МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Кафедра инфокоммуникаций

«Основы ветвления Git»

Отчет по лабораторной работе № 1.3

по дисциплине «Основы кроссплатформенного программирования»

Выполнил студент группы ИВТ-б-о-21-1

Шайдеров Дмитрий Викторович.

«18 » мая 2022г.

Подпись студента_____

Работа защищена « »_____20__г.

Проверил Воронкин Р.А. _____
(подпись)

Ставрополь 2022

Цель работы: исследование базовых возможностей по работе с локальными и удаленными ветками Git.

Порядок выполнения работы:

1. Создал общедоступный репозиторий на GitHub, в котором использована лицензия MIT.

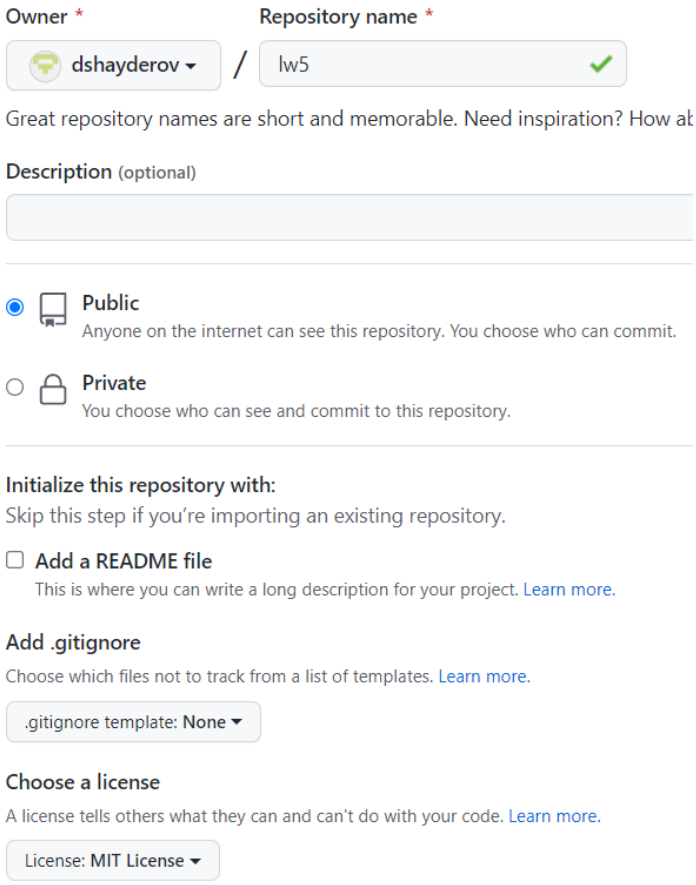

Рисунок 1 - Создание репозитория

2. Создал три файла: 1.txt, 2.txt, 3.txt.

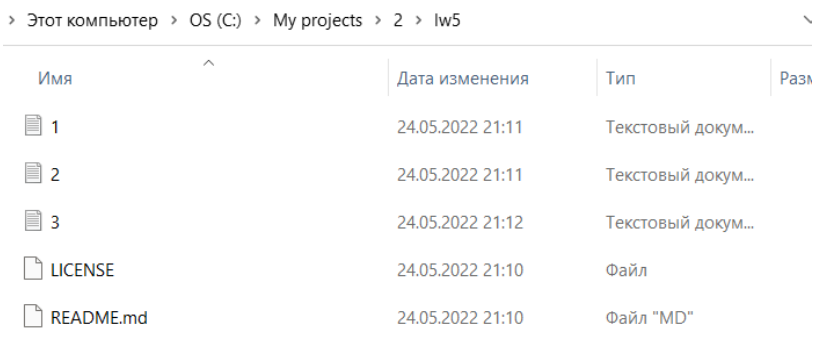

Рисунок 2 - Создание текстовых файлов

3. Проиндексировал первый файл и сделал коммит с комментарием "add 1.txt file".

