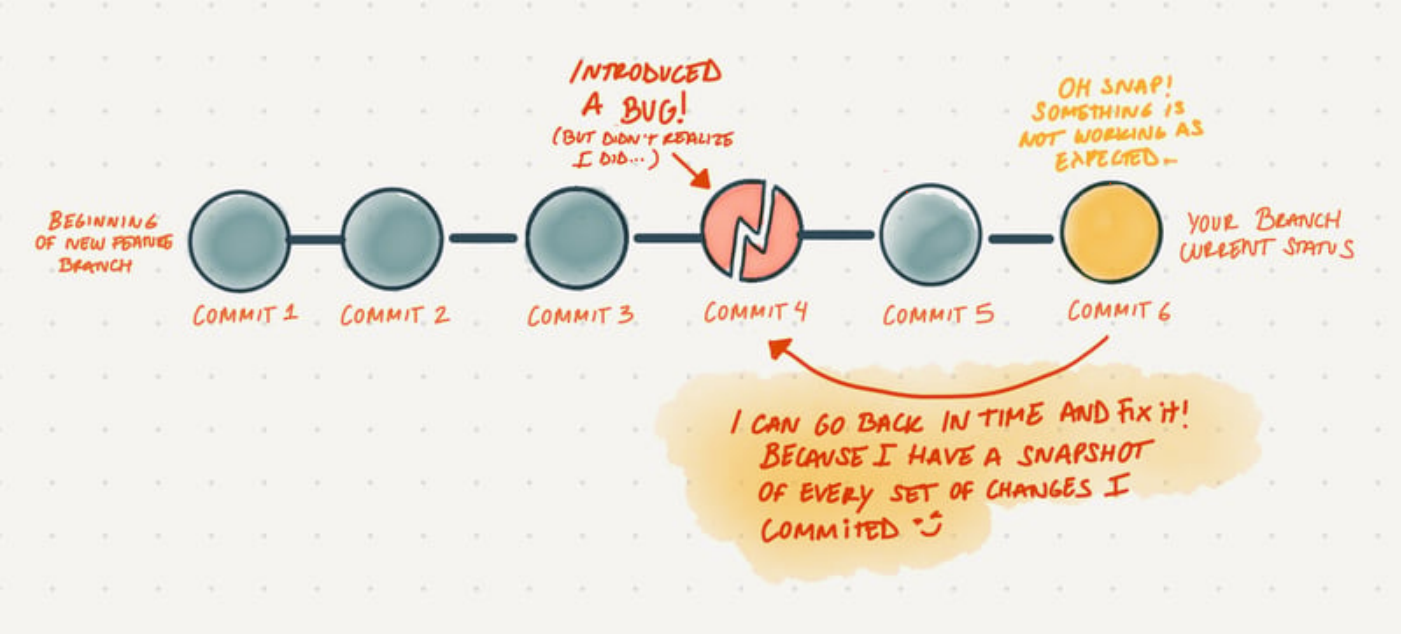
# Basic Programming

# Module 10 : Git

# What is Git?

Git is a Version Control System Software. A version control keeps track of every change in your code and it allows you to go back in time when something goes wrong.




# Why Use Git?

- Prevents overwriting changes when multiple people work on the same file.
- Keeps a history of changes, making it easy to debug issues.
- Facilitates collaboration with branches and merging.

# Git File Status

## 1. Untracked

- Files that are not yet being tracked by Git.
- **Example:** A new file created in the working directory.
- **Command to Track:**

```
git add <file_name>
```

or

```
git add .
```

to add all files in the directory.

## 2. Tracked

- Files that are being tracked by Git, can have three states:
- **Unmodified:** Files that have not been changed since the last commit.
- **Modified:** Files that have been changed since the last commit, this file is needed to be staged with `git add` command again.
- **Staged:** Files that have been added to the staging area and are ready to be committed.

## 3. Commited

- Files that have been saved to the Git database.
- Command to commit:

```
git commit -m "Commit message"
```

## 4. Ignored

- Files that are set to be ignored/not tracked by Git.
- Can be set in a `.gitignore` file, for example if there is a file named `secret.txt` that you don't want to track, you can add it to the `.gitignore` file.

```
secret.txt
```

This way `secret.txt` will not be staged or committed.

