

Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт цифрового развития
Кафедра инфокоммуникаций

«Основы ветвления Git»

ОТЧЕТ
по лабораторной работе №3
дисциплины
«Основы программной инженерии»

Выполнил:
Борсуков Владислав Олегович
2 курс, группа ПИЖ-б-о-21-1,
09.03.04 «Программная
инженерия»

—
(подпись)

Проверил:

—
(подпись)

Отчет защищен с оценкой _____ Дата защиты _____

Ставрополь, 2022 г.

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

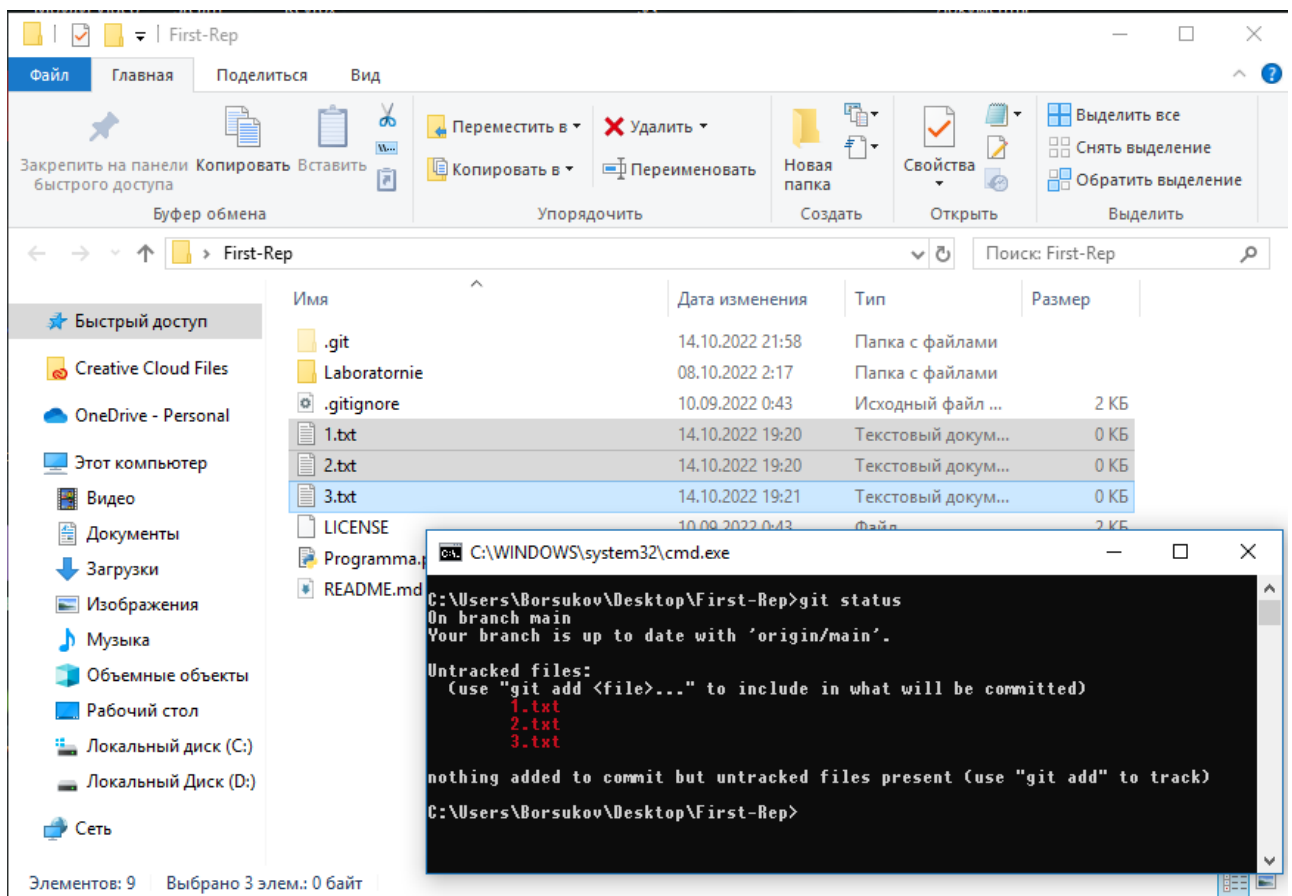


Рисунок 1 – Создание файлов

2. Проиндексировать первый файл и сделать коммит с комментарием "add 1.txt file".