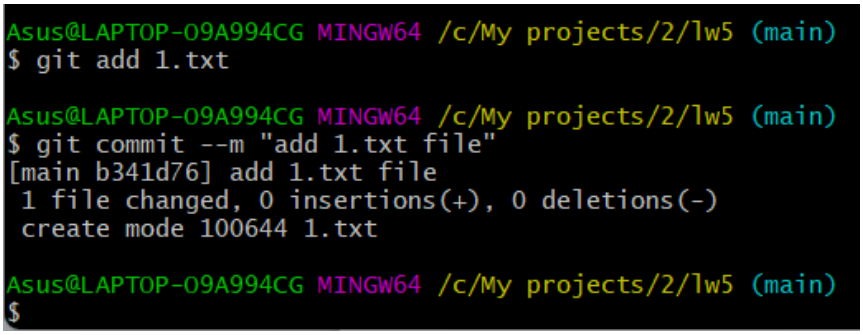

Рисунок 3 - Коммит 1

4. Проиндексировал второй и третий файлы. Перезаписал уже сделанный коммит с новым комментарием "add 2.txt and 3.txt."

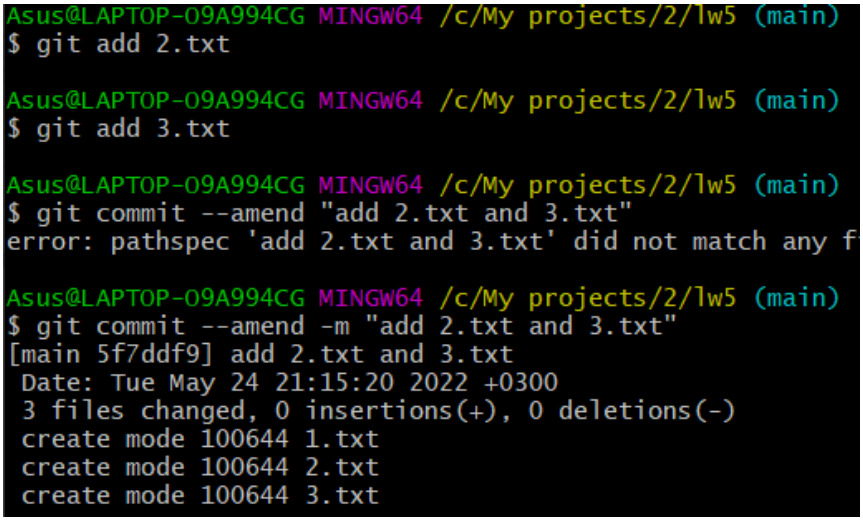

Рисунок 4 - Перезаписанный коммит

5. Создал новую ветку my_first_branch. Перешел на ветку и создал новый файл in_branch.txt, закоммитил изменения.

