GIT AND GITHUB: A STEP-BY-STEP GUIDE

Install, configure, and push your code.

Introduction

Git is a distributed version control system that allows you to track changes to your code, collaborate with others, and revert to previous versions if needed. GitHub is a web-based platform for version control using Git. This guide will walk you through the process of installing Git, creating a repository, and pushing your code to GitHub.

Step 1: Install Git

Choose your operating system for installation instructions:

Windows

1. Download Git for Windows from the official website: https://git-scm.com/download/win
2. Run the installer and follow the on-screen instructions. Accept the default settings for most options.
3. Open Git Bash (installed with Git) to use Git commands.

macOS

1. **Using Homebrew:** If you have Homebrew installed, open Terminal and run:
2. **Using the Installer:** Download the Git installer for macOS from the official website: https://git-scm.com/download/mac
3. Follow the installation instructions.

Linux

Debian/Ubuntu:

Open Terminal and run: then

Fedora/CentOS:

Open Terminal and run:

Arch Linux:

Open Terminal and run:

Step 2: Configure Git

After installing Git, configure your username and email. Open your terminal (Git Bash on Windows) and run the following commands, replacing the example values with your own:

You can verify your configuration by running:

Step 3: Create a Local Repository

1. Navigate to your project directory in the terminal using the command (e.g.,).
2. Initialize a new Git repository using the following command:

This creates a subdirectory in your project directory, which contains the repository metadata.

Step 4: Create a GitHub Repository

1. Go to GitHub (https://github.com/) and sign in or create an account.
2. Click the "+" button in the top right corner and select "New repository".
3. Enter a name for your repository. Choose a descriptive name.
4. Add a description (optional).
5. Choose whether the repository should be public or private.
6. You can initialize with a README file, license, or .gitignore file. If you are starting with an existing project, it is better to leave these options unchecked.
7. Click "Create repository".

Step 5: Connect Local Repository to GitHub

You can connect your local repository to GitHub using either HTTPS or SSH. SSH is generally more secure but requires additional setup.

HTTPS

1. Copy the HTTPS URL of your GitHub repository (found on the repository's page after creation, it looks like:).
2. In your terminal, run the following command, replacing the URL with your copied URL:

SSH

1. Generate an SSH key pair if you don't already have one. Use the following command:

Accept the default file location (or choose a different one).
2. Add your SSH key to your GitHub account (copy the contents of to your GitHub SSH settings).
3. Copy the SSH URL of your GitHub repository (found on the repository's page after creation, it looks like:).
4. In your terminal, run the following command, replacing the URL with your copied URL:

Step 6: Push Code to GitHub

1. Add your project files to the staging area:
2. Commit your changes with a descriptive message:
3. Push your code to the GitHub repository:

(The flag sets the upstream branch, so you only need to use it the first time. Subsequent pushes can be done with .)

Step 7: Verify the Process

Go to your GitHub repository in your web browser. You should see your committed files in the repository.

Common Git Commands

Command	Description
`git init`	Initializes a new Git repository.
`git clone <repository_url>`	Clones a repository from a remote URL.
`git add <file>`	Adds a file to the staging area.
`git commit -m "message"`	Commits changes with a descriptive message.
`git push origin <branch>`	Pushes changes to a remote repository.
`git pull origin <branch>`	Pulls changes from a remote repository.
`git status`	Shows the status of the working directory.
`git log`	Shows the commit history.

Next Steps

Congratulations! You have successfully installed Git, created a repository, and pushed your code to GitHub. Explore more advanced Git concepts such as branching, merging, and conflict resolution to enhance your collaborative workflow.