
## + What is Git?

Git is a version control system used to track changes in your code. It is free and open-source software available for Windows, macOS, and Linux. Git allows developers to manage their code efficiently, collaborate with teams, and maintain different versions of a project.

**Remember:** Git is a **software** that must be installed on your computer.

**Check if Git is Installed**
To check if Git is installed on your system, run: git --version

---

## +What is GitHub?

GitHub is a web-based platform for hosting Git repositories. It allows developers to store, share, and collaborate on code with others. While GitHub is not the only platform for hosting Git repositories, it is one of the most popular ones.

## Key Git Commands and Concepts

### 1. Repository

A **repository (repo)** is a collection of files and directories stored together. It is like a folder that holds your project code, history, and configurations.

To check the current state of your repository, run: git status

### 2. Configuring Git

Before you start using Git, set up your username and email (recommended to use your GitHub email):

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

To verify your configuration, run: git config --list

### 3. Creating a New Repository

If you are starting a new project:

```
mkdir my-project  # Create a new directory
cd my-project  # Navigate into the directory
git init  # Initialize Git in the folder
```

This creates a **hidden .git folder**, marking it as a Git repository.

If you want to clone an existing GitHub repository: `git clone <repository-link>`

This downloads the repository to your local machine.

### 4. Staging and Committing Changes

After making changes, you need to **stage** and **commit** them.

**Adding files to the staging area:**

```
git add <file-name>  # Add a specific file
git add .  # Add all changes
```

**Committing changes:** `git commit -m "Your commit message here"`

A commit acts as a **snapshot** of your project at a given time.

### 5. Pushing Changes to GitHub

To upload local changes to a remote repository: `git push origin main  # Push code to the 'main' branch`

### 6. Pulling Changes from GitHub

To fetch and merge changes from GitHub: `git pull origin main`

This keeps your local repository updated with the latest changes.

---

## + Two Ways to Use GitHub

**Method 1: Start from GitHub and Clone to Local Machine (Local machine -means your laptop or computer)**

1. Create a repository on GitHub
2. Clone it to your computer: `git clone <repository-link>`
3. Check repository status: `git status`
4. Make changes, add, commit, and push:

```
git add .
git commit -m "Your commit message"
```

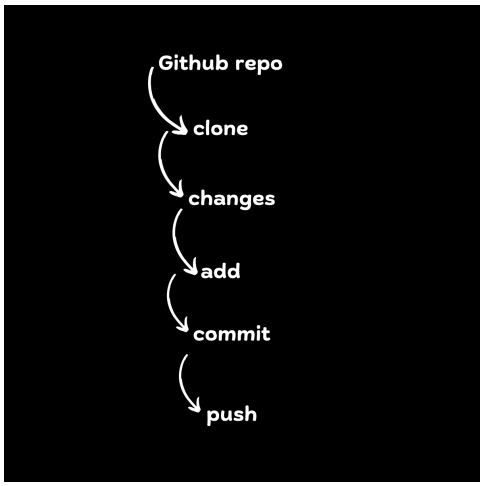
```
git push origin main
```

---

### Method 2: Start from Local Machine and Push to GitHub

1. **Create a new project folder:** `mkdir my-project  cd my-project`
2. **Initialize Git:** `git init`
3. **Create a repository on GitHub** and copy the repository link.
4. **Connect the local repository to GitHub:** `git remote add origin <repository-link>`
5. **Verify the remote connection:** `git remote -v`
6. **Check the current branch:** `git branch`
7. **Rename the branch to 'main':** `git branch -M main`
8. **Push the initial commit to GitHub:** **`git push -u origin main`**

**Work Flow:**




---

## Git Branches: Working with Multiple Versions of Your Code

**A branch allows you to work on new features without affecting the main codebase.**

### 1. Create a New Branch

`git branch new-feature`

### 2. Switch to a Branch

`git checkout new-feature`

**Or use:**

`git switch new-feature`

### 3. Merge a Branch into Main

**First, switch back to main:**

```
git checkout main
```

**Then merge:**

```
git merge new-feature
```

## 4. Delete a Branch

```
git branch -d new-feature
```