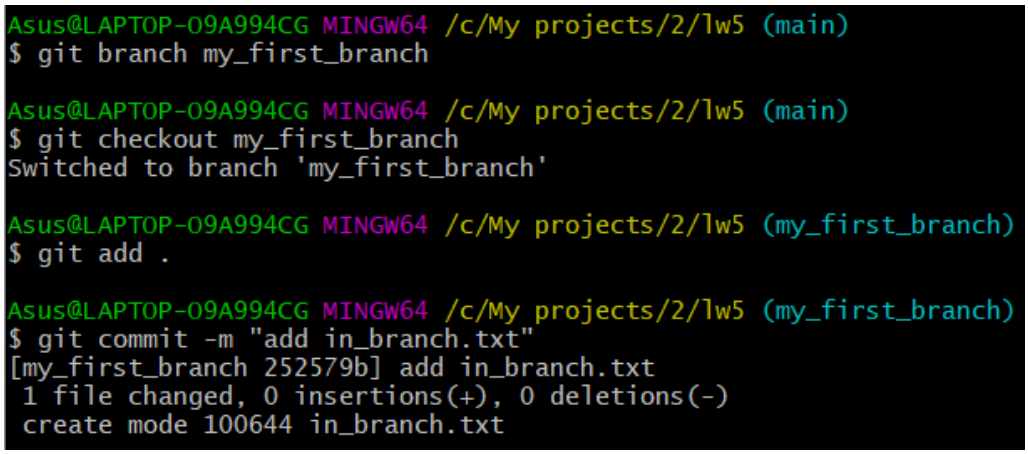

Рисунок 5 - Новая ветка my_first_branch

6. Вернулся на ветку master. Создал и сразу перешел на ветку new_branch.

```
Asus@LAPTOP-O9A994CG MINGW64 /c/My projects/2/lw5 (my_first_branch)
$ git checkout main
Switched to branch 'main'
Your branch is ahead of 'origin/main' by 1 commit.
  (use "git push" to publish your local commits)

Asus@LAPTOP-O9A994CG MINGW64 /c/My projects/2/lw5 (main)
$ git checkout -b new_branch
Switched to a new branch 'new_branch'
```

Рисунок 6 - Новая ветка new_branch

7. Сделал изменения в файле 1.txt, добавил строчку “new row in the 1.txt file”, закоммитил изменения.

```
Asus@LAPTOP-O9A994CG MINGW64 /c/My projects/2/lw5 (new_branch)
$ git add .

Asus@LAPTOP-O9A994CG MINGW64 /c/My projects/2/lw5 (new_branch)
$ git commit -m "new row in 1.txt"
[new_branch 11b1908] new row in 1.txt
 1 file changed, 1 insertion(+)
```

Рисунок 7 - Новая строчка в файле 1

8. Перешел на ветку master и слил ветки master и my_first_branch, после чего слил ветки master и new_branch.

```
Asus@LAPTOP-O9A994CG MINGW64 /c/My projects/2/lw5 (main|MERGING)
$ git merge my_first_branch
fatal: You have not concluded your merge (MERGE_HEAD exists).
Please, commit your changes before you merge.

Asus@LAPTOP-O9A994CG MINGW64 /c/My projects/2/lw5 (main|MERGING)
$ git merge new_branch
fatal: You have not concluded your merge (MERGE_HEAD exists).
Please, commit your changes before you merge.

Asus@LAPTOP-O9A994CG MINGW64 /c/My projects/2/lw5 (main|MERGING)
$ git add .

Asus@LAPTOP-O9A994CG MINGW64 /c/My projects/2/lw5 (main|MERGING)
$ git commit -m "merge with my_first_branch"
[main 2b38100] merge with my_first_branch
```

Рисунок 8 - Слияние веток

9. Удалил ветки my_first_branch и new_branch.

```
Asus@LAPTOP-O9A994CG MINGW64 /c/My projects/2/lw5 (main)
$ git branch -d my_first_branch
Deleted branch my_first_branch (was 252579b).

Asus@LAPTOP-O9A994CG MINGW64 /c/My projects/2/lw5 (main)
$ git branch -d new_branch
Deleted branch new_branch (was 11b1908).
```

Рисунок 9 - Удаление веток

10. Создал ветки branch_1 и branch_2.

```
Asus@LAPTOP-O9A994CG MINGW64 /c/My projects/2/lw5 (main)
$ git branch branch_1

Asus@LAPTOP-O9A994CG MINGW64 /c/My projects/2/lw5 (main)
$ git branch branch_2
```

Рисунок 10 - Создание новых веток

11. Перешел на ветку branch_1 и изменил файл 1.txt, удалил все содержимое и добавил текст “fix in the 1.txt”, изменил файл 3.txt, удалил все содержимое и добавил текст “fix in the 3.txt”, закоммитил изменения.

```
Asus@LAPTOP-O9A994CG MINGW64 /c/My projects/2/lw5 (main)
$ git checkout branch_1
Switched to branch 'branch_1'

Asus@LAPTOP-O9A994CG MINGW64 /c/My projects/2/lw5 (branch_1)
$ git add .

Asus@LAPTOP-O9A994CG MINGW64 /c/My projects/2/lw5 (branch_1)
$ git commit -m "fix 1"
[branch_1 a25903c] fix 1
 2 files changed, 2 insertions(+), 1 deletion(-)
```

Рисунок 11 - Изменение файлов на ветке branch_1

12. Перешел на ветку branch_2 и также изменил файл 1.txt, удалил все содержимое и добавио текст “My fix in the 1.txt”, изменил файл 3.txt, удалил все содержимое и добавил текст “My fix in the 3.txt”, закоммитил изменения.