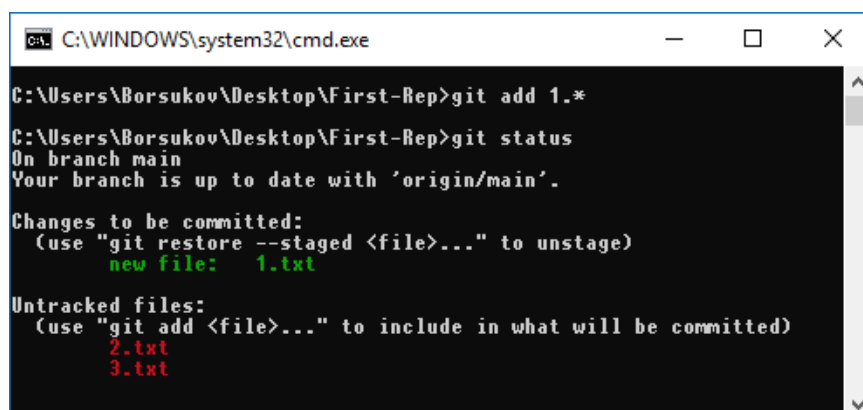
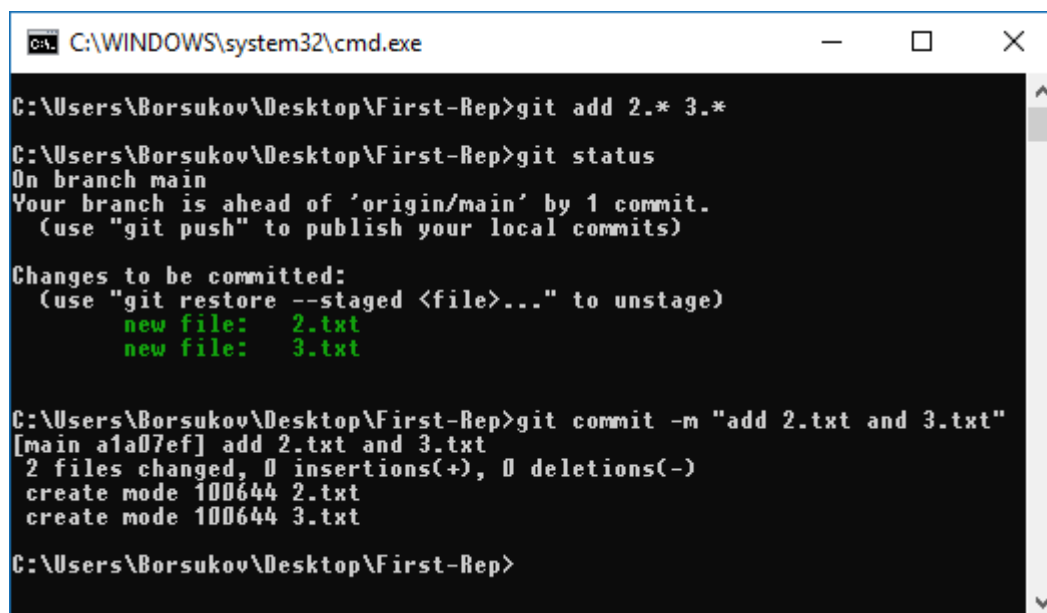


