<Brian W. Lee>

CSC 2510: DevOps

Lab 11 – Cooperative Git

General Instructions

Using your book and previous lecture material, fill out this assignment sheet. **Use RED text to signify your answers.** You should utilize online resources to answer these questions as well.

Submission Instructions

To submit, **change the name in the header** and save this document as a PDF. Attach your PDF document to the iLearn dropbox. **The changes (The tag) must be pushed in order for your lab to be graded.**

Lab Directions

1. For this lab we will simulate a cooperative, development environment between two people: Alice and Bob. In your **home directory**, clone your repository twice. Once to a folder named `Alice` and once to a folder named `Bob`. **In each folder you should be working directly in the `dev` branch for this lab unless otherwise specified.**
2. In `Alice`'s folder:
    a. Create a folder called `lab-11`
    b. Within that folder create a file called `cat.txt`.
        i. You may put whatever contents you like into this file.
    c. Be sure to commit and push your changes.
3. In `Bob`'s folder:
    a. **DO NOT PULL** the changes you pushed from `Alice`'s folder.
    b. Create a folder called `lab-11`.
    c. Create a file called `dog.txt`.
        i. Again, put whatever you like in its contents.
    d. Commit the file, but do not push it.
    e. Run `git log --all --graph --decorate --oneline`
    f. Run `git pull`
    g. Run the `log` command again.
    h. Do an `ls` command and note that the changes from `Alice`'s folder are present in this folder.
    i. Push Bob's work.
4. In `Alice`'s folder:
    a. Run a `git pull`.

   b. Run the `git log` command and compare it to `Bob`'s.
   c. Identify where `HEAD`.
   d. Do a `git log` and identify the commit *hash* from just before you added the `cat.txt` and `dog.txt` files.
   e. Run `git checkout <your-hash>`.
   f. Note the contents of the repository.
   g. Identify where `HEAD` is now.
   h. Can you move `HEAD` to the commit where `dog.txt` was created? Is `dog.txt` there? Is `cat.txt` there? Why or why not? Moving head does not do anything
   i. Checkout `master`.
   j. Where is `HEAD`? What do your files look like? What does `git checkout` do?
5. Ensure that both `Alice`'s and `Bob`'s folders are on `dev` and in the same state.
6. In `Alice`'s folder:
   a. Change the first line of `cats.txt` to `Alice likes cats`.
   b. Commit and push the changes.
7. In `Bob`'s folder:
   a. **DO NOT PULL** the changes you pushed from `Alice`'s folder.
   b. Change the first line of `cats.txt` to `Bob likes cats`.
   c. Commit the changes and attempt to push.
   d. You should receive and error message. What is it? REJECTED
   e. Do a `git pull`.
   f. Resolve the resulting conflict. So that both `Alice and Bob like cats`.
   g. Commit and push your changes.
8. In `Alice`'s folder, pull the changes. Ensure that `Bob`'s folder and `Alice`'s folder are in the same state.
9. Merge your changes to master.
10. Return to your original clone of your repository.
11. Tag your last commit as `L.11`.

Lab Questions

1. (2) What happens when you run `git log --merges` in your `master` branch? Shows all the merge history. How does this differ from your answer in your last lab? It didn't show before.
2. (4) What is a merge conflict and how do you resolve it? Initially, Alice and Bob had different contents in cats.txt, so the conflict happened when Bob tried to git pull. To fix the issue, I changed contents in cats.txt from Bob's folder to "Alice and Bob like cats." then git add, commit, and pushed. Finally, I git pull from Alice.
3. (2) What happens when you use the command `git log -S 'Alice'`? Shows all the commit history and it messages made by Alice.
4. (2) How would you fix a mistyped commit message if it was the last commit you made? You would use "git commit --amend" to edit the last commit message.