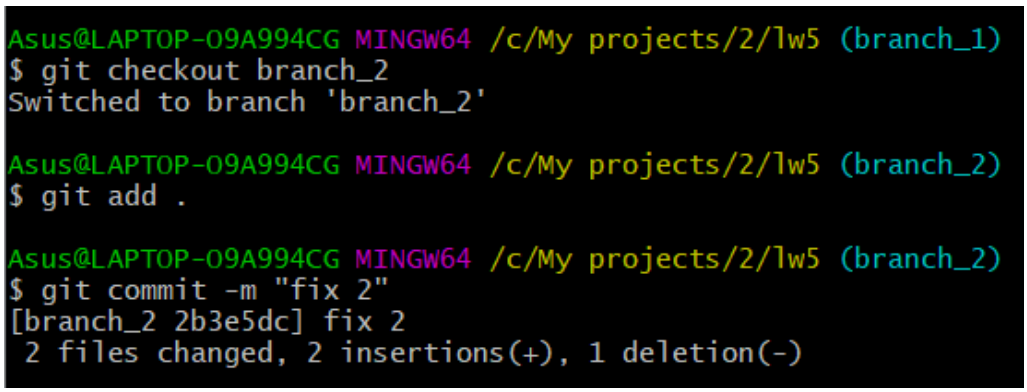

Рисунок 12 - Изменение файлов на ветке branch_2

13. Слил изменения ветки branch_2 в ветку branch_1.

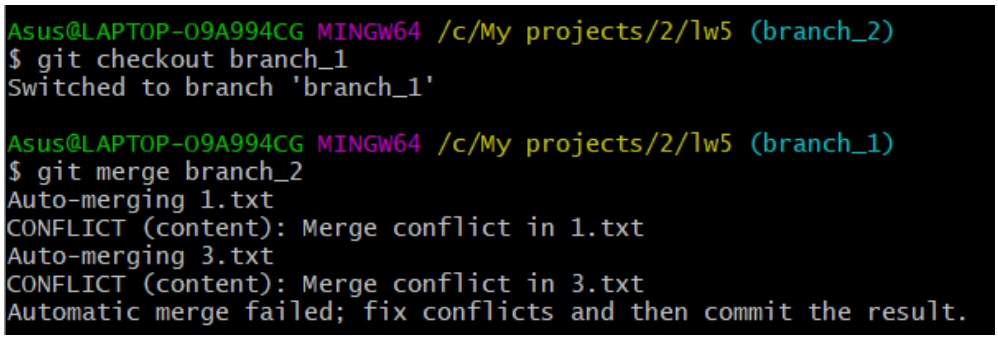

Рисунок 13 - Конфликт в слиянии веток

14. Решил конфликт файла 1.txt в ручном режиме, а конфликт 3.txt используя команду git mergetool с помощью одной из доступных утилиты vimdiff.

