<Brian W. Lee>

CSC 2510: DevOps

Lab 10 – Git Branching

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. **If your changes are not pushed, this lab will not be graded.**

Lab Instructions

In addition to the commands from last lab session you will need to use the following:

- `git branch`
  - `git branch -a -v`
- `git tag`
- `git log`
  - `git log --all --graph --decorate --oneline`
- `git checkout`

1. Tag your last commit for your submission for lab 9 as `L.09`.
2. Be sure to push the tag to the remote.
3. Create and change to a branch called `dev`.
4. In your `dev` branch:
   a. Create a file called `animals.txt` and add in a line about your favorite animal.
   b. Add, commit, and push your change.
5. From `dev` create a new branch called `lab/10`.

<Brian W. Lee>

6. In your `lab/10` branch, in addition to regularly adding, committing, and pushing changes, do the following:
    a. Create a folder called `poetry`.
    b. Create a file called `dailygrind.java` in `poetry`.
    c. In `dailygrind.java`:
        i. Add in the following text:

```
import java.util.Date;

public class DailyGrind {
      public static final void main(String[] args) {
      }
}
```

    d. Use the `git log --all --graph --decorate --oneline` command to view the history of the branch. Make note of the output. Where is root displayed? towards the bottom What direction is it meant to be read? bottom up
    e. In `dailygrind.java`:
        i. Add the following code to the main function:

```
boolean its_time_to_go_home = false;
boolean away_the_hours = true;

while (away_the_hours) {
  boolean away_the_hours = true;

  Date now = new Date();
  its_time_to_go_home = now.getHours() > 17
  && now.getMinutes() > 30;

      if (its_time_to_go_home) {
            break;
       }
```

7. In your `dev` branch:
    a. Run the `git log` command found in step 6.d and compare it to the previous output. What's different? There is another history commit message with the corresponding branch.
    b. Merge your `lab/10` branch into `dev`.
8. In your `master` branch:
    a. Run the `git log` command from 6.d again.
    b. Merge your `dev` branch into `master`.
    c. Run `git log` one more time.
9. Make sure that your lab code is successfully merged into master.

<Brian W. Lee>

10. Delete your `lab/10` branch. You've successfully implemented a feature and released it! This branch is no longer needed. Ensure that this delete happens on remote as well as local.
11. Ohno! Someone has identified a bug in our `dailygrind.java` file. Create a `hotfix/10.1` branch and add the following code to fix the error at the end of the `while` loop.

```
try {
      Thread.sleep(60000);
} catch (InterruptedException e) {
      // ignore
}
```

12. Merge `hotfix/10.1` into both `master/main` and `dev` (so that each have the bug fix). Run the `git log` command from 6.d on `master/main` after you are done, then delete the hotfix branch. Ensure all of your changes appear on the remote. Run the `git log` command from 6.d after you've deleted the branch.
13. Tag your last commit for this lab as `L.10`. Ensure all your tags appear on remote with the correct commits. Feel free to experiment with additional branches and commits. Your tagged commit is what will be used for grading.

Note: The 'poem' we wrote is what is called 'code poetry'. The original poem can be found online and was written by Paul Illingworth.

Lab Questions

1. (5) What are branches in git? Branches are divergent development paths often used to develop work simultaeously, frequently joined together via "merges", and represented using an acyclic directed graph.
2. (5) What do each of the following commands do?
   - `git branch shows a list all the branches and which branch you are on`
   - `git tag shows a list of all the tags`
   - `git log shows a list of all the history of commits with full commit IDs with author and date and its messages applied to the tags and branches.`
   - `git log --all --graph --decorate --oneline shows a list of all the history of commits with partial commit IDs without author and date and its messages applied to the tags and branches`
   - `git checkout switches branches or restore working tree files`
3. (2) What happened when we deleted branches and then ran the `git log` command from 6.d? The deleted branch was not showing on the log.
4. (3) What happens when you run `git log --merges` in your `master` branch? Nothing happens How is this different than the other `git log` commands we've used? The other git log command shows a list of commits and current merges, but the "git log --merges" do not show anything.

<Brian W. Lee>

5. (5) Look up the documentation and use of the command `git blame`. Experiment with it in your repository. What does `git blame` do? It shows what revision and author last modified each line of a file which can be specified from which revision up to last modified line, set the range of lines of a particular file, and many optional variants with command parameters that follow after "git blame" to fit the need.