**Lab Assignment: Git Rollbacks, Merge Conflicts, Rebasing, and Reverting**

**Overview**

In this lab, you will work with Git to:

* Initialize a repository and create a commit history.
* Practice different methods of rolling back changes.
* Create and resolve a merge conflict.
* Use both rebasing and merging techniques.
* Revert a specific commit.

**Estimated Completion Time:** 1 hour

**Part A: Setup and Initial Commits**

1. **Repository Setup:**
   * Create a new directory for the lab.
   * Initialize a new Git repository.
   * Create an initial file (e.g., app.py or index.html) with some content.
   * Stage and commit your changes.
2. **Build the Commit History:**
   * Make at least two additional changes to your file(s) and commit each change.
   * Verify that your commit history contains multiple commits.

**Part B: Rollbacks in Git**

1. **Explore Commit History:**
   * Use a Git command to view your commit history and identify commit hashes.
2. **Reverting Changes:**
   * Use a Git command to revert the latest commit by creating a new inverse commit.
   * Verify the changes in your file to see the effect of the revert.
3. **Resetting the Repository:**
   * Practice using both soft and hard resets to move the repository state backward.
   * Observe and document the differences in how each reset type affects the working directory and staging area.
4. **Reflection:**
   * Briefly describe the role of HEAD in Git and how it changes during resets.

**Part C: Merge Conflict Resolution**

1. **Branch Creation and Divergence:**
   * Create a new branch to simulate parallel development.
   * Modify the same section of a file differently on this new branch and commit the change.
   * Switch back to the main branch, make a different modification to the same section, and commit.
2. **Merging and Conflict:**
   * Attempt to merge the new branch into the main branch.
   * Encounter and identify the merge conflict.
3. **Conflict Resolution:**
   * Manually resolve the merge conflict by editing the conflicting file.
   * Stage and commit the resolution.
4. **Reflection:**
   * Explain how the merge conflict occurred and outline the steps you took to resolve it.

**Part D: Rebasing vs. Merging**

1. **Branch Experimentation:**
   * Create another branch for experimenting with rebasing.
   * Make a couple of commits on this branch.
2. **Main Branch Update:**
   * Switch back to the main branch and make an update commit.
3. **Perform a Rebase:**
   * Rebase the experimental branch onto the updated main branch.
   * Resolve any conflicts that arise during the rebase process.
4. **Comparison:**
   * Reflect on the differences between merging and rebasing in terms of commit history and workflow.
   * Document the advantages and potential pitfalls of each method.

**Part E: Reverting a Specific Commit**

1. **Commit Identification:**
   * Identify a specific commit from your history that you want to undo.
2. **Reverting:**
   * Use Git to revert that specific commit.
   * Verify that a new commit has been created which undoes the changes introduced by the original commit.
3. **Reflection:**
   * Summarize how reverting a commit differs from resetting the repository.

**Deliverables**

* **Git Repository:** Ensure your repository includes:
  + A series of commits reflecting your initial work, rollback attempts, merge conflict resolution, rebasing activity, and the specific revert.
* **Documentation:** Provide a brief written summary that covers:
  + How HEAD moves during the various rollback operations.
  + Your approach to resolving merge conflicts.
  + A comparison of rebasing versus merging and their impacts on project history.