Git Basics: A Friendly Guide for Beginners

Hey there! 👋 Welcome to your journey into Git—the tool that makes tracking changes in your code (and life!) a whole lot easier. If you've ever wished for a time machine for your code, you're in the right place.

What You'll Learn

- What Git is (in plain English!)
- Setting up Git on your computer
- The essential commands you'll use daily
- How to work with GitHub
- Real examples using our Rotto Rocks project

Why Git Matters

Imagine you're writing a story, and every time you make a change, you save it as "story_final.txt", "story_really_final.txt", "story_absolutely_final_v2.txt"... Sound familiar? Git solves this problem by keeping track of all your changes in a smart way.

Getting Started

🛠️ First Things First: Installation

Let's get Git installed on your computer. Don't worry—we'll walk through this step-by-step for every operating system.

Windows Users

1. Download Git from git-scm.com
2. Run the installer (Next, Next, Install—we'll tell you which options matter)
3. Open Command Prompt or PowerShell to verify with git --version

Mac Users

1. Open Terminal
2. Type git --version
3. If not installed, you'll be prompted to install the developer tools

Linux Users

Ubuntu/Debian:

1. sudo apt-get update
2. sudo apt-get install git

🎨 Setting Up Your Identity

Before we dive in, let's tell Git who you are. Open your terminal and type:

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

Your First Git Project

Let's use our Rotto Rocks project as an example. We'll start with a simple webpage and see how Git can help us track changes.

📁 Creating Your First Repository

A Git repository (or "repo") is like a project folder that Git watches for changes. Here's how to create one using your terminal:

Create a new folder for your project:

```
mkdir rotto-rocks
cd rotto-rocks
```

Tell Git to start watching this folder:

```
git init
```

Congratulations! You've just created your first Git repository. 🎉

🔄 The Three States of Git

Think of Git like a photography system:

1. Working Directory: Your workspace (like your photo shoot location)
2. Staging Area: Your photo selection process (choosing which photos to keep)
3. Repository: Your final album (where changes are permanently stored)

📸 Basic Git Commands

Let's learn the commands you'll use most often:

Checking Status

```
git status
```

This shows you what Git is tracking and what's changed. Think of it as your project's health check!

Adding Files

```
git add index.html
# or add all files:
git add .
```

This is like telling Git "Hey, I want to keep track of these changes!"

Committing Changes

```
git commit -m "Added header section to homepage"
```

This saves your changes with a message explaining what you did. Make your messages clear and meaningful!

🌟 Best Practices for Commits

Good commit messages are like notes to your future self (and others). Here's how to write them:

✅ **Do:**

- Be specific: "Added navigation menu with home and about links"
- Use present tense: "Add" instead of "Added"
- Keep it under 50 characters

❌ **Avoid:**

- Vague messages: "Updated stuff"
- Super long messages
- Multiple unrelated changes in one commit

🌿 Working with Branches

Branches let you work on different versions of your project at the same time. Think of them as parallel universes for your code!

- Create a new branch

```
git branch feature-contact-form
```

- Switch to that branch

```
git checkout feature-contact-form
```

- Or do both at once

```
git checkout -b feature-contact-form
```

🤝 Collaboration with GitHub

GitHub is like social media for your code. Here's how to get your project online:

1. Create a GitHub account at github.com
2. Create a new repository on GitHub
3. Connect your local repo to GitHub:

```
git remote add origin https://github.com/yourusername/rotto-rocks.git
git branch -M main
git push -u origin main
```

🆘 Common Problems and Solutions

"Help! I made a mistake!"

Don't worry! Here are some common fixes:

- Undo last commit (but keep the changes):
  `git reset --soft HEAD~1`

- Discard changes in a file:
  `git checkout -- filename.html`

- Oh no, wrong branch!
  `git checkout correct-branch-name`

🎯 Practice Project: Rotto Rocks Homepage

Let's put it all together with a real example. We'll create the Rotto Rocks homepage:

1. Create the project structure:
```
mkdir rotto-rocks
cd rotto-rocks
git init
```
2. Create your first files:
```
touch index.html styles.css
```
3. Add some basic HTML:
```
<!DOCTYPE html>
<html>
<head>
    <title>Welcome to Rotto Rocks</title>
</head>
<body>
    <h1>Discover Rottnest Island</h1>
</body>
</html>
```
4. Track your changes:
```
git add .
git commit -m "Create initial homepage structure"
```

📚 What's Next?

- Now that you've got the basics down, you might want to:
- Practice these commands with your own projects
- Explore GitHub's documentation at docs.github.com
- Watch for our upcoming intermediate Git tutorial

Check back soon for more Graphitedge tutorials on web development

Want to stay updated? Follow us on social media to know when new tutorials are released.

🎉 You Did It!

Remember, every developer starts somewhere, and you've just taken your first steps into version control. Be proud of that! If you get stuck, remember these three things:

- git status is your friend
- Take small steps
- Don't be afraid to experiment

Ready to learn more? Check out our Git tutorials, where we'll continue exploring features and real-world workflows!