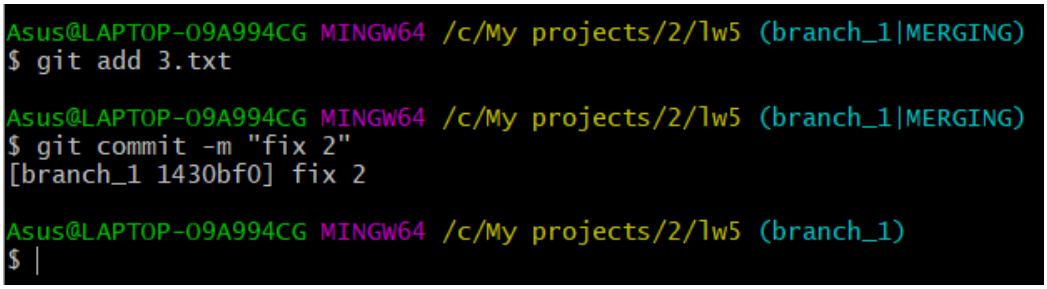

Рисунок 14 - Разрешение конфликта

15. Отправил ветку branch_1 на GitHub.

```
Asus@LAPTOP-O9A994CG MINGW64 /c/My projects/2/lw5 (branch_1)
$ git push origin branch_1
Enumerating objects: 23, done.
Counting objects: 100% (23/23), done.
Delta compression using up to 8 threads
Compressing objects: 100% (14/14), done.
Writing objects: 100% (22/22), 1.86 KiB | 950.00 KiB/s, done.
Total 22 (delta 6), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (6/6), done.
remote:
remote: Create a pull request for 'branch_1' on GitHub by visiting:
remote:      https://github.com/dshayderov/lw5/pull/new/branch_1
remote:
To https://github.com/dshayderov/lw5.git
 * [new branch]      branch_1 -> branch_1
```

Рисунок 15 - Отправление ветки branch_1

16. Создал средствами GitHub удаленную ветку branch_3.

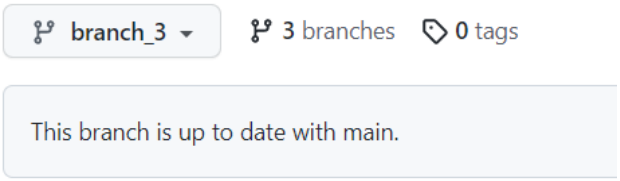

Рисунок 16 - Создание ветки branch_3

17. Создал в локальном репозитории ветку отслеживания удаленной ветки branch_3.

```
Asus@LAPTOP-O9A994CG MINGW64 /c/My projects/2/lw5 (branch_1)
$ git fetch origin
From https://github.com/dshayderov/lw5
 * [new branch]      branch_3   -> origin/branch_3

Asus@LAPTOP-O9A994CG MINGW64 /c/My projects/2/lw5 (branch_1)
$ git checkout -b branch_3 origin/branch_3
Switched to a new branch 'branch_3'
branch 'branch_3' set up to track 'origin/branch_3'.
```

Рисунок 17 - Создание ветки отслеживания

18. Перешел на ветку branch_2 и добавил в файл 2.txt строку "the final fantasy in the 4.txt file".

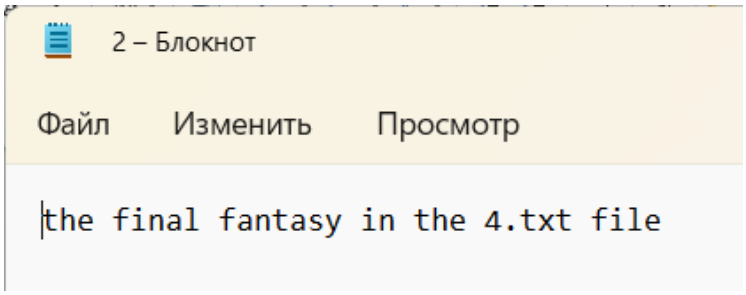

Рисунок 18 - Изменение в файле 2

19. Выполнил перемещение ветки master на ветку branch_2.

```
Asus@LAPTOP-O9A994CG MINGW64 /c/My projects/2/lw5 (branch_2)
$ git add .

Asus@LAPTOP-O9A994CG MINGW64 /c/My projects/2/lw5 (branch_2)
$ git commit -m "changes in 2.txt"
[branch_2 3c1e438] changes in 2.txt
 1 file changed, 1 insertion(+)

Asus@LAPTOP-O9A994CG MINGW64 /c/My projects/2/lw5 (branch_2)
$ git rebase main
Current branch branch_2 is up to date.
```

