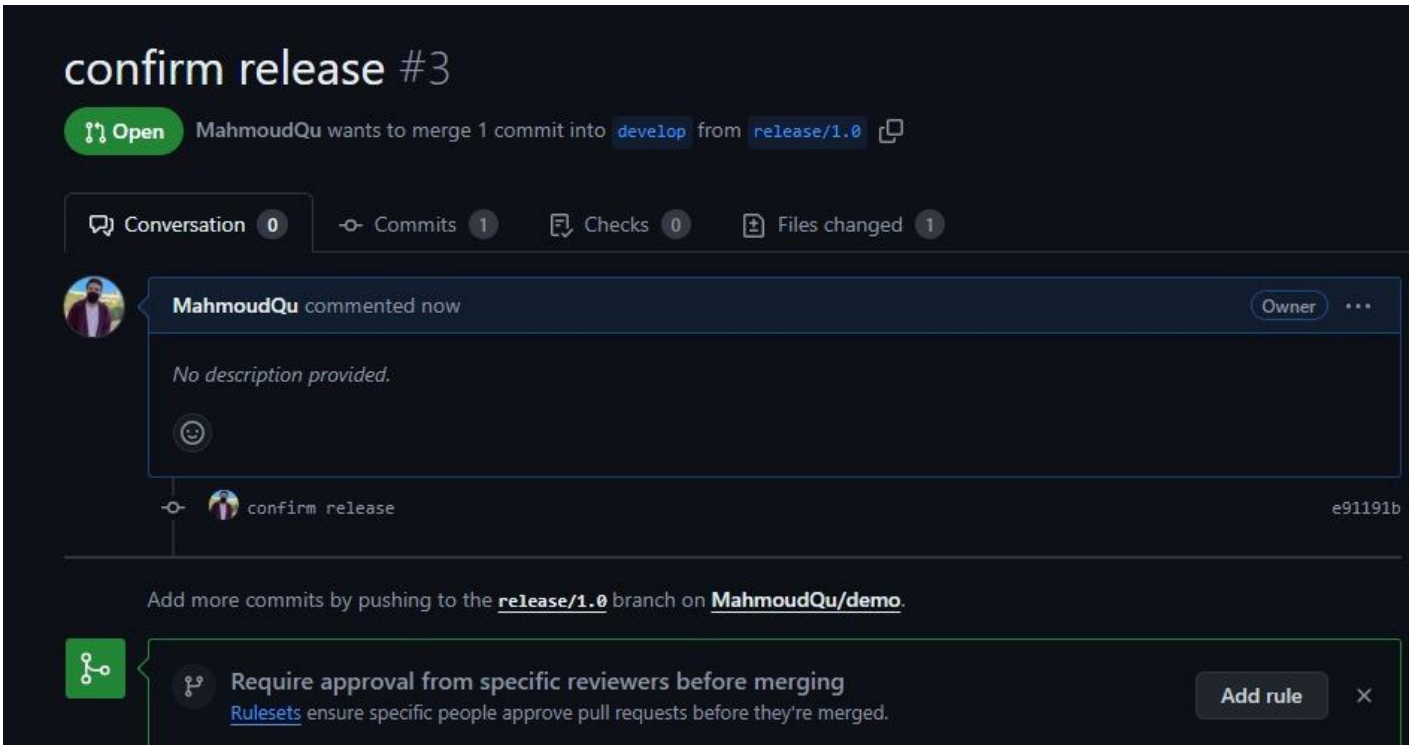


After confirming the pull request, we merged the feature branch into the development branch, and now all features have been pushed.

Release:

Mahmoud created the release to finalize the decision to push into production after reviewing the work done so far. Here is the process: from the develop branch, I created the release/1.0 branch to make the final changes. I added a file named 'finalFile.txt' to simulate the release work and then pushed it to the server. After that, I merged the release branch with the main branch, as this is the best practice to avoid directly merging with the main branch.

