Open Source Software Development

Homework assignment 4

Exercise 1 Conceptual Questions

Answer the following questions in your own words. Add a reference to the corresponding slide of the lecture to your answer.

1. **Explain the difference between `git pull` and `git fetch`. When would you use each?**
2. **What does `git rebase` do? How is it different from `git merge`?**
3. **Why might a team prefer to use rebase instead of merge in their development workflow?**
4. **What are the potential dangers of rebasing shared branches, and how can you avoid them?**

Exercise 2 Git Rebasing

In this exercise, you will simulate working on multiple feature branches and practicing rebasing locally.

Setup

1. Create a new directory for the assignment:
   ```
   mkdir git-rebase-homework && cd git-rebase-homework
   git init
   ```
2. Create a `main` branch with a base project:
   ```
   echo "Initial project setup" > project.txt
   git add project.txt
   git commit -m "Initial commit on main"
   ```
3. Create a branch for Feature A:
   ```
   git checkout -b feature-a
   echo "Feature A - line 1" >> project.txt
   git commit -am "Add Feature A line"
   ```
4. Go back to `main` and simulate work done in parallel:
   ```
   git checkout main
   echo "Hotfix on main - line 1" >> project.txt
   git commit -am "Add hotfix to main"
   ```
5. Now rebase `feature-a` onto the updated `main`:
   ```
   git checkout feature-a
   git rebase main
   ```
6. If there's a conflict, resolve it manually in `project.txt`, then:
   ```
   git add project.txt
   git rebase --continue
   ```

Deliverables

Combine your answer in one pdf file. It should contain the content of **`project.txt`** as the final version after rebasing, showing both the hotfix and Feature A.

A step-by-step log of commands you ran, with brief descriptions. Example:

```
git checkout -b feature-a   # start work on a new feature
...
git rebase main   # integrate changes from main into feature-a
```

Bonus Task (Optional)

Create a second feature branch (`feature-b`) and repeat the process. Then use interactive rebase to reorder and squash commits:

```
git rebase -i main
```

Submit your updated `git_log.txt` with a note about what changes you made during interactive rebase.