Рисунок 2 – Индексация первого файла

3. Проиндексировать второй и третий файлы, перезаписать уже сделанный коммит с новым комментарием "add 2.txt and 3.txt."



```
C:\WINDOWS\system32\cmd.exe

C:\Users\Borsukov\Desktop\First-Rep>git add 2.* 3.*

C:\Users\Borsukov\Desktop\First-Rep>git status
On branch main
Your branch is ahead of 'origin/main' by 1 commit.
(use "git push" to publish your local commits)

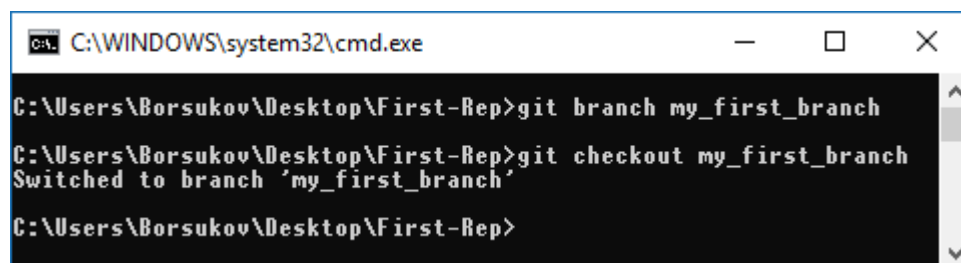
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   2.txt
        new file:   3.txt

C:\Users\Borsukov\Desktop\First-Rep>git commit -m "add 2.txt and 3.txt"
[main a1a07ef] add 2.txt and 3.txt
 2 files changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 2.txt
 create mode 100644 3.txt

C:\Users\Borsukov\Desktop\First-Rep>
```

Рисунок 3 – Индексация второго и третьего файла

4. Создать новую ветку my_first_branch



```
C:\WINDOWS\system32\cmd.exe

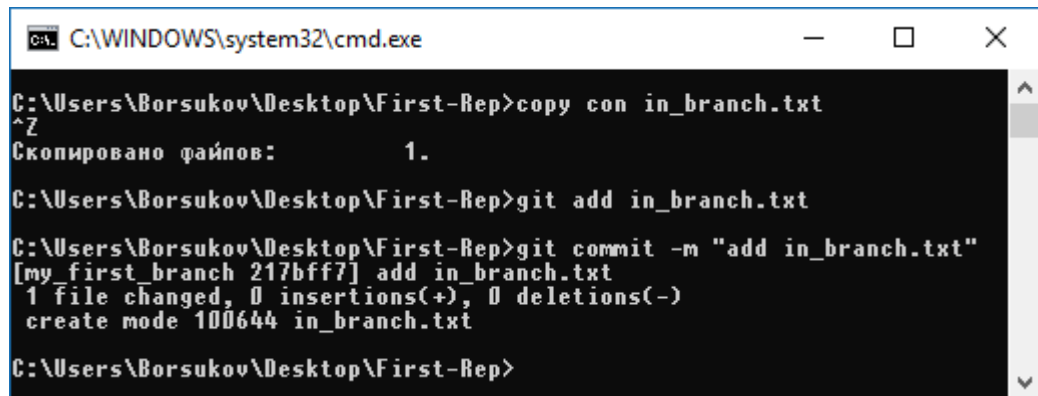
C:\Users\Borsukov\Desktop\First-Rep>git branch my_first_branch

C:\Users\Borsukov\Desktop\First-Rep>git checkout my_first_branch
Switched to branch 'my_first_branch'

C:\Users\Borsukov\Desktop\First-Rep>
```

Рисунок 4 – Создание новой ветки «my_first_branch»