Рисунок 19 - Перемещение ветки

20. Отправил изменения веток master и branch_2 на GitHub.

```
Asus@LAPTOP-O9A994CG MINGW64 /c/My projects/2/lw5 (branch_2)
$ git push origin branch_2
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 286 bytes | 286.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
remote:
remote: Create a pull request for 'branch_2' on GitHub by visiting:
remote:      https://github.com/dshayderov/lw5/pull/new/branch_2
remote:
To https://github.com/dshayderov/lw5.git
 * [new branch]      branch_2 -> branch_2

Asus@LAPTOP-O9A994CG MINGW64 /c/My projects/2/lw5 (branch_2)
$ git checkout main
Switched to branch 'main'
Your branch is ahead of 'origin/main' by 4 commits.
  (use "git push" to publish your local commits)

Asus@LAPTOP-O9A994CG MINGW64 /c/My projects/2/lw5 (main)
$ git push origin main
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/dshayderov/lw5.git
   c6d6893..2b38100  main -> main
```

Рисунок 20 - Отправление изменений на GitHub

Контрольные вопросы:

1. Что такое ветка?

Ветка в Git — это простой перемещаемый указатель на один из коммитов.

2. Что такое HEAD?

HEAD – это указатель, задача которого ссылаться на определенный коммит в репозитории. Суть данного указателя можно попытаться объяснить с разных сторон.

Во-первых, HEAD – это указатель на коммит в вашем репозитории, который станет родителем следующего коммита.

Во-вторых, HEAD указывает на коммит, относительного которого будет создана рабочая копия во-время операции checkout .

3. Способы создания веток.

Новую ветку можно создать командой git branch <название_ветки> или git checkout –b <название_ветки> илина удаленном репозитории git hub.

4. Как узнать текущую ветку?

При помощи команды git branch.

5. Как переключаться между ветками?

При помощи команды git checkout .

6. Что такое удаленная ветка?

Это ветка, находящаяся на удаленном репозитории. Или ссылка на состояние ветки на удаленном репозитории.

7. Что такое ветка отслеживания?

Ветки слежения — это ссылки на определённое состояние удалённых веток. Это локальные ветки, которые нельзя перемещать; Git перемещает их автоматически при любой коммуникации с удаленным репозиторием, чтобы гарантировать точное соответствие с ним.

8. Как создать ветку отслеживания?

Командой git checkout --track origin/branch.

9. Как отправить изменения из локальной ветки в удаленную ветку?

Командой git push origin branch.

10. В чем отличие команд git fetch и git pull ?

Git pull – это сочетание команд git fetch (получение изменений с удаленного репозитория) и git merge (объединение веток).

11. Как удалить локальную и удаленную ветки?

Используя команду git branch -d branch. Для удаление удаленной ветки существует команда git push origin –d branch.

12. Изучить модель ветвления git-flow (использовать материалы статей https://www.atlassian.com/ru/git/tutorials/comparing-workflows/gitflow-workflow, https://habr.com/ru/post/106912/). Какие основные типы веток присуствуют в модели git-flow? Как организована работа с ветками в модели git-flow? В чем недостатки git-flow?

Центральный репозиторий содержит две главные ветки, существующие всё время.

- master
- develop

Мы используем следующие типы вспомогательных ветвей:

- Ветви функциональностей (Feature branches)
- Ветви релизов (Release branches)
- Ветви исправлений (Hotfix branches)

Последовательность действий при работе по модели Gitflow:

1. Из ветки main создается ветка develop.
2. Из ветки develop создается ветка release.
3. Из ветки develop создаются ветки feature.
4. Когда работа над веткой feature завершается, она сливается в ветку develop.
5. Когда работа над веткой release завершается, она сливается с ветками develop и main.
6. Если в ветке main обнаруживается проблема, из main создается ветка hotfix.
7. Когда работа над веткой hotfix завершается, она сливается с ветками develop и main.

Git-flow — это устаревшая версия рабочего процесса Git, в свое время ставшая принципиально новой стратегией управления ветками в Git. Популярность Git-flow стала снижаться под влиянием магистральных рабочих процессов, которые на сегодня считаются предпочтительными для современных схем непрерывной разработки ПО и применения DevOps.

Кроме того, Git-flow не слишком удобно применять в процессах CI/CD. В этой публикации приводится описание Git-flow для истории.

Вывод: исследовал базовых возможностей по работе с локальными и удаленными ветками Git.