30538 Problem Set 3: git Solution

Aurora Zhang

2024-10-24

1. Late coins used this pset: **_1_**
2. Late coins left after submission: **_2_**
3. "I have uploaded the names of anyone I worked with on the problem set here" **_AZ_** (1 point)
4. Knit your ps3_solo.qmd as a pdf.
5. Push ps3_solo.qmd and ps3_solo.pdf to your github repo. Use command line git (not github desktop)
6. Submit ps3_solo.pdf through the Problem Set 3 Solo assignment on Gradescope. (4 points)
7. Tag your submission in Gradescope (applies only to the solo part) "This submission is my work alone and complies with the 30538 integrity policy." Add your initials to indicate your agreement: **_AZ_**

problem set setup (5 points)

SOLO

Learn git branching (15 points)

Go to https://learngitbranching.js.org. This is the best visual git explainer we know of.

1. Complete all the levels of main "Introduction Sequence". Report the commands needed to complete "Git rebase" with one line per command.

```
git checkout -b bugFix
git commit
git checkout main
git commit
git checkout bugFix
git rebase main
```

2. Complete all the levels of main "Ramping up". Report the commands needed to complete "Reversing changes in git" with one line per command.

```
git reset HEAD~1
git checkout pushed
git revert HEAD
```

3. Complete all the levels of remote "Push & Pull – Git Remotes!". Report the commands needed to complete "Locked Main" with one line per command.

```
git checkout -b feature
git push origin feature
git checkout main
git reset --hard o/main
git checkout feature
```

Exercises

- Basic Staging and Branching (10-15)

1. Exercise. For your pset submission, tell us only the answer to the last question (22).

```
On branch master
nothing to commit, working tree clean
```

2. Exercise. For your pset submission, tell us only the output to the last question (18).

```
diff --git a/file1.txt b/file1.txt
deleted file mode 100644
index 28360e5..0000000
Binary files a/file1.txt and /dev/null differ
diff --git a/file2.txt b/file2.txt
new file mode 100644
index 0000000..9e9eb0b
Binary files /dev/null and b/file2.txt differ
```

- Merging

1. Exercise. After completing all the steps (1 through 12), run git log –oneline –graph –all and report the output.

```
* b68d25f (HEAD -> master) Change greeting to uppercase
* 52ddf0f Add content to greeting.txt
* 053c9f2 Add file greeting.txt
```

2. Exercise. Report the answer to step 11.

```
*   21b4bd8 (HEAD -> master) Merge branch 'greeting'
|\
| * 32496b8 (greeting) Add favorite greeting to greeting.txt
* | 48541c2 Add README.md with repository information
|/
* cd1f479 Add content to greeting.txt
* 6ff4318 Add file greeting.txt
```

3. Identify the type of merge used in Q1 and Q2 of this exercise. In words, explain the difference between the two merge types, and describe scenarios where each type would be most appropriate.

Fast-Forward Merge: This occurs when the master branch has no new commits since branching. Git simply moves the master pointer to the latest commit on the feature branch, without creating a new commit. It keeps the history linear. This merge type is ideal when integrating a feature branch into master without parallel changes, providing a clean, simple history.

3-Way Merge: Used when both branches have diverged with unique commits. Git creates a new "merge commit" by combining changes from both branches and their common ancestor. This approach is suitable for teams working on separate features or making concurrent changes, as it maintains a detailed history of branching and merging activities.

In short, Fast-Forward is best for single-user, linear workflows, while 3-Way fits collaborative, multi-branch projects.

- Undo, Clean, and Ignore

1. Exercise. Report the answer to step 13.

```
commit af54ba260517defe28a641ac68cd1d1f72b53579
Author: git-katas trainer bot <git-katas@example.com>
Date:   Sun Oct 27 13:21:08 2024 -0500

    Add credentials to repository

diff --git a/credentials.txt b/credentials.txt
new file mode 100644
index 0000000..8995708
--- /dev/null
```

```
+++ b/credentials.txt
@@ -0,0 +1 @@
```

2. Exercise. Look up `git clean` since we haven't seen this before. For context, this example is about cleaning up compiled C code, but the same set of issues apply to random files generated by knitting a document or by compiling in Python. Report the terminal output from step 7.

```
Removing README.txt~
Removing obj/
Removing src/myapp.c~
Removing src/oldfile.c~
```

3. Exercise. Report the answer to 15 ("What does git status say?")

```
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        deleted:    file1.txt

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   .gitignore

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        file1.txt
        file2.txt
        file3.txt
        foo.s
```