CMSC 6950

Assignment #4

Local and Remote Version Control with Git

Due to Monday, June 12, 2017

The purpose of this assignment is to practice the workflow of working with the Git version control system.

1 Tasks

1. Configure git setting your full name (user.name) and MUN email address (user.email) - use the @mun.ca address that is listed in D2L. **Important:** If the recorded commits have incorrect name and email address, the assignment will receive 0 points!!!

2. Create a new local git repository named "`cmsc_git_assignment`"

3. Create a file called "`README.md`" with the following content:

```
# CMSC6950 - Assignment 4 for {your full name}

Git is a distributed version control system that was created by
Linus Torvalds, the creator of the Linux kernel.
```

4. Commit this file with the message "Add first paragraph"

5. Add the second paragraph (below) to the file `README.md` and commit this change with the message "Add second paragraph".

```
The basic Git workflow consists of the following steps:

1. Create or edit files.
2. Add the new or changed files to the staging area.
3. Store your changes in the git database, by making a commit.
4. Choose an informative commit message. This helps at a later time to find
   specific commits and to understand the intention of the changes.
5. Continue at step 1.
```

6. Create and then checkout a new branch called "`my_branch`".
7. Add the third paragraph (below) to the file `README.md` and commit this change with the message: "`Add third paragraph`".

```
Unlike centralized version control systems, Git users can commit new changes,
without the need of an internet connection.  They can work offline and then later
push batches of commits at once to a server - like GitHub or Bitbucket.
```

8. Change the second paragraph to match the text below and commit with the message: "`Expanded Git workflow`".

```
The Git workflow involving a remote repository consists of the following steps:

1. Pull changes from the remote repository.
2. Create or edit files.
3. Add the new or changed files to the staging area.
4. Store your changes in the git database, by making a commit.
5. Choose an informative commit message. This helps at a later time to find
   specific commits and to understand the intention of the changes.
6. Push your commits to the remote repository to share them with others
   and be able to access them yourself on a different computer.
7. Continue at step 1.
```

9. Checkout the `master` branch.
10. Change the word "`database`" to "`repository`" and "`version control system`" to "`Version Control System (VCS)`" in the whole document and commit with the comment: "`Wording: database/repository`"
11. Merge branch "`my_branch`" into the `master` branch.
12. Resolve the merge conflict. Make sure that all the conflict markers have been removed and that no duplicate lines that arose from both versions are present.

    At this point you may use the command "`git log --oneline --graph --all`" and compare your output with the one in the assessment section. If not, you can start over by deleting the `.git` directory and `README.md` file.
13. Create a new (empty) repository called "`git_assignment`" on the GitHub website.
14. Set this GitHub repository as remote "`origin`" on your local repository.
15. Push the local repository - with all branches (master and my_branch) to GitHub. You can use the option `--all` or push the branches after each other.
16. Use the history command to redirect the commands of your final attempt into a text file. Submit a text file named `assignment_4_USERID.txt` with the URL of your GitHub repository (e.g.: "`https://github.com/USERID/cmsc_git_assignment`") and the git commands to the D2L drop-box. Make sure "**USERID**" is replaced by your user id.

2 Assessment

The final `README.md` file should read:

```
# CMSC6950 - Assignment 4 for {your full name}.

Git is a distributed Version Control System (VCS) that was created by
Linus Torvalds, the creator of the Linux Kernel.

The Git workflow involving a remote repository consists of the following steps:

1. Pull changes from the remote repository.
2. Create or edit files.
3. Add the new or changed files to the staging area.
4. Store your changes in the git repository, by making a commit.
5. Choose an informative commit message. This helps at a later time to find
   specific commits and to understand the intention of the changes.
6. Push your commits to the remote repository to share them with others
   and be able to access them yourself on a different computer.
7. Continue at step 1.

Unlike centralized version control systems, Git users can commit new changes,
without the need of an internet connection.  They can work offline and then later
push batches of commits at once to a server - like GitHub or Bitbucket.
```

After step 12 the output of `git log --oneline --graph --all` should look like this (the commit IDs will be different):

```
$ git log --oneline --graph --all
*   9638881 Merge branch 'my_branch'
|\
| * bcce03e Expanded Git workflow
| * b3014f2 Add third paragraph
* | 59beb13 Wording: database/repository
|/
* 3a761a5 Add second paragraph
* 7d591c7 Add first paragraph
```

You can also look at `https://github.com/ostueker/cmsc_git_assignment` for reference.

2.1 Marking Scheme

1. All commits need to have the correct name and email address. ....(1 point)
2. The repository should consist of the following commits: ....(6 points)

```
A--B--------E--M
    \-C--D----/
```

   - A: Add first paragraph
   - B: Add second paragraph
   - C: Add third paragraph
   - E: Expanded Git workflow [my_branch]
   - D: Wording: database/repository
   - M: Merge branch 'my_branch' [HEAD][master]

   Every missing commit costs 1 point.
   Every extraneous or incomplete commit costs 0.5 point.

3. There must be none of the conflict markers left in the merge commit. ....(1 point)
4. The changes of commits D and E must have been manually integrated. ....(1 point)
5. The repository has been pushed with both branches to GitHub. ....(1 point)

Total: ....10 points