# Git and GitHub

**Instructor: Pramod Kumar Jena**

## What is Git?

- **Git** is a free, open-source version control system designed to handle everything from small to large projects. It allows developers to keep track of changes in their code, collaborate with others, and revert back to previous versions when necessary.
- **Version Control** helps developers manage different versions of files without creating multiple copies. Git makes it easy to branch off, experiment, and then merge back changes into the main project.

**Key Features of Git:**

- **Version Control**: Tracks changes in files.
- **Branching**: Allows developers to work on different features without disturbing the main codebase.
- **Merging**: Combine changes from different branches.
- **Revert**: Undo mistakes by reverting to an earlier version.

## What is GitHub?

- **GitHub** is a cloud-based hosting service for Git repositories. It allows developers to store, share, and collaborate on Git projects online.
- **Repository**: A central location where all files, versions, and branches are stored.
- GitHub enhances collaboration by allowing multiple people to work on the same project and merge their code using Git.

**Why Git and GitHub Are Important for Developers?**

1. **Collaboration**: Multiple developers can work on the same project simultaneously.
2. **Backup and Security**: GitHub serves as a backup for code and provides security through version history.
3. **Open Source Contributions**: GitHub is popular in the open-source community, allowing developers to contribute to various projects.
4. **Deployment**: GitHub integrates with various CI/CD tools for continuous integration and deployment.

---

## Installing Git

1. **Windows:**
    - Go to Git for Windows and download the installer.
    - Run the installer and follow the instructions.
    - During installation, select "Use Git from Git Bash only" or "Git from the command line and 3rd-party software".
    - Once installed, open **Git Bash** from the Start menu.
2. **MacOS:**

Open the terminal and type:

```
brew install git
```

    - Alternatively, download Git from Git's official site.
3. **Linux:**

Open your terminal and type:

```
sudo apt-get update
sudo apt-get install git
```

    -

## Setting Up Git for the First Time

1. **Configure your username and email:**

Open your terminal or Git Bash and type:

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

    -
2. **Check the configuration:**

To check if everything is set correctly, type:

```
git config --list
```

    -

---

## How to Clone a Repository Using Git

1. **Find the Repository URL:**

    - Go to the repository on GitHub you want to clone.
    - Click on the "Code" button and copy the repository URL (usually looks like `https://github.com/username/repository.git`).
2. **Clone the Repository:**
    - Open **Git Bash** or your terminal.

Navigate to the directory where you want to clone the repository:

```
cd /path/to/your/directory
```

Use the `git clone` command followed by the repository URL:

```
git clone https://github.com/username/repository.git
```

3. **Check the Cloned Repository:**

Once cloned, navigate into the cloned repository folder:

```
cd repository
```

    - You now have a local copy of the project and can start working on it!

---

## Common Git Commands for Beginners

**git status**: Check the status of your changes.

```
git status
```

**git add**: Stage files to be committed.

```
git add filename
git add . # Add all files
```

**git commit**: Commit your changes with a message.

```
git commit -m "Add feature X"
```

**git push**: Push your local commits to GitHub.

```
git push origin main
```

**git pull**: Pull updates from the repository to your local machine.

```
git pull origin main
```

---

## Real-Life Example for Beginners

Imagine you're working on a website project with a team. Each team member is responsible for different parts: one works on the header, another on the footer, and someone else on the homepage content. Git allows each person to work independently on their part using **branches**. Once everyone finishes, Git helps **merge** all the changes into the main project without losing work or creating conflicts.

- If you make a mistake, Git allows you to **revert** to a previous version.
- If you want to try a new feature, you can create a **branch**, test the feature, and then **merge** it back into the main project if successful.