**🌲 Stage 4 – Git Forest (Pro-Level Git Automation)**

**📁 Folder:** GetInspiredQuoteSite\_GIT\_Pro  
**🎯 Objective:** Learn how to automate, optimize, and professionalize Git usage with real-world tools like Git aliases, hooks, Husky, GitHub Actions, and squash commits for clean project history.

**✅ 1. Git Aliases – Shortcuts for Common Git Commands**

**🔍 What are they?**  
Git aliases are like nicknames for long commands. Instead of typing git checkout, you can type git co.

**📦 Why are they useful?**  
They save time, reduce typing, and make your Git workflow feel smoother and faster.

**🔧 How to create aliases:**

Open your terminal (PowerShell, CMD, Git Bash, etc.) and type:

git config --global alias.co checkout

git config --global alias.br branch

git config --global alias.cm commit

git config --global alias.st status

**✅ Usage Examples:**

git co main # Instead of git checkout main

git br # Instead of git branch

git cm -m "feat: add nav" # Instead of git commit -m "feat: add nav"

git st # Instead of git status

**✅ 2. Git Hooks – Automate Checks Before You Commit**

**🔍 What are they?**  
Git hooks are scripts that run automatically at specific points in the Git lifecycle. The most common is the **pre-commit** hook — it lets you check your code before you commit it.

**💡 Example Use Case:**  
Automatically run npm test before every commit. If tests fail, the commit is blocked. This protects your codebase.

**🛠 How to set up a manual pre-commit hook:**

1. Go to the .git/hooks folder inside your Git project.
2. Create or edit a file called pre-commit (no file extension).
3. Add the following content:

#!/bin/sh

echo "Running tests before commit..."

npm test || exit 1

1. Make the file executable:

chmod +x .git/hooks/pre-commit

1. Now, every time you do a git commit, the script will run. If npm test fails, the commit won’t go through.

**✅ 3. Husky – Modern Git Hook Manager for JavaScript Projects**

**🔍 What is Husky?**  
Husky is a tool that simplifies Git hooks, especially in Node.js or React projects. It lets you manage hooks through your package.json.

**🪜 Step-by-step Setup for Husky:**

1. **Initialize your project:**

npm init -y

1. **Install Husky:**

npm install husky --save-dev

1. **Initialize Husky in your project:**

npx husky install

1. **Tell npm to install Husky hooks on every install:**

Add this in your package.json under "scripts":

"scripts": {

"prepare": "husky install"

}

1. **Add a pre-commit hook that runs tests before committing:**

npx husky add .husky/pre-commit "npm test"

1. **Try a commit:**

Make a change, stage it with git add ., and then run:

git commit -m "test: trying husky hook"

If tests fail, the commit will be blocked — just like we want.

**✅ 4. GitHub Actions – CI/CD on the Cloud**

**🔍 What is GitHub Actions?**  
GitHub Actions lets you automate workflows like testing, building, and deploying code every time you push code to GitHub or open a pull request.

**🛠 How to Set It Up:**

1. In your project root, create the following folders and file:

.github/workflows/test.yml

1. Inside test.yml, paste the following:

name: Run Tests

on: [push, pull\_request]

jobs:

test:

runs-on: ubuntu-latest

steps:

- uses: actions/checkout@v2

- name: Setup Node.js

uses: actions/setup-node@v3

with:

node-version: 18

- run: npm install

- run: npm test

1. Commit and push your changes to GitHub:

git add .

git commit -m "ci: setup GitHub Actions for tests"

git push origin main

1. Go to your repo on GitHub → Click **"Actions"** tab — you'll see the CI job running on every push or PR.

**✅ 5. Squash Commits – Clean Up Your History**

**🔍 What is it?**  
Squashing combines multiple messy commits into one clean commit. This is helpful when you’ve made several small changes (like fixing typos) and want to merge them together.

**🪜 How to do it:**

git rebase -i HEAD~3

You’ll see something like:

pick 123abc fix navbar bug

pick 456def added console log

pick 789ghi removed log

Change the second and third pick to squash:

pick 123abc fix navbar bug

squash 456def added console log

squash 789ghi removed log

Then Git will ask you to write a new commit message. Combine them into something like:

fix: cleanup navbar bug and console logs

Save and close the editor. Now, 3 commits are merged into 1.

**🔄 Optional Tools You Can Add:**

| **Tool** | **Purpose** |
| --- | --- |
| Prettier | Automatically format your code on save |
| ESLint | Detect and fix JavaScript issues early |
| Husky | Prevent bad commits using Git hooks |
| GitHub Actions | Automatically run tests and build workflows |

**✅ Final Summary of What I Learned**

* 🧠 Set up Git **aliases** for speed.
* 🧪 Used manual **Git hooks** to run tests before commit.
* 🐶 Installed and configured **Husky** for automated pre-commit checks.
* 🤖 Built **GitHub Actions** workflow to run tests on push/PR.
* 🧹 Practiced **interactive rebase** to squash commits into one clean history.