Complete Git & GitHub Guide - From Zero to Hero

What is Git & GitHub?

Git = Version control system (tracks changes in your code) **GitHub** = Online platform to store and share Git repositories

Think of Git as a save system for your code that remembers every change, and GitHub as Google Drive for developers.

Installation & Setup

Install Git

- **Windows**: Download from git-scm.com
- **Mac**: `brew install git` or download from website
- **Linux**: `sudo apt install git` or `sudo yum install git`

First-time Setup

bash

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

Essential Git Commands

Starting a Repository

bash

```
git init                    # Create new repository
git clone <url>             # Copy existing repository
```

Basic Workflow

bash

```
git status                  # Check what's changed
git add <file>              # Stage specific file
git add .                   # Stage all changes
git commit -m "message"     # Save changes with description
git push                    # Upload to GitHub
git pull                    # Download latest changes
```

Checking History

bash

```
git log                     # See commit history
git log --oneline           # Compact history
git diff                    # See current changes
```

Understanding Git Workflow

```
Working Directory → Staging Area → Repository → GitHub
     (edit)          (git add)    (git commit)  (git push)
```

1. **Working Directory**: Where you edit files
2. **Staging Area**: Files ready to be committed
3. **Repository**: Saved snapshots (commits)
4. **GitHub**: Cloud storage

Branching Made Simple

Branches let you work on features without breaking main code.

bash

```
git branch                  # List branches
git branch <name>           # Create branch
git checkout <name>         # Switch to branch
git checkout -b <name>      # Create and switch
git merge <branch>          # Merge branch into current
git branch -d <name>        # Delete branch
```

Common Branch Names:

- `main` or `master` - Main production code
- `develop` - Development branch
- `feature/login` - New feature
- `bugfix/navbar` - Bug fixes

GitHub Basics

Creating Repository on GitHub

1. Go to GitHub.com → Sign up/Login
2. Click "New Repository"
3. Name it, choose public/private
4. Don't initialize if you have local code

Connecting Local to GitHub

bash

```
git remote add origin <github-url>
git branch -M main
git push -u origin main
```

Collaboration Features

- **Issues**: Track bugs/features
- **Pull Requests**: Propose changes
- **Fork**: Copy someone's repository
- **Star**: Bookmark repositories

Essential GitHub Workflow

Contributing to Projects

1. **Fork** the repository
2. **Clone** your fork locally
3. Create a **branch** for your feature
4. Make changes and **commit**
5. **Push** to your fork
6. Create **Pull Request**

Common Git Problems & Solutions

Undo Changes

bash

```
git checkout -- <file>        # Undo changes in working directory
git reset HEAD <file>          # Unstage file
git revert <commit-hash>       # Undo a commit safely
git reset --hard HEAD~1        # Delete last commit (dangerous!)
```

Fix Mistakes

bash

```
git commit --amend             # Edit last commit message
git stash                      # Temporarily save changes
git stash pop                  # Restore stashed changes
```

Merge Conflicts

When Git can't automatically merge:

1. Open conflicted files
2. Look for `<<<<<<<`, `=======`, `>>>>>>>` markers
3. Choose which code to keep
4. Remove conflict markers
5. `git add` and `git commit`

Best Practices

Commit Messages

- **Good**: "Add user login validation"
- **Bad**: "fixed stuff"

Format: `Type: Short description`

- `feat: Add new feature`
- `fix: Fix bug`
- `docs: Update documentation`

- `style: Code formatting`

Repository Structure

```
project/
├── README.md
├── .gitignore
├── src/
├── docs/
└── tests/
```

Hands-On Project: Personal Portfolio Website

Let's build a simple portfolio website to practice everything!

Step 1: Setup

bash

```bash
# Create project folder
mkdir my-portfolio
cd my-portfolio

# Initialize Git
git init

# Create files
touch index.html style.css script.js README.md
```

Step 2: Create Basic Files

index.html

html

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>My Portfolio</title>
    <link rel="stylesheet" href="style.css">
</head>
<body>
    <header>
        <h1>Your Name</h1>
        <nav>
            <a href="#about">About</a>
            <a href="#projects">Projects</a>
            <a href="#contact">Contact</a>
        </nav>
    </header>

    <main>
        <section id="about">
            <h2>About Me</h2>
            <p>I'm learning Git and GitHub!</p>
        </section>

