

# GitFlow Workflow Assignment










By Hamza Alhalabi

## Contents

- [GitFlow Workflow Assignment](#)
  - [Contents](#)
  - [1. Project Overview](#)
  - [2. Implementation of GitFlow in a Simulated Team Environment](#)
    - [2.1. Overview of GitFlow](#)
    - [2.2. Team Setup](#)
    - [2.3. Workflow Steps](#)
    - [2.4. Challenges and Solutions](#)
  - [3. Learning Experience and Insights](#)
    - [3.1. Key Learnings](#)
    - [3.2. Applying GitFlow in Real-World Scenarios](#)
  - [4. Conclusion](#)

## 1. Project Overview

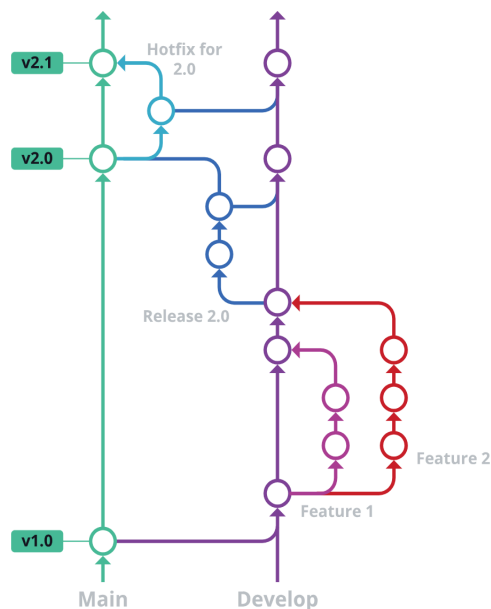
- **Objective:** I built a simple repository using Gitflow workflow principles to simulate team collaboration in real-world projects and learn how to work with a team with members who work in different positions.
- **Description:** I created a remote repository on GitHub called "Gitflow-simulation" and cloned the repository locally on my device twice, one in a repository called "Michael", and another one on a repository called "Zaid". I used the two repositories to simulate a real team workflow like in companies, I let both users edit and work on the same files in parallel.

<b>Zaid modified the file</b>  Hamza-Alhalabi-03 committed yesterday	b9470ab		
<b>Michael created the first file</b>  Hamza-Alhalabi-03 committed yesterday	9181d60		
<b>Initial commit</b>  Hamza-Alhalabi-03 authored yesterday	<b>Verified</b> 53fbe57		

## 2. Implementation of GitFlow in a Simulated Team Environment

### 2.1. Overview of GitFlow

Gitflow is a workflow model that enables team members to develop and deploy their project in a safer and more organized way. The idea of Gitflow is to divide the work into multiple branches, the main branch "main" is used only for production, everything pushed into it should be working without any problems or bugs. The development branch "develop" is the most used branch by developers, we can take a new branches from it to add or modify something, these branches are called "feature" branches, we merge it back to the develop branch after finishing. There are also "release" branches that used to prepare for production after development processes, and "hotfix" branches that are used for quick maintenance for production files on the "main".



### 2.2. Team Setup

- **Roles:** There were not a specific roles for my virtual developers (Michael and Zaid). I used both of them for development in feature branches, but Michael was used more frequently for merging and pull requests.

```
C:\Users\HamzaWH\Downloads\My-Github\Zaid\Gitflow-simulation (feature/add-utility-file -> origin)
λ git add .

C:\Users\HamzaWH\Downloads\My-Github\Zaid\Gitflow-simulation (feature/add-utility-file -> origin)
λ git commit -m "Zaid added a method to Utility class"
[feature/add-utility-file e3e3705] Zaid added a method to Utility class
1 file changed, 6 insertions(+), 1 deletion(-)
```

Michael Zaid

```
C:\Users\HamzaWH\Downloads\My-Github\Michael\Gitflow-simulation (feature/add-utility-file -> origin)
λ git add .
as a version control, and

C:\Users\HamzaWH\Downloads\My-Github\Michael\Gitflow-simulation (feature/add-utility-file -> origin)
λ git commit -m "Michael solved the conflict of two methods in the the utility class"
[feature/add-utility-file 2cddb29] Michael solved the conflict of two methods in the the utility class

C:\Users\HamzaWH\Downloads\My-Github\Michael\Gitflow-simulation (feature/add-utility-file -> origin)
```

- **Tools Used:** I used **Git** as a version control, and **GitHub** as a collaboration tool and the server between team members. I used **Visual Studio Code** for editing files and writing this report in Markdown.