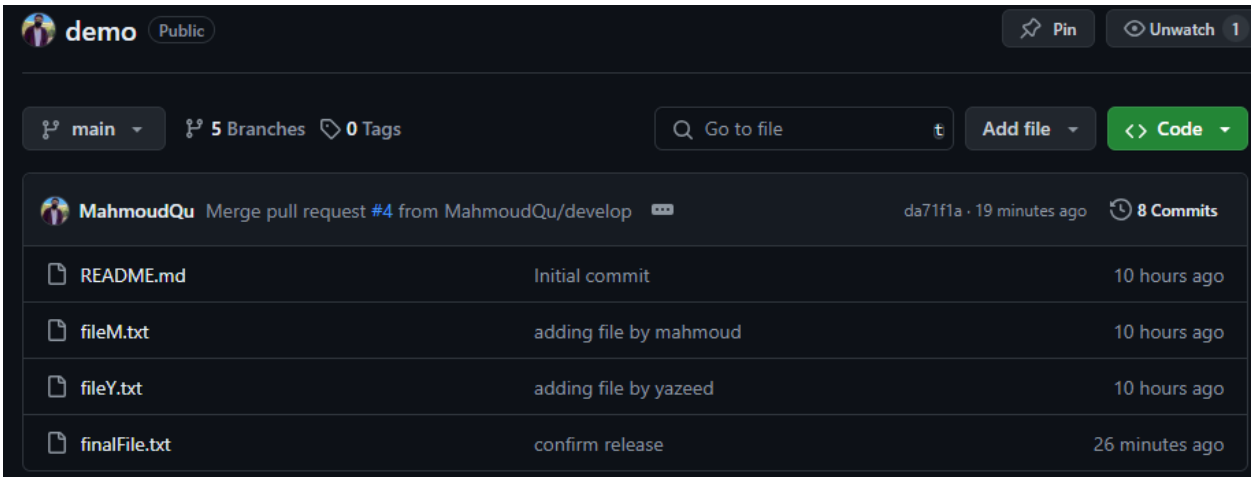




And then to make all consistant, I merge the develop branch from the release/1.0, doing this for all the features branching.

```
mahmo@DESKTOP-U19PM2V MINGW64 ~/desktop/git_users/Mahmoud/demo (develop)
$ ls
README.md  fileM.txt  fileY.txt

mahmo@DESKTOP-U19PM2V MINGW64 ~/desktop/git_users/Mahmoud/demo (develop)
$ git merge release/1.0
Updating 0a71826..e91191b
Fast-forward
 finalFile.txt | 1 +
 1 file changed, 1 insertion(+)
 create mode 100644 finalFile.txt

mahmo@DESKTOP-U19PM2V MINGW64 ~/desktop/git_users/Mahmoud/demo (develop)
$ git push origin main
Everything up-to-date

mahmo@DESKTOP-U19PM2V MINGW64 ~/desktop/git_users/Mahmoud/demo (develop)
$ ls
README.md  fileM.txt  fileY.txt  finalFile.txt
```

Main branch after merging release:

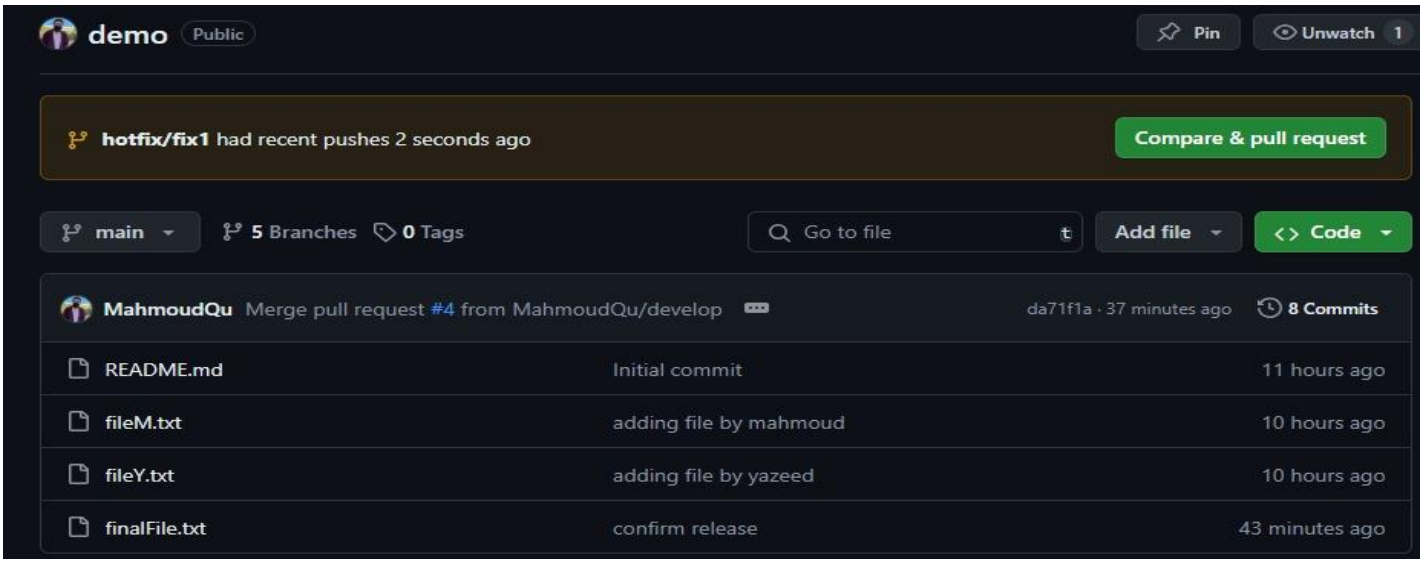


Hotfix, when to use? And why?

