Beginner’s Guide to Version Control with Git & GitHub

Subtitle: Learn how to track, manage, and share your code like a professional developer.

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

Introduction

Version control is essential for developers. Git allows you to **track changes, collaborate, and recover code**, while GitHub provides a platform to share your projects. This guide shows beginners how to start using both effectively.

Step 1: Install Git

- **Website:** https://git-scm.com
- **Instructions:**
    1. Download Git for your OS (Windows/Mac/Linux)
    2. Install and configure with your name and email:

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

Step 2: Initialize a Repository

- Navigate to your project folder in terminal
- Run:

```
git init
```

- **Takeaway:** This creates a local Git repository for version control

---

Step 3: Stage & Commit Changes

- Stage files:

```
git add .
```

- Commit changes with a message:

```
git commit -m "Initial commit"
```

- **Tip:** Commit often with meaningful messages

Step 4: Create a GitHub Repository

- Go to https://github.com and create a new repo
- Copy the remote URL

Step 5: Connect Local Repo to GitHub

- Add remote:

```
git remote add origin https://github.com/username/repo.git
```

- Push your code:

```
git push -u origin main
```

- **Takeaway:** Your code is now online and backed up

Step 6: Work with Branches

- Create a new branch:

```
git checkout -b feature-branch
```

- Merge branch into main after changes:

```
git checkout main
git merge feature-branch
```

- **Tip:** Use branches for new features or experiments

Step 7: Pull, Fetch, and Sync

- Keep your repo updated:

```
git pull origin main
```

- **Takeaway:** Collaborating with others requires syncing frequently

---

Step 8: Resolve Conflicts

- Conflicts happen when multiple edits clash
- Open conflicted files, fix issues, stage, and commit
- **Tip:** Always test after resolving conflicts

Step 9: Best Practices

- Commit early and often
- Use descriptive commit messages
- Use branches for features or fixes
- Push regularly to GitHub

Step 10: Showcase Your Projects

- Keep all your portfolio projects on GitHub
- Include README.md for each project
- Share GitHub links in resumes, LinkedIn, and portfolios

Key Takeaways

- Git + GitHub = essential tools for professional developers
- Track changes, manage versions, and collaborate with confidence
- Regular commits, branches, and deployments improve workflow
- Every beginner should start using Git from day one

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

Visit **haas.dev** for step-by-step project guides, Git tutorials, and beginner-friendly coding resources.

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