## 2.3. Workflow Steps

I have demonstrated before what is Gitflow workflow and what are its main characteristics, here is a more deep explanation of this workflow methodology:

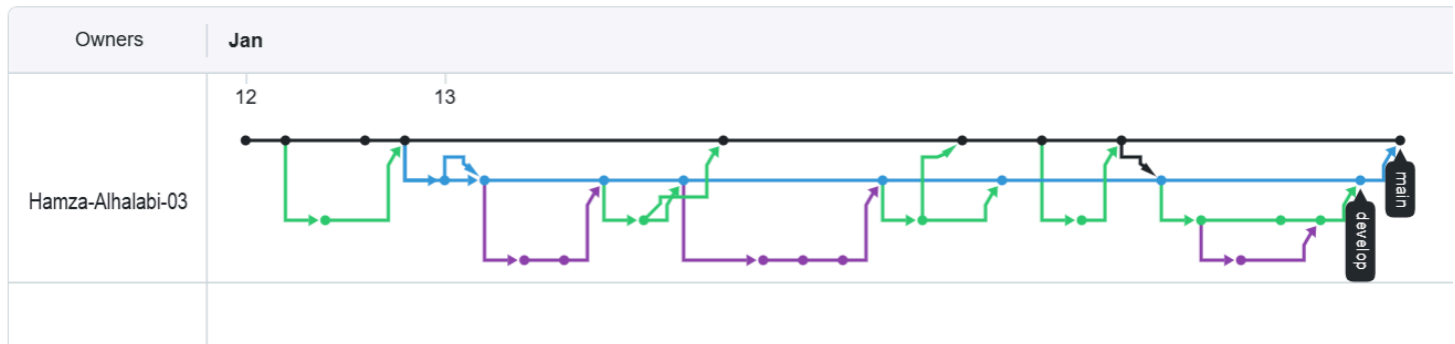
- **Main Branches:**
  - **main** : Known as **master** also, this branch is permanent and existed all the time during team work (it can't be deleted). It is the main production branch in the project, we can't push any changes or add any features into it directly without following the traditional process ( **feature** branches then **develop** branch then push to **main** ); the reason behind that is that we want to keep the project's version on the **main** ready all the time, it should be ready for deployment and for clients to use. So, everything should be completed like development and testing.
  - **develop** : The majority of work is done on this branch. Like the **main** branch, this branch is existed always and can't be removed. Developers collaborate and work on this branch using multiple feature branches. Pull requests usually happen here by a team leader. This branch keeps receiving commits and changes until we are ready to release a new version or update the production files directly.
- **Feature Branches:** The atomic changes of the whole project is completed in these branches. Developers branch out of the **develop** branch to work on particular feature like adding, modifying, and removing. When developers are finished they branch the changes back into **develop** branch using a merge operation with **Git**, or a pull request from **GitHub** GUI. Deletion of these branches after branching them into the **develop** branch is considered a best practice.
- **Release Branches:** Temporary branches that are used as a stage of preparation before real production in the **main** branch. They may have more than one commit of changes before pushing into the **main**. It is important to branch into **develop** branch also when branching into production at the end.
- **Hotfix Branches:** These branches are used for urgent changes on the **main** only, for quick bug fixing or modifying simple tasks, otherwise we should follow the standard way with **develop**

branch and its feature branches. It should be branch into the develop after completing the fix to keep developers up to date with project changes.

Branch	Updated	Check status	Behind	Ahead	Pull request
main	25 minutes ago		Default		
develop	31 minutes ago		1	0	#12

## Network graph

Timeline of the most recent commits to this repository and its network ordered by most recently pushed to.



## 2.4. Challenges and Solutions

In this section, I will discuss the most critical challenges that faced me while implementing Gitflow and how they were resolved:

- **Challenge 1:** I faced some conflicts in when the two virtual users modified the same file in the same time. This happens when more than one change occur on the same file in a period where the two users were not updated by the changes of each other.

```
Cmder
> git status
On branch feature/add-utility-file
Your branch is up to date with 'origin/feature/add-utility-file'.

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:   Utility.java

no changes added to commit (use "git add" and/or "git commit -a")

C:\Users\Hamza\Downloads\My-Github\Michael\Gitflow-simulation (feature/add-utility-file -> origin)
^ git add .

C:\Users\Hamza\Downloads\My-Github\Michael\Gitflow-simulation (feature/add-utility-file -> origin)
^ git commit -m "Michael added a method to the utility class"
[feature/add-utility-file bef7568] Michael added a method to the utility class
1 file changed, 9 insertions(+)

C:\Users\Hamza\Downloads\My-Github\Michael\Gitflow-simulation (feature/add-utility-file -> origin)
^ git pull
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (1/1), done.
remote: Total 3 (delta 2), reused 3 (delta 2), pack-reused 0 (from 0)
Unpacking objects: 100% (3/3), 348 bytes | 11.00 KiB/s, done.
From github.com:Hamza-Alhalabi-03/Gitflow-simulation
8f9ef0c..e3e3705 feature/add-utility-file -> origin/feature/add-utility-file
Auto-merging Utility.java
CONFLICT (content): Merge conflict in Utility.java
Automatic merge failed; fix conflicts and then commit the result.

C:\Users\Hamza\Downloads\My-Github\Michael\Gitflow-simulation (feature/add-utility-file -> origin)
```