        <section id="projects">
            <h2>My Projects</h2>
            <div class="project">
                <h3>Portfolio Website</h3>
                <p>Built with HTML, CSS, and JavaScript</p>
            </div>
        </section>

        <section id="contact">
            <h2>Contact</h2>
            <p>Email: your.email@example.com</p>
        </section>
    </main>

    <script src="script.js"></script>
</body>
</html>
```

style.css

css

```css
* {
    margin: 0;
    padding: 0;
    box-sizing: border-box;
}

body {
    font-family: Arial, sans-serif;
    line-height: 1.6;
    color: #333;
}

header {
    background: #2c3e50;
    color: white;
    padding: 1rem;
    text-align: center;
}

nav a {
    color: white;
    text-decoration: none;
    margin: 0 1rem;
}

main {
    max-width: 800px;
    margin: 2rem auto;
    padding: 0 1rem;
}

section {
    margin-bottom: 2rem;
}

.project {
    background: #f4f4f4;
    padding: 1rem;
    margin: 1rem 0;
    border-radius: 5px;
}
```

script.js

javascript

```javascript
// Simple scroll effect
document.addEventListener('DOMContentLoaded', function() {
    const links = document.querySelectorAll('nav a');

    links.forEach(link => {
        link.addEventListener('click', function(e) {
            e.preventDefault();
            const target = document.querySelector(this.getAttribute('href'));
            target.scrollIntoView({ behavior: 'smooth' });
        });
    });
});
```

README.md

markdown

```markdown
# My Portfolio Website

A simple portfolio website built while learning Git and GitHub.

## Features
- Responsive design
- Smooth scrolling navigation
- Clean and modern layout

## Technologies Used
- HTML5
- CSS3
- JavaScript

## How to Run
1. Clone this repository
2. Open `index.html` in your browser

## Learning Goals
- Practice Git commands
- Understand GitHub workflow
- Build a real project
```

Step 3: Git Workflow Practice

bash

```
# Check status
git status

# Stage files
git add .

# First commit
git commit -m "feat: Add initial portfolio structure"

# Create .gitignore
echo "*.log" > .gitignore
echo ".DS_Store" >> .gitignore

# Commit gitignore
git add .gitignore
git commit -m "chore: Add gitignore file"
```

Step 4: Create GitHub Repository

1. Go to GitHub.com
2. Click "New Repository"
3. Name: "my-portfolio"
4. Don't initialize with README (we have one)
5. Click "Create Repository"

Step 5: Connect and Push

bash

```
# Add remote origin
git remote add origin https://github.com/YOUR_USERNAME/my-portfolio.git

# Push to GitHub
git branch -M main
git push -u origin main
```

Step 6: Practice Branching

bash

```bash
# Create feature branch
git checkout -b feature/improve-styling

# Make changes to style.css (add animations, colors, etc.)
git add style.css
git commit -m "feat: Improve website styling with animations"

# Switch back to main
git checkout main

# Merge feature
git merge feature/improve-styling

# Push changes
git push origin main

# Delete feature branch
git branch -d feature/improve-styling
```

Step 7: Enable GitHub Pages

1. Go to your repository on GitHub
2. Settings → Pages
3. Source: Deploy from branch
4. Branch: main
5. Your site will be live at: `https://YOUR_USERNAME.github.io/my-portfolio`

Project Extensions

Once comfortable, try these:

- Add more pages
- Create a blog section
- Add a contact form
- Use different branches for different features
- Collaborate with friends
- Add CSS animations
- Make it responsive

Quick Reference Card

bash

```
# Daily Git Commands
git status              # Check status
git add .               # Stage everything
git commit -m "msg"     # Commit with message
git push                # Upload to GitHub
git pull                # Download updates

# Branch Commands
git branch              # List branches
git checkout -b new     # Create & switch branch
git merge branch        # Merge branch
git branch -d branch    # Delete branch

# Undo Commands
git checkout -- file    # Undo file changes
git reset HEAD file     # Unstage file
git commit --amend      # Edit last commit
```

Congratulations!

You've learned Git basics and built a real project! You now understand:

- Version control concepts
- Essential Git commands
- GitHub workflow
- Branching and merging
- Collaboration basics
- Real-world project structure

Keep practicing by contributing to open source projects or building more personal projects. The more you use Git, the more natural it becomes!