GitHub Tutorial

Getting Started with Git and GitHub

A Complete Guide to Version Control

July 4, 2025

Contents

1 Introduction

This tutorial will guide you through the essential steps of using Git and GitHub for version control. You'll learn how to create repositories, manage files, and understand important concepts like `.gitignore` and `README.md`.

2 Getting Started with GitHub

2.1 Step 1: Create Repository

The first step in any GitHub project is to create a repository. A repository (or "repo") is a storage space where your project lives.

1. Go to GitHub.com and sign in to your account
2. Click the "+" icon in the top right corner
3. Select "New repository"
4. Give your repository a name
5. Choose whether it should be public or private
6. Initialize with a README (optional)
7. Click "Create repository"

2.2 Step 2: Push Files to GitHub

Once your repository is created, you'll need to push your local files to GitHub using Git commands.

3 Important Files

3.1 .gitignore File

3.1.1 What is it?

A special file that tells Git to ignore certain files or folders, so they won't be tracked or pushed to GitHub.

3.1.2 Why use it?

To avoid committing:

- Temporary files (like `.log`, `.tmp`)
- Secret files (`.env`)
- Big folders (`node_modules`, `build`)
- System files (`.DS_Store`, etc.)

3.2 README.md File

3.2.1 What is it?

A Markdown file that explains your project, and shows at the top of your GitHub repo.

3.2.2 Why use it?

To help others (and yourself) understand:

- What your project does
- How to install or use it
- Its features
- Any special instructions

4 Essential Git Commands

4.1 1. Git Configuration

Before you start using Git, you need to configure your identity:

```
git config --global user.name "Karim Assi"
git config --global user.email "karim@example.com"
```

Important Notes:

- Use `--global` if you want it to apply to all projects
- Without `--global`, it only applies to the current repo

4.2 Git Configuration Commands Reference

Command	What it does
`git config --global user.name "Name"`	Sets your name for all projects
`git config --global user.email "Email"`	Sets your email for commits
`git config --list`	Shows all config settings
`git config --local`	Sets config for just one repo
`git config core.editor`	Changes the text editor

Table 1: Git Configuration Commands

4.3 2. Git Clone

```
git clone [link of file]
```

What it does:

- Downloads a complete copy of an existing repository (e.g., from GitHub) to your local computer
- Includes all files, folders, and the full commit history

4.4 3. Git Pull

```
git pull origin main
```

What it does:

- Downloads the latest changes from the remote repository (usually GitHub) to your local project folder

When to use it:

- When someone else pushes updates to GitHub
- When you worked on another computer and pushed changes
- To sync your local repo with the latest version on GitHub

5 Common Git Workflow

Here's a typical workflow when working with Git and GitHub:

1. **Clone** the repository to your local machine
2. **Make changes** to your files
3. **Stage** your changes with `git add`
4. **Commit** your changes with `git commit`
5. **Push** your changes to GitHub with `git push`
6. **Pull** latest changes when needed with `git pull`

6 Best Practices

- Always write clear, descriptive commit messages
- Use `.gitignore` to exclude unnecessary files
- Keep your README.md updated with project information
- Pull before pushing to avoid conflicts
- Use branches for different features or experiments

7 Conclusion

Git and GitHub are powerful tools for version control and collaboration. By following this tutorial, you now have the foundation to start managing your projects effectively. Remember to practice these commands regularly to become more comfortable with the Git workflow.