- **Solution:** The solution of this problem is simple, and as the Doctor solved it in the lecture, it was not difficult. It needs to modify the file manually and merge the changes.

```
C: > Users > HamzaWH > Downloads > My-Github > Michael > Gitflow-simulation > J Utility.java
1 public class Utility {
2
3     Accept Current Change | Accept Incoming Change | Accept Both Changes | Compare Changes
4     <<<<<< HEAD (Current Change)
5
6     public static String reverseString(String str) {
7         if (str == null) {
8             return null;
9         }
10        return new StringBuilder(str).reverse().toString();
11    }
12
13    =====
14    public static int square(int number) {
15        return number * number;
16    }
17    >>>>>> e3e37055b7b421eca521254fb52def66a8b76b49 (Incoming Change)
18
19    public static void main(String[] args) {
20
21        System.out.println("Is 10 even? " + isEven(10));
22
23        Accept Current Change | Accept Incoming Change | Accept Both Changes | Compare Changes
24        <<<<<< HEAD (Current Change)
25
26        System.out.println("Reverse of 'hello': " + reverseString("hello"));
27
28    =====
29        System.out.println("Square of 5: " + square(5));
30    }
31    >>>>>> e3e37055b7b421eca521254fb52def66a8b76b49 (Incoming Change)
32
33    }
```

```
C: > Users > HamzaWH > Downloads > My-Github > Michael > Gitflow-simulation > J Utility.java
1 public class Utility {
2
3     public static boolean isEven(int number) {
4         return number % 2 == 0;
5     }
6
7     public static String reverseString(String str) {
8         if (str == null) {
9             return null;
10        }
11        return new StringBuilder(str).reverse().toString();
12    }
13
14    public static int square(int number) {
15        return number * number;
16    }
17
18    public static void main(String[] args) {
19
20        System.out.println("Is 10 even? " + isEven(10));
21
22        System.out.println("Reverse of 'hello': " + reverseString("hello"));
23
24        System.out.println("Square of 5: " + square(5));
25    }
26 }
27 }
```

- **Challenge 2:** Merge VS Pull request: I was confused at first between using merge with command line, and using pull requests with GitHub GUI to branch a branch into another one.
  - **Solution:** After researching and trying both techniques, I found that pull requests are more professional and common between developers. In addition, pull requests give the ability to review changes before merging branches, especially with teams as I am simulating a team workflow.
- **Challenge 3:** At first I did not know if I should add feature branches that I created locally to the remote server. I was confused because I felt like it is the same as merging them locally and push the new commits to develop branch.

- **Solution:** I did my research and found that new branches should be pushed to the server like this: `git push --set-upstream origin feature/feature-name`, because other developers have to be able to contribute to this temporary branch also, not only `develop` branch. But as I was working alone with virtual users, I could not imagine the situation correctly.
- **Challenge 4:** I did changes on the `develop` branch without creating new `feature`, and all the project was about to be ruined.
  - **Solution:** I used the reset command: `git reset --hard Commit-Hash-Code` to return to the previous commit point and everything was fine.

## 3. Learning Experience and Insights

### 3.1. Key Learnings

- I understood how version control systems especially with such workflows help us in managing and organizing the process of development between team members. And how it saves time and effort and help us avoid many potential conflicts and problems.
- This assignment gave me the opportunity to dive into the world of project management and collaboration between teams in real projects.
- During my search I learnt about extensions that organize Gitflow workflow between developers by using specific Git commands after installing certain libraries.
- I learnt how people work in organizations, this gave me the confidence and the belief that I am kind of ready to work professionally in a real company, and broke my fear from work.
- I get familiar with command line environment, as this was my first time using it seriously. I find it easier and faster than the graphical user interface.

### 3.2. Applying GitFlow in Real-World Scenarios

- **Benefits Observed:** I believe that applying Gitflow workflow in real scenarios have many advantages. It helps in organizing and managing the development lifecycle easily, and prevent many problems and conflicts that may appear when working with traditional ways like centralized workflow or normal feature branching workflow. I am not sure if it is the best way to control projects' versions, but I think that it is powerful enough to be standardized.
- **Potential Challenges:** On the other hand, there are some challenges that might face this workflow. For example, it might add unnecessary complexity for small teams, and increase

merging conflicts in larger projects. It might be difficult for new programmers to join teams with complex workflow and large number of branches. I also read about performance problems with CI/CD in huge projects as continuous merging branches may be an overload.

Another challenge that faces developers is the confusion that occurs when programmers can't determine where to make their changes. For instance, they might not be able to choose between normal feature branching or hotfix quick branching, especially in complicated situations.

- **Suitability for Real-World Use:** Based on challenges that face Gitflow applications, applying this workflow requires solid management on the process. The team should be careful when deciding to use it with large projects. Other things may be helpful for implementing Gitflow are regular merges to reduce conflicts and decrease efforts, and clear convention naming for branches.

## 4. Conclusion

Gitflow is an amazing tool for teams and developers, I believe that it created a major change in the history of development team management and version control.

In general, it was a great experience, this simulation helped me to understand the concepts of this tool easily. I was surprised by the simplicity and harmony between team members while working on the project. As someone tried the old way of development by sending all files to other developers and receiving them back after updating, this was a revolutionary experience.