Git Lab Assignment: Building a Collaborative Portfolio Repository

This lab is designed for beginners to early intermediate students in the DevOps course, aligning with Sessions 7 and 8 on Git Fundamentals. It builds on the in-class exercises and homework, reinforcing key concepts like local workflows, branching, merging, conflict resolution, and remote collaboration via GitHub. Students will create a simple "personal portfolio" repository, simulate team collaboration, and practice safe workflows.

Lab Objectives

- Master basic Git commands for local version control.
- Practice branching and merging to develop features safely.
- Resolve merge conflicts in a realistic scenario.
- Use GitHub for remote repositories, forking, and pull requests (PRs).
- Understand how Git enables collaborative DevOps practices.

Prerequisites

- Git installed and verified (*git --version*).
- A free GitHub account.
- Basic text editor (e.g., VS Code) for editing files.
- Completion of Session 7 homework (cloning and editing a repo).

Materials Provided

- None required, but use a simple starter template if desired (e.g., create a basic README.md file).
- Optional: Use VS Code's Git integration for visualization.

Step-by-Step Instructions

1. **Set Up Your Local Repository**
    - Create a new directory for your project: *mkdir my-portfolio && cd my-portfolio.*

- Initialize a Git repository: *git init*.
- Create a *README.md* file with initial content (e.g., using echo "# My DevOps Portfolio" > README.md or edit in your text editor). Add a brief introduction about yourself.
- Stage and commit the file: *git add README.md* followed by *git commit -m "Initial commit: Add README".*
- Verify with *git status* and *git log --oneline* to see your commit history.
- **Checkpoint:** Your repo should have at least one commit. Take a screenshot of *git log* for submission.

2. **Work with Branches for Feature Development**
    - Create a new branch for a feature: *git branch feature/about-me (or git switch -c feature/about-me)*.
    - Switch to the branch: *git checkout feature/about-me.*
    - Add a new file *about.md* with content like your skills or interests (e.g., "Skills: Linux, Bash Scripting").
    - Stage and commit: *git add about.md* and *git commit -m "Add about me section".*
    - Make another change: Edit README.md to link to about.md (e.g., add "About Me"). ***
    - Commit the update: *git add README.md* and *git commit -m "Update README with link".*
    - Switch back to main: *git checkout main.*
    - Merge the feature: *git merge feature/about-me.*
    - **Checkpoint:** Run *git log --graph --oneline* to visualize the branch merge. No conflicts yet—ensure the merge succeeds.

3. **Simulate and Resolve Merge Conflicts**
    - Create another branch: *git branch feature/projects.*
    - Switch to it: *git checkout feature/projects.*
    - Edit *README.md* to add a "Projects" section (e.g., add "## Projects-DevOps Lab 1").
    - Commit: *git add README.md* and *git commit -m "Add projects section".*
    - Switch back to main: *git checkout main.*

- Now, intentionally create a conflict: Edit *README.md* on main to add a different "Projects" section (e.g., "## Projects- Git Basics").
- Commit on main: *git add README.md* and *git commit -m "Add projects on main".*
- Attempt to merge: *git merge feature/projects*. This would trigger a conflict!
- Resolve the conflict: Open *README.md,* remove conflict markers (<<<<<<<, =======, >>>>>>>), and keep the best changes (e.g., combine both sections).
- Stage and commit the resolution: *git add README.md* and *git commit -m "Resolve merge conflict in README"*.
- **Checkpoint:** Run *git status* to confirm resolution. Screenshot the resolved file and commit message.

4. **Remote Collaboration with GitHub**
   - Create a new repository on GitHub (name it "my-portfolio", make it public or private).
   - Add the remote: *git remote add origin* https://github.com/your-username/my-portfolio.git (replace with your repo URL).
   - Push to remote: *git push -u origin main*.
   - To simulate collaboration : Create a new feature branch in the same repo.
     - Locally: *git checkout -b feature/enhancement.*
     - Make a change (e.g., add a skills.md file with content like "DevOps Skills: Git, Docker").
     - Commit: *git add skills.md* and *git commit -m "Add skills file".*
     - Push the branch: *git push origin feature/enhancement.*
   - On GitHub, create a Pull Request (PR): Go to your repo, select the "Pull requests" tab, click "New pull request", choose feature/enhancement as the compare branch and main as the base. Add a description like "Adds skills section for better portfolio."
   - Review and merge the PR on GitHub (you can approve and merge it yourself for this solo exercise).
   - Pull changes back locally: git checkout main and git pull to sync the merge.

    - **Optional for Real Collaboration:** Pair with a classmate, fork *their* repo, make changes on a branch in your fork, push, and create a PR back to their original repo. They can review and merge.
    - **Checkpoint:** Screenshot your GitHub PR page showing the creation and merge.

Submission Guidelines

- Push your final repo to GitHub and share the repo URL (e.g., via class platform).
- Include a lab-report.md file in your repo with:
    - Screenshots from checkpoints.
    - A brief reflection: What was challenging? How does Git help in DevOps teams?
    - Any issues faced and how resolved.
- Due: After Session 8.

Tips and Best Practices

- Use *git status* frequently to avoid mistakes.
- Commit messages should be clear and descriptive (e.g., "Fix typo in README" not "Update").
- If stuck, refer to the Git cheat sheet from class or git help <command>.
- Collaborate if possible: Invite a peer as a collaborator on GitHub for real PR reviews (add them via repo settings > Collaborators).
- Common pitfalls: Forgetting to pull before pushing, or not resolving conflicts fully—always verify with git log. For PRs in the same repo, remember GitHub allows this directly for safe, reviewed merges.

Submission:

Click here to submit