5. Перейти на ветку и создать новый файл in_branch.txt, закоммитить изменения.



```
C:\WINDOWS\system32\cmd.exe

C:\Users\Borsukov\Desktop\First-Rep>copy con in_branch.txt
^Z
Скопировано файлов:      1.

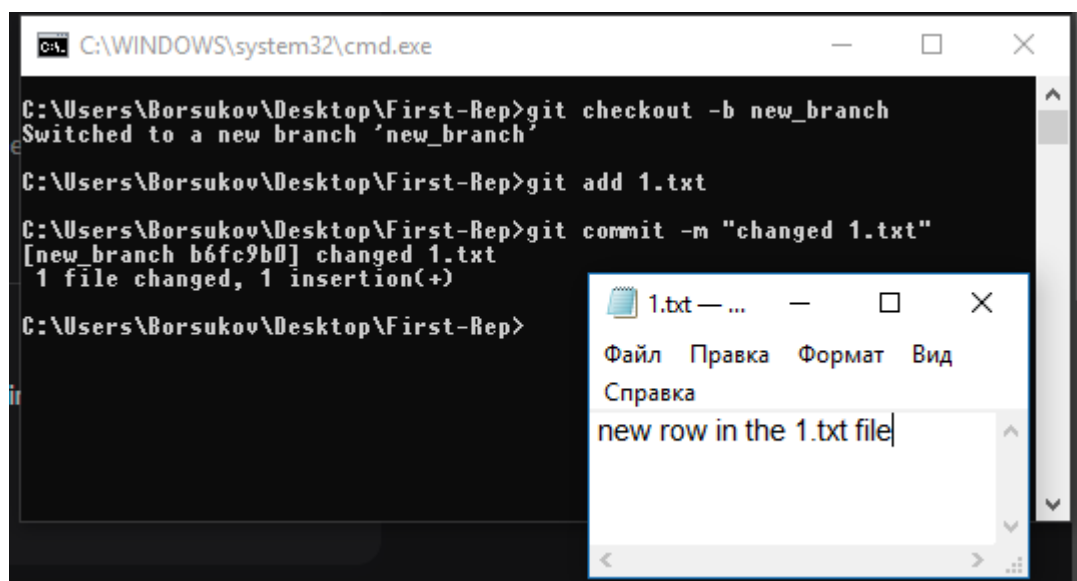
C:\Users\Borsukov\Desktop\First-Rep>git add in_branch.txt

C:\Users\Borsukov\Desktop\First-Rep>git commit -m "add in_branch.txt"
[my_first_branch 217bff7] add in_branch.txt
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 in_branch.txt

C:\Users\Borsukov\Desktop\First-Rep>
```

Рисунок 5 – Создание, индексация и коммит файла

6. Вернуться на ветку master (main), создать и сразу перейти на ветку new_branch, сделать изменения в файле 1.txt, добавить строчку “new row in the 1.txt file”, закоммитить изменения.



```
C:\WINDOWS\system32\cmd.exe

C:\Users\Borsukov\Desktop\First-Rep>git checkout -b new_branch
Switched to a new branch 'new_branch'

C:\Users\Borsukov\Desktop\First-Rep>git add 1.txt

C:\Users\Borsukov\Desktop\First-Rep>git commit -m "changed 1.txt"
[new_branch b6fc9b0] changed 1.txt
1 file changed, 1 insertion(+)

C:\Users\Borsukov\Desktop\First-Rep>
```

1.txt — ...

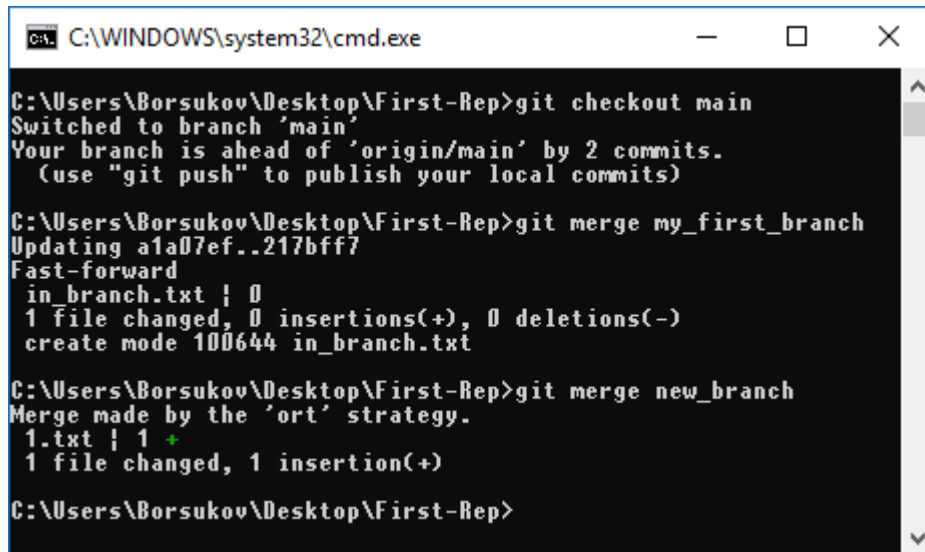
Файл Правка Формат Вид

Справка

new row in the 1.txt file

Рисунок 6 – Возврат к ветке master (main) создание новой ветки с переходом на неё и добавление изменений в файл с последующим коммитом