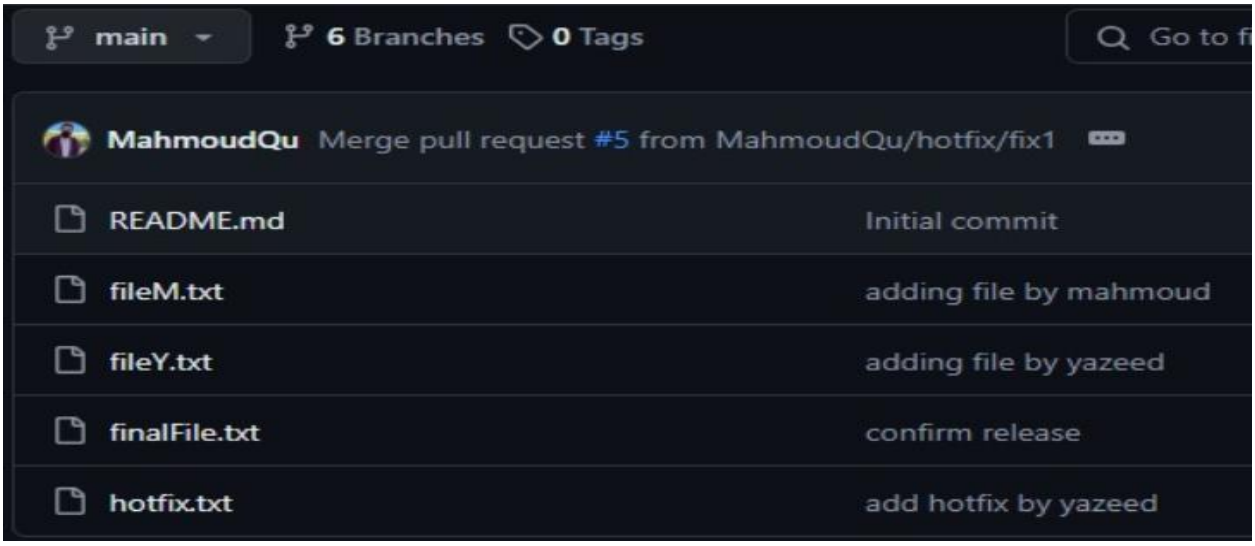
Hotfixes are crucial for addressing critical issues that need immediate attention in the production environment. A hotfix is typically added when a bug or issue is discovered in the main branch that impacts users and requires a swift resolution without waiting for the next release cycle.

Let's assume that the user "Yazeed" want to create one hotfix for this project, he want to add a file with the name "hotfix.txt", so he switched to the main branch, and created a hotfix/fix1 branch, the file is created and committed, and then he pushed the hotfix branch into the server to make pull request, after confirming the request, we got the fixing in the branch main,

