# Introduction to Git & GitHub

Yusuf Ferhat Yılman

## Contents

# 1 Introduction

Git is a distributed version control system that allows developers to track changes to their codebase, create branches for new features or fixes, and merge changes. GitHub is a web-based platform built around Git, providing features for hosting repositories, managing projects, and collaboration among developers.

# 2 Getting Started with Git

## 2.1 Installation

To install Git on your system, follow these steps:

1. Visit the official Git website (https://git-scm.com/) and download the appropriate installer for your operating system.
2. Run the installer and follow the instructions to complete the installation.
3. Once installed, open a terminal or command prompt and type `git -version` to verify that Git has been installed successfully.
4. You can also download a GUI client for Git. It might be more suitable for beginners. You can choose from various GUI clients here (https://git-scm.com/downloads/guis).

## 2.2 Configuration

After installing Git, you need to configure your user name and email address. This information is used to identify your commits. To configure Git, use the following commands:

```
git config --global user.name "Your Name"
git config --global user.email "your_email@example.com"
```

You can also set other configurations such as default text editor, merge tool, etc. using similar commands.

One alternative is to use a credential manager so that you don't have to keep typing your username or an access token every time Git prompts you to. An automated way to do this is to set up git-credential-manager. Follow the install instructions based on your operating system. After the installation you need to set the manager: `git config -global credential.helper manager` assigns GCM as the credential helper. After this step you need to choose a method in which to store your credentials: here you can see various options for credential storing. A depreciated or rather insecure way of doing this is to set a plaintext method to store credentials (Warning: this is not a secure way): `git config -global credential.credentialStore plaintext`.

## 2.3 Basic Commands

Git provides a set of basic commands to start working with repositories. Here are some essential commands:

- `git init`: Initialize a new Git repository in the current directory.
- `git add <file>`: Add file changes. Use: `git add .` to add all files to the working repository.
- `git commit -m "Message"`: Commit staged changes to the repository with a descriptive message.

- `git status`: Check the status of files in the repository.
- `git log`: View commit history.

# 3 Working with Repositories

## 3.1 Creating a Repository

To create a new Git repository, follow these steps:

1. Navigate to the directory where you want to create the repository using the terminal or command prompt.
2. Use the command `git init` to initialize a new Git repository in that directory.
3. Your repository is now created, and you can start adding files, making commits, and managing your project's history.

## 3.2 Cloning a Repository

To clone an existing Git repository from a remote location (such as GitHub), use the `git clone` command followed by the repository's URL. Here's how you can do it:

```
git clone <repository_url>
```

This command creates a local copy of the remote repository on your machine, allowing you to work on the project locally and push changes back to the remote repository when needed.

## 3.3 Forking Repositories

Forking a repository on GitHub allows you to create a copy of someone else's project under your GitHub account. This copy is entirely separate from the original repository, allowing you to make changes without affecting the original project. Here's how you can fork a repository:

1. Navigate to the repository you want to fork on GitHub.
2. On the repository's page, click on the "Fork" button located in the top-right corner of the page.
3. Once the forking process is complete, you'll have a copy of the repository under your GitHub account.

After forking a repository, you can make changes to the code, add new features, or fix issues as needed. If you believe your changes would benefit the original project, you can create a pull request to propose your changes be incorporated back into the original repository.

# 4 Branching and Merging

## 4.1 Creating a Branch

Branches in Git allow you to work on different features or fixes without affecting the main codebase. Here's how you can create a new branch:

1. Make sure you're on the branch you want to base your new branch off of. Typically, this is the main branch (e.g., `master` or `main`).
2. Use the command `git branch <branch_name>` to create a new branch.

3. To switch to the newly created branch, use the command `git checkout <branch_name>`.
4. Now you can start making changes on this branch without affecting the main codebase.

### 4.2 Merging Branches

Once you've completed the changes on your branch and they're ready to be integrated into the main codebase, you can merge your branch back into the main branch using the following steps:

1. Switch to the branch you want to merge into (e.g., `main`).
2. Use the command `git merge <branch_name>` to merge the changes from the specified branch into the current branch.
3. Git will attempt to merge the changes automatically. If there are conflicts, you'll need to resolve them manually.
4. After resolving any conflicts and completing the merge, commit the changes to finalize the merge operation.

## 5 Collaborating with GitHub

### 5.1 Pushing Changes

After making changes to your local repository, you may want to share those changes with others on GitHub. Here's how you can push your changes to a remote repository:

1. Ensure that you're on the branch where you want to push your changes.
2. Use the command `git push <remote_name> <branch_name>` to push your changes to the remote repository. By default, the remote name is usually `origin`.
3. If you're pushing changes for the first time, you may need to set the upstream branch using `git push -u <remote_name> <branch_name>`.
4. Your changes are now visible to others who have access to the remote repository on GitHub.

### 5.2 Pulling Changes

To retrieve changes made by others from a remote repository and incorporate them into your local repository, you can use the following command:

```
git pull <remote_name> <branch_name>
```

This command fetches the changes from the specified branch on the remote repository and merges them into your current branch. It's essential to pull changes regularly to stay up-to-date with the latest developments in the project.