7. Перейти на ветку master (main) и слить ветки master (main) и my_first_branch, после чего слить ветки master (main) и new_branch.



```
C:\WINDOWS\system32\cmd.exe

C:\Users\Borsukov\Desktop\First-Rep>git checkout main
Switched to branch 'main'
Your branch is ahead of 'origin/main' by 2 commits.
(use "git push" to publish your local commits)

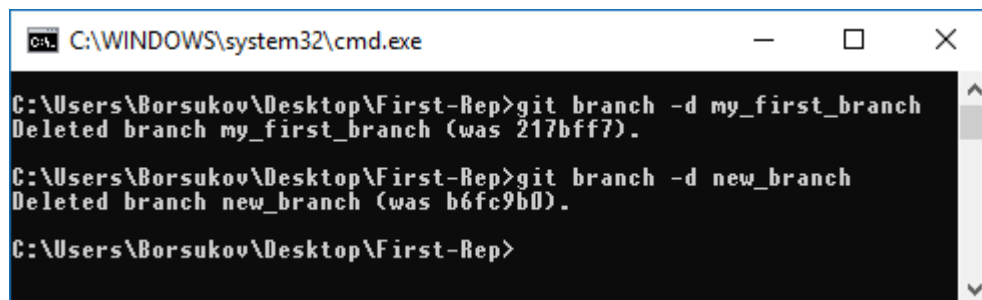
C:\Users\Borsukov\Desktop\First-Rep>git merge my_first_branch
Updating a1a07ef..217bff7
Fast-forward
 in_branch.txt | 0
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 in_branch.txt

C:\Users\Borsukov\Desktop\First-Rep>git merge new_branch
Merge made by the 'ort' strategy.
 1.txt | 1 +
 1 file changed, 1 insertion(+)

C:\Users\Borsukov\Desktop\First-Rep>
```

Рисунок 7 – Слияние веток

8. Удалить ветки my_first_branch и new_branch.



```
C:\WINDOWS\system32\cmd.exe

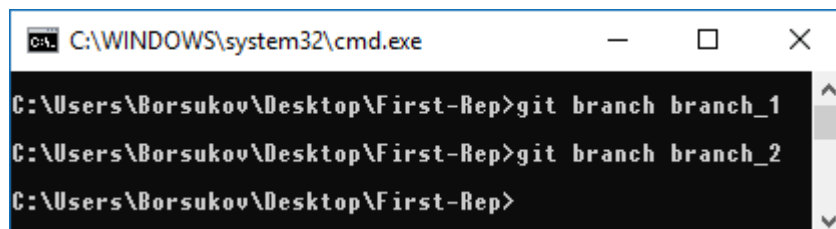
C:\Users\Borsukov\Desktop\First-Rep>git branch -d my_first_branch
Deleted branch my_first_branch (was 217bff7).

C:\Users\Borsukov\Desktop\First-Rep>git branch -d new_branch
Deleted branch new_branch (was b6fc9b0).

C:\Users\Borsukov\Desktop\First-Rep>
```

Рисунок 8 – Удаление веток

9. Создать ветки branch_1 и branch_2



```
C:\WINDOWS\system32\cmd.exe

C:\Users\Borsukov\Desktop\First-Rep>git branch branch_1
C:\Users\Borsukov\Desktop\First-Rep>git branch branch_2
C:\Users\Borsukov\Desktop\First-Rep>
```

Рисунок 9 – Создание веток

10. Перейти на ветку `branch_1` и изменить файл `1.txt`, удалить все содержимое и добавить текст “fix in the 1.txt”, изменить файл `3.txt`, удалить все содержимое и добавить текст “fix in the 3.txt”, закоммитить изменения.