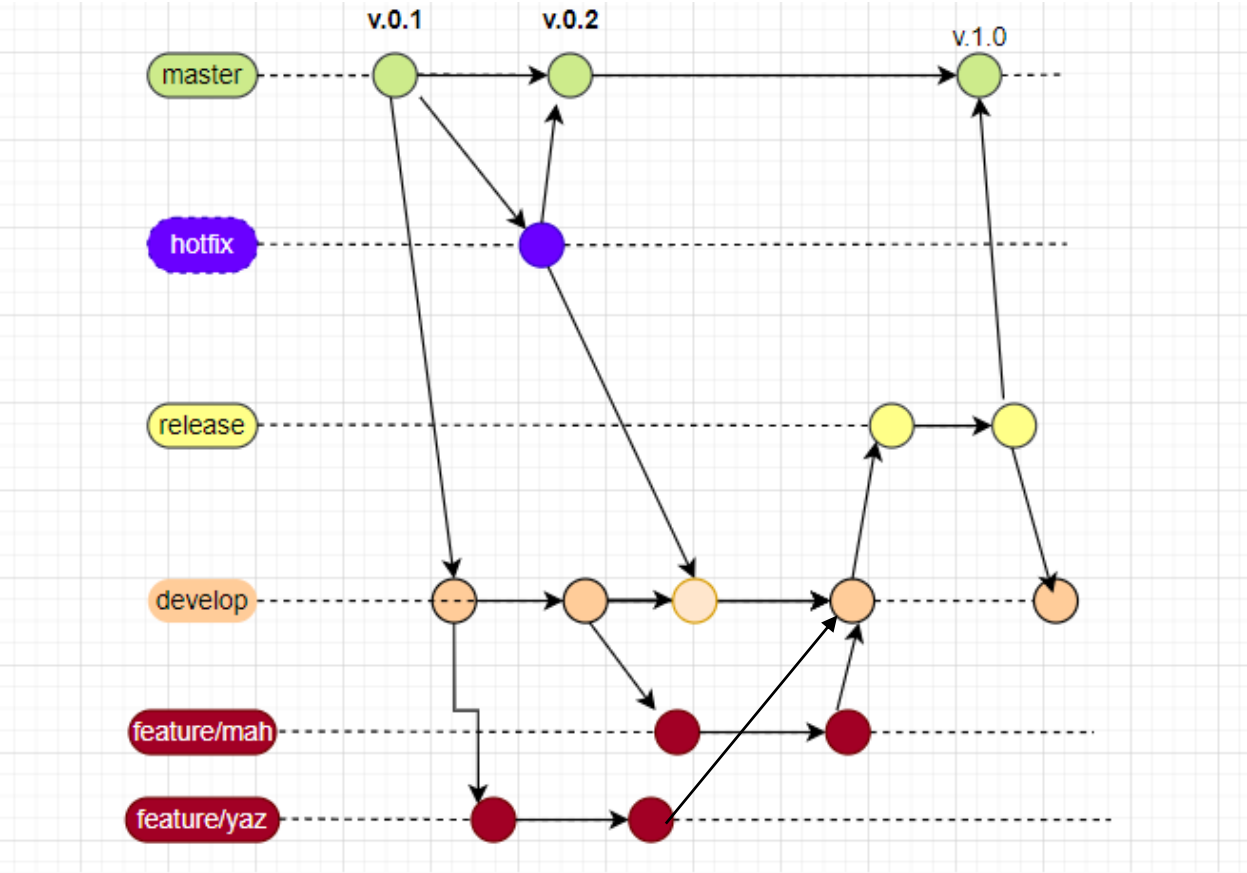
```
mahmo@DESKTOP-U19PM2V MINGW64 ~/desktop/git_users/Yazeed/demo (main)
$ git checkout -b hotfix/fix1
Switched to a new branch 'hotfix/fix1'

mahmo@DESKTOP-U19PM2V MINGW64 ~/desktop/git_users/Yazeed/demo (hotfix/
fix1)
$ vi hotfix.txt

mahmo@DESKTOP-U19PM2V MINGW64 ~/desktop/git_users/Yazeed/demo (hotfix/
fix1)
$ git add hotfix.txt
warning: in the working copy of 'hotfix.txt', LF will be replaced by C
RLF the next time Git touches it

mahmo@DESKTOP-U19PM2V MINGW64 ~/desktop/git_users/Yazeed/demo (hotfix/
fix1)
$ git commit -m "add hotfix by yazeed"
[hotfix/fix1 6704e55] add hotfix by yazeed
 1 file changed, 1 insertion(+)
 create mode 100644 hotfix.txt

mahmo@DESKTOP-U19PM2V MINGW64 ~/desktop/git_users/Yazeed/demo (hotfix/
fix1)
$ git push origin hotfix/fix1
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 309 bytes | 309.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
remote:
```


demo
Public
Pin
Unwatch 1
hotfix/fix1 had recent pushes 2 seconds ago
Compare & pull request
main
5 Branches
0 Tags
Go to file
t
Add file
Code
MahmoudQu Merge pull request #4 from MahmoudQu/develop
da71f1a · 37 minutes ago
8 Commits
README.md
Initial commit
11 hours ago
fileM.txt
adding file by mahmoud
10 hours ago
fileY.txt
adding file by yazeed
10 hours ago
finalFile.txt
confirm release
43 minutes ago


add hotfix by yazeed #5
Merged
MahmoudQu merged 1 commit into main from hotfix/fix1
now
Conversation 0
Commits 1
Checks 0
Files changed 1
MahmoudQu commented now
No description provided.
add hotfix by yazeed
MahmoudQu merged commit 0fc0884 into main now


main
6 Branches
0 Tags
Go to f
MahmoudQu Merge pull request #5 from MahmoudQu/hotfix/fix1
README.md
Initial commit
fileM.txt
adding file by mahmoud
fileY.txt
adding file by yazeed
finalFile.txt
confirm release
hotfix.txt
add hotfix by yazeed

Gitflow workflow chart:



So, we are done, and I would like to say that, through this assignment, I had the opportunity to learn about the real work environment, gaining practical experience with Git and the Gitflow workflow. Although it was a bit challenging, it was incredibly interesting and rewarding. The process helped me understand how collaborative development works and enhanced my skills in managing branches and integrating changes effectively.

More details can be found on the GitHub repository that I will share.
github repo link : *https://github.com/MahmoudQu/demo*
video link : *https://youtu.be/e_zkXaScXWc*

Thank You.