How to... Git and GitHub

Introduction

Git is a program designed to track and manage your code changes over time. We will discuss the difference between Git and GitHub in a moment, but first, let's look at Git itself. You typically use Git in your terminal, though some IDEs (Integrated Development Environments) have Git extensions that provide a graphical user interface (UI).

Git for Versioning

Imagine writing code. Everything works perfectly. After a while, you want to add a feature, and suddenly, the code no longer works. Normally, you would have to manually delete all changes until the code works again. With Git, however, you can save any stage of your code at any time. This allows you to jump back in time (and changes) as needed.

The Git Workflow

The basic workflow is quite simple:

1. Create a Git repository.
2. Write code (make changes).
3. Add the changed files to your "Staging Area".
4. Commit the changes with a descriptive message.

These are the essential steps for creating cleaner code projects.

1. Create a Git Repository

To begin, we need to create a Git repository (a special folder) where Git starts tracking our code.

```
git init
```

2. Write Code

First, we'll create an empty code file to have a change to track.

```
touch my_code.py
```

3. Stage the File

By staging our changes, we prepare all files to be versioned by Git. They are moved into the "Staging Area".

```
git add my_code.py
```

4. Commit the Changes

Now we can commit our changes by saving them with a short note. The `git commit` command will save all changes we staged in the previous step.

```
git commit -m "Initial commit: Created an empty file"
```

Done! Now we can start coding and come back to this point anytime we make a mistake.

What About GitHub?

While Git runs only locally on your computer, sometimes we want to collaborate on the same project with colleagues. Instead of only creating a local Git repository with `git init`, we can use GitHub (or similar services like GitLab/Bitbucket) to provide a repository in the Cloud for all users.

The workflow remains largely the same, except for two additional commands:

1. **Pull the code from GitHub**

2. Write code
3. Stage the file
4. Commit the changes
5. **Push to GitHub**

1. Get the Latest Code

Before you start coding, make sure you have the newest version of the code from the Cloud.

```
git pull
```

Now you can follow the normal Git workflow as you learned above.

5. Update the Code on GitHub

Your committed changes are only saved locally. You need to push your commit to GitHub to update the code for all other users.

```
git push
```

> Note: This step only works when you have correctly followed the Git workflow by staging (`git add`) and committing (`git commit`) your changes first.