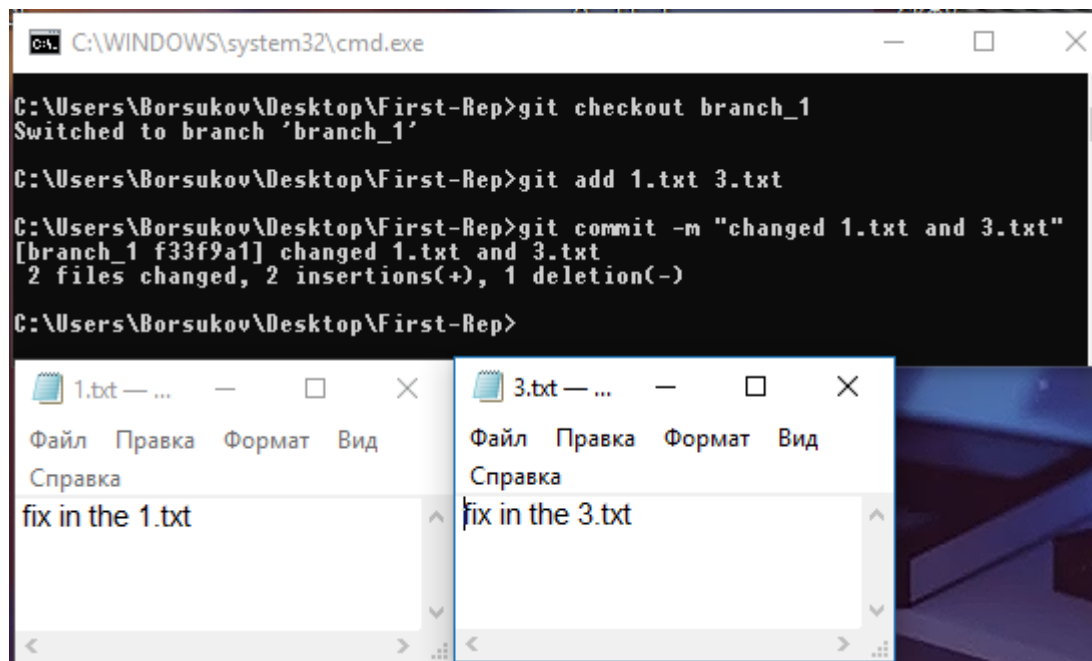


Рисунок 10 – Переход на ветку `branch_1`, изменение файлов

11. Перейти на ветку `branch_2` и также изменить файл `1.txt`, удалить все содержимое и добавить текст “My fix in the 1.txt”, изменить файл `3.txt`, удалить все содержимое и добавить текст “My fix in the 3.txt”, закоммитить изменения.

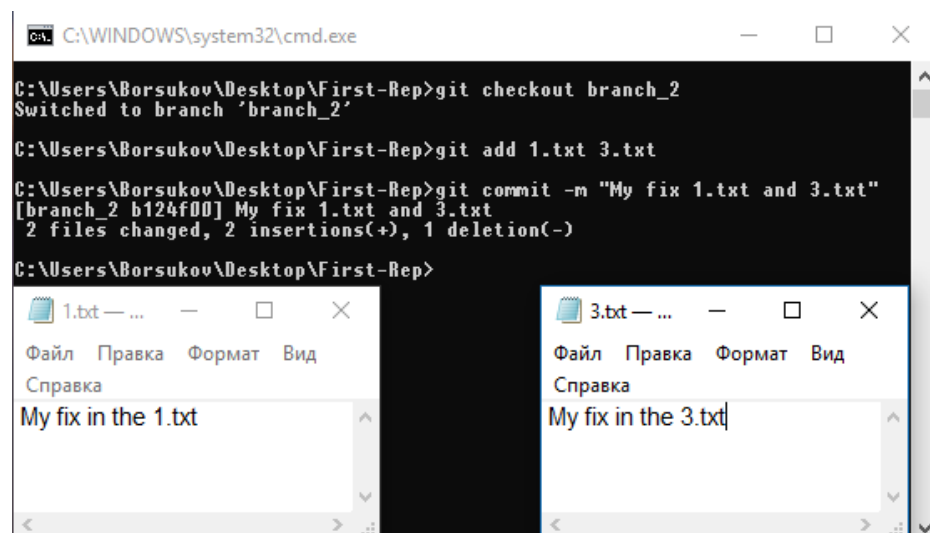