# Common Git Commands Cycle

| Command | Description |
| --- | --- |
| `git add <file_name>` | Add a file to the staging area |
| `git add .` | Add all files to the staging area |
| `git commit -m "Commit message"` | Commit the staged files |
| `git push origin <branch_name>` | Push the committed changes to a remote repository |

# Git Branch

A branch is a separate line of development, just like a `multiverse`. It allows you to work on a feature without affecting the main codebase. You can create a new branch with the following command:

```
git branch <branch_name>
```

to show all branches and the current branch:

```
git branch
```

to switch to a branch:

```
git checkout <branch_name>
```

# Git Merge

Merging is the process of combining changes from different branches. To merge a branch into your current branch you can do:

```
git merge <branch_name>
```

# Git Conflict

A conflict is a situation where two branches have made changes to the same line in a file. If that happen, git didn't know which change to keep, so it will ask you to resolve the conflict manually. You can resolve the conflict by editing the file and removing the conflict markers `<<<<<<<`, `=======`, `>>>>>>>`.

example of a conflict:

```
<<<<<<< HEAD
This is the change in the current branch
=======
This is the change in the branch you are merging
>>>>>>> branch_name
```

# GitHub

GitHub is a web-based Git repository hosting service. It works just like Google Drive, but for code. It allows you to store your code in the cloud and collaborate with others. You can create a repository, add collaborators, and work on the same codebase. GitHub also has a feature called `pull requests` which allows you to request changes to be merged into the main codebase.

# GitHub Repository

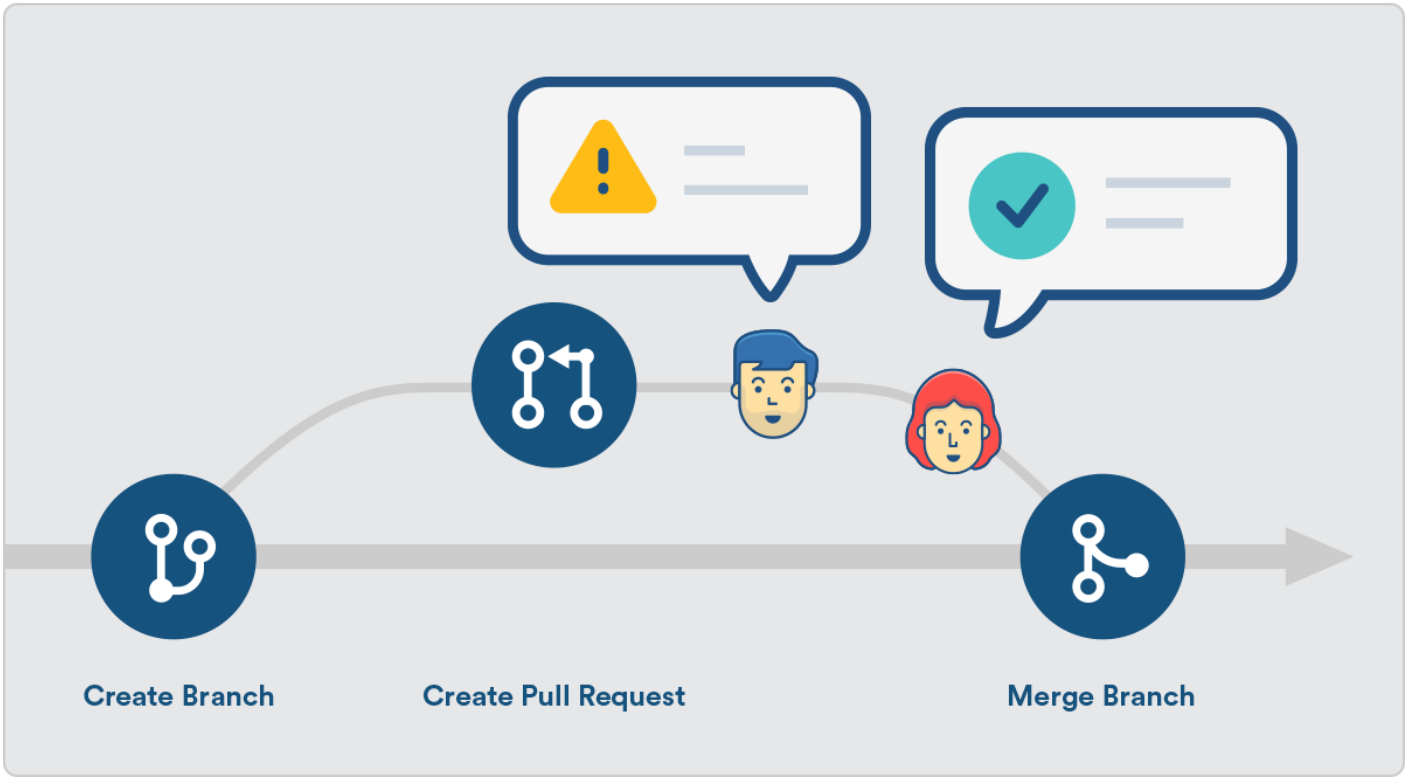
A repository is a place where your code is stored, like a folder on your computer but in the cloud. A GitHub repository can be public or private. A Public repository is visible to everyone, while a Private repository is only visible to you and the collaborators you add.

