POLYTECHNIQUE
MONTRÉAL

LOG1410
Analyse et conception de logiciels

Laboratoire 1

Soumis par:
Ioana Daria Danciu - 2081744
Alexandre Gélinas - 2083465

Le 8 octobre 2020

Équipe : 01 Groupe : 01

- Coéquipière 1 : Ioana Daria Danciu #2081744
- Coéquipier 2 : Alexandre Gélinas #2083465

Question 4.1.1

```
[gigl@localhost tp1-log1410_01l_1]$ git log
commit edc0032e9a719d414180eac59842747a89bb5c62 (HEAD -> master, origin/master, origin/HEAD)
Author: Alexandre Gélinas <gelinas.alexandre.2001@gmail.com>
Date:   Wed Jan 20 15:43:02 2021 -0500

    Création de Readme.txt

commit b3a466736ee1ff8df2b805097d2b555fa7e01c3d
Author: github-classroom[bot] <66690702+github-classroom[bot]@users.noreply.github.com>
Date:   Wed Jan 20 19:29:34 2021 +0000

    Initial commit
```

Figure 1: création et modification du fichier Readme.txt

Question 4.1.2

Git log -p montre les différences entre les versions tandis que git log montre uniquement les commits.

Question 4.3.1

```
[gigl@localhost save]$ git pull origin master
Username for 'https://github.com': lex0re
Password for 'https://lex0re@github.com':
remote: Enumerating objects: 7, done.
remote: Counting objects: 100% (7/7), done.
remote: Compressing objects: 100% (1/1), done.
remote: Total 4 (delta 2), reused 4 (delta 2), pack-reused 0
Dépaquetage des objets: 100% (4/4), 461 octets | 230.00 Kio/s, fait.
Depuis https://github.com/LOG1410-H21/tp1-log1410_01l_1
 * branch            master     -> FETCH_HEAD
   3efc3f8..f1ff713  master     -> origin/master
Fusion automatique de TP1_CodeSource/main.c
CONFLIT (contenu) : Conflit de fusion dans TP1_CodeSource/main.c
La fusion automatique a échoué ; réglez les conflits et validez le résultat.
[gigl@localhost save]$
```

Figure 2 : Conflits lors de 2 push consécutifs

Question 4.4.1

```
[gigl@localhost TP1_CodeSource]$ make clean
rm -rf newton *.o
[gigl@localhost TP1_CodeSource]$ make
gcc -o main.o -c main.c
gcc -o newton.o -c newton.c
gcc -o newton main.o newton.o -lm
[gigl@localhost TP1_CodeSource]$ ./newton 49 3
Compute the root value of 49.000000
With 3 iterations
Result = 7.128205
[gigl@localhost TP1_CodeSource]$
```

Figure 3 : Exécution de newton

Question 4.5.1

```
[gigl@localhost TP1_CodeSource]$ make clean
rm -rf newton *.o
[gigl@localhost TP1_CodeSource]$ make install
gcc -o newton.o -c newton.c
mkdir -p install
gcc -c newton.c
cp newton.o install/
[gigl@localhost TP1_CodeSource]$
```

Figure 3 : Création du dossier install avec l'exécutable

Question 4.6

1. Qu'est-ce qu'un Merge Request? Quelle est l'utilité d'un Merge Request?

C'est une demande à fusionner deux branches ensemble, de façon à voir les modifications apportées au fichiers. C'est utile car il permet d'avoir une meilleure communication avec les autres programmeurs travaillant sur le même projet. Ainsi, il est plus simple d'optimiser le code et de l'assembler. [1]

2. Sur la barre latérale gauche du projet, sélectionnez la section *Merge Requests* puis la colonne *All*, et entrez dans la barre de recherche le nom suivant : *Anti-Feature icons*. Décrivez brièvement ce que fait cette révision.

Cette révision nous permet de voir les *Merge Requests*, les *Merge* ainsi que les *commits* des membres de ce projet. Les divers pipelines, commentaires et autres modifications sont aussi affichés.

3. En cliquant sur l'onglet *Pipelines*, combien de commits passent les tests? Combien échouent les tests? Combien ont été annulés?

Quatre commits passent les tests, six commits échouent les tests et trois commits ont été annulés.

4. En cliquant sur l'onglet *Commits*, combien de commits passent le pipeline? Quel est le rôle des pipelines sur Gitlab?

Un seul commit passe le test. Les pipelines sont des travaux comme des tests ou des compilations pour bien vérifier l'avancement du projet en question. On définit des stages pour se fixer des objectifs et des points à atteindre dans notre projet. C'est un guide pour les programmeurs.[2]

5. Quel est le lien entre les tests et les pipelines sur Gitlab?

Les pipelines font des tests sur le code pour vérifier le programme en cours du projet et vérifier ainsi qu'il passe les objectifs.[2]

6. Pourquoi les pipelines comportant plusieurs étapes (comme https://gitlab.com/uniqx/ fdroidclient/pipelines/109834613) sont-ils utiles pour l'intégration continue?

Car ils permettent de déployer le programme dès qu'il passe les tests et voir donc un avancement du projet.[2]

Question 5.1

Nous avons chacun passé environ 4 heures et demi sur ce projet, pour un total de 9 heures-personnes. L'effort pour ce travail est adéquat.

Question 5.2

Les directives pourraient être plus claires en ce qui concerne la rédaction du rapport et GitLab.

Source :

[1] : GitLab (s.d.). « Merge Requests », sur le site *GitLab docs.* Consulté le 5 février 2020. Repéré à : https://docs.gitlab.com/ee/user/project/merge_requests/
[2] : GitLab (s.d.). « CI/CD pipelines » sur le site *GitLab docs.* Consulté le 5 février 2020. Repéré à : https://docs.gitlab.com/ee/ci/pipelines/