Beginner’s Guide to Version Control with Git: Branching, Merging & Collaboration

Subtitle: Learn how to manage code versions, collaborate with others, and work efficiently using Git.

Website Name: haas.dev
Website Link: https://dev-roast-app.vercel.app

Introduction

Git is the standard **version control system** used by developers to track changes, collaborate with teams, and manage projects. This guide teaches beginners the core Git concepts and practical commands to work confidently on real projects.

Step 1: Understanding Git Basics

- Git tracks **changes in your code** over time
- Key concepts:
    - **Repository (repo):** Project folder tracked by Git
    - **Commit:** Snapshot of changes
    - **Branch:** Separate line of development
- Local vs Remote repositories

Step 2: Setting Up Git

```
# Install Git
git --version

# Configure username and email
git config --global user.name "Hafsa Waseem"
git config --global user.email "hafsa@example.com"
```

Step 3: Initialize a Repository

```
git init      # Create a local repo
git clone <repo_url>  # Clone remote repo
```

- **Takeaway:** `init` = start project, `clone` = copy existing repo

---

Step 4: Making Changes & Commits

```
git status         # Check changes
git add .          # Stage all changes
git commit -m "Initial commit"  # Save changes with message
```

- Commit often with clear messages

Step 5: Branching

- Branches let you work on **features independently**

```
git branch feature-login    # Create branch
git checkout feature-login  # Switch to branch
git checkout main           # Return to main
```

- **Tip:** Always create a new branch for features/bugs

Step 6: Merging Branches

- Merge feature branch into main

```
git checkout main
git merge feature-login
```

- Resolve conflicts if Git cannot merge automatically

Step 7: Working with Remote Repositories

```
git remote add origin <repo_url>  # Link local repo to remote
git push -u origin main           # Push changes
git pull origin main              # Pull updates
```

Step 8: Collaboration Workflow

1. Fork or clone a repo
2. Create a branch for your feature

3. Make changes → commit → push
4. Open a **Pull Request (PR)**
5. Team reviews → merge to main

Step 9: Undoing Mistakes

- Undo last commit but keep changes staged:

```
git reset --soft HEAD~1
```

- Undo changes in a file:

```
git checkout -- filename
```

- **Takeaway:** Git allows safe experimentation

Step 10: Mini Projects to Practice

- Personal portfolio with version history
- To-Do App with feature branches
- Collaborate with a friend on a small project using GitHub

Key Takeaways

- Git tracks changes, enables collaboration, and protects your code
- Branching & merging help manage features safely
- Always commit often with meaningful messages
- Practice by working on real projects and collaborating on GitHub

---

Visit **haas.dev** for step-by-step Git tutorials, branching workflows, and beginner-friendly project collaboration guides.

Website Name: haas.dev
Website Link: https://dev-roast-app.vercel.app