On GitHub repository you can see the past changes made to the codebase, who made them, and when they were made. You can also see the current state of the codebase, the branches, and the pull requests.

# GitHub Pull Request

A pull request is a proposal to merge changes from one branch into another, just like a `git merge`. It allows you to request changes to be merged into the main codebase. You can create a pull request on GitHub by clicking on the `New pull request` button on the repository page.

The difference between a `git merge` and a `pull request` is that a `git merge` is done locally on your machine, while a `pull request` is done on GitHub and needs a review or someone to approve the changes.




# GitHub Fork

A fork is a copy of a repository, It Allows you to freely experiment with changes without affecting the original project. This also allows you to contribute to a repository that you don't have direct access to. Once you've made changes to your fork, you can create a pull request to propose your changes to the original repository.

## Common GitHub Commands

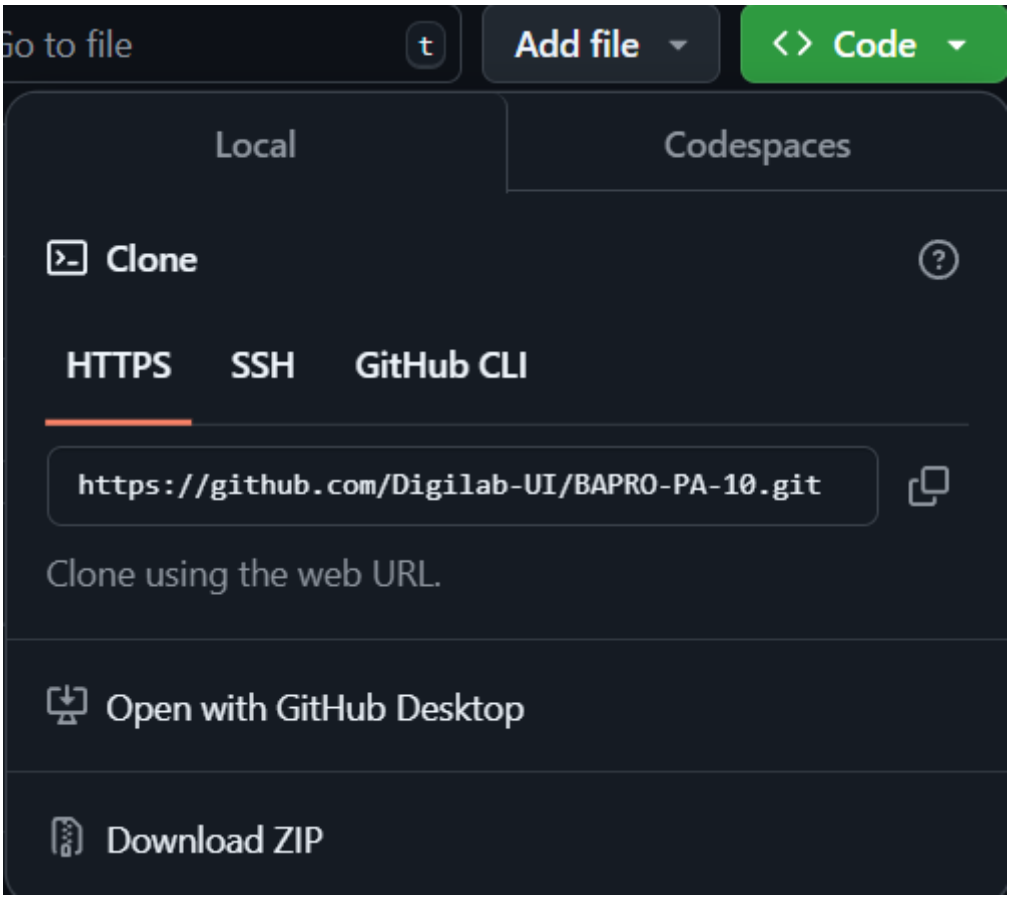
| Command | Description |
| --- | --- |
| `git clone <repository_url>` | Clone a repository to your local machine |
| `git pull` | Get the latest changes from the remote repository |
| `git push origin <branch_name>` | Push the committed changes to a remote repository |

# Setting up GitHub with local Git

Configuring your local Git with GitHub is a one-time process

```
git config --global user.name "Your name"
git config --global user.email "Your email"
```

There are two ways you can connect your local Git with GitHub:




## Creating the local folder first

1. Create a new folder on your local machine
2. Initialize a new Git repository
3. Add your files to the staging area
4. Commit your changes
5. Create a new repository on GitHub **(without a README file)**
6. Add the remote repository URL to your local repository

   ```
   git remote add origin <repository_url>
   ```

7. Push your changes to the remote repository

## Creating a repository on GitHub first

1. Create a new repository on GitHub
2. Clone the repository to your local machine

   ```
   git clone <repository_url>
   ```