Advanced Git Techniques

Scenario:

In this activity, you will practice advanced Git techniques for managing branches and repositories. By completing the steps, you will gain hands-on experience with creating branches, resolving conflicts, using tags, stashing changes, and performing interactive rebases.

Instructions:

Step 1: Create and Manage Branches

1. Open your terminal and navigate to your Git project directory.
2. Create and switch to a new branch named feature-new-feature.
3. Make changes to any file in the repository.
4. Stage and commit the changes to the branch with a meaningful message.

Step 2: Merge Branches and Resolve Conflicts

1. Switch to the main branch.
2. Make and commit changes to the same file you modified in the feature branch.
3. Switch back to the feature-new-feature branch and make additional changes to the file.
4. Commit these changes in the feature branch.
5. Attempt to merge the feature-new-feature branch into the main branch.
6. If conflicts occur:
    - Open the conflicted file in a text editor.
    - Resolve the conflicts by keeping the desired changes.
    - Stage and commit the resolved file to complete the merge.

Step 3: Use Tags to Mark Important Commits

1. Create an annotated tag named v1.0 to mark the current state of your repository.
2. Push the tag to the remote repository.

Step 4: Stash Changes Temporarily

1. Switch to the feature-new-feature branch and make changes to a file without committing them.
2. Stash the changes to save them temporarily.
3. Switch to the main branch.
4. View the list of stashed changes.
5. Apply the stashed changes back to your working directory.

Step 5: Perform Interactive Rebase

1. Start an interactive rebase for the last 4 commits in the feature-new-feature branch.
2. In the rebase editor:
    - Use pick to keep a commit.
    - Use squash to combine commits.
    - Use edit to modify a commit message.
3. Save the changes and continue the rebase process.
4. If conflicts occur:
    - Resolve them and stage the resolved file.
    - Continue the rebase process.
5. Force push the changes to the remote repository.

Git code:

```
# === Lab: Advanced Git Techniques ===

# Initialize project

mkdir lab-advanced-git

cd lab-advanced-git

git init


# Create initial file and first commit

echo "# Advanced Git Lab" > README.md
```

```
git add README.md

git commit -m "Initial commit with README"


# Step 1: Create and Manage Branches

git checkout -b feature-new-feature

echo "Feature: Add login form" >> README.md

git add README.md

git commit -m "Add login form section in README"


# Step 2: Merge Branches and Resolve Conflicts

git checkout main

echo "Main: Add welcome message" >> README.md

git add README.md

git commit -m "Add welcome message section in README"


git checkout feature-new-feature

echo "Feature: Add logout info" >> README.md

git add README.md

git commit -m "Add logout info to README"


git checkout main

git merge feature-new-feature || echo "Merge conflict occurred"


# Simulate manual conflict resolution

echo "Final version: login, welcome, logout" > README.md

git add README.md

git commit -m "Resolve merge conflict between feature and main"


# Step 3: Use Tags to Mark Important Commits
```

```
git tag -a v1.0 -m "Release version 1.0"

# Step 4: Stash Changes Temporarily
git checkout feature-new-feature
echo "Uncommitted test notes" >> README.md
git stash
git checkout main
git stash list
git stash apply

# Step 5: Perform Interactive Rebase (manual editor required)
git checkout feature-new-feature
git commit --allow-empty -m "Temp commit 1"
git commit --allow-empty -m "Temp commit 2"
git commit --allow-empty -m "Temp commit 3"
git commit --allow-empty -m "Temp commit 4"
git rebase -i HEAD~4
# In the interactive editor: pick, squash, edit as needed, then:
# If "edit" is used:
git commit --amend
git rebase --continue

# Simulate conflict resolution during rebase
echo "Rebase conflict resolved" > README.md
git add README.md
git rebase --continue

# Final step: force push (if remote exists)
# git push origin feature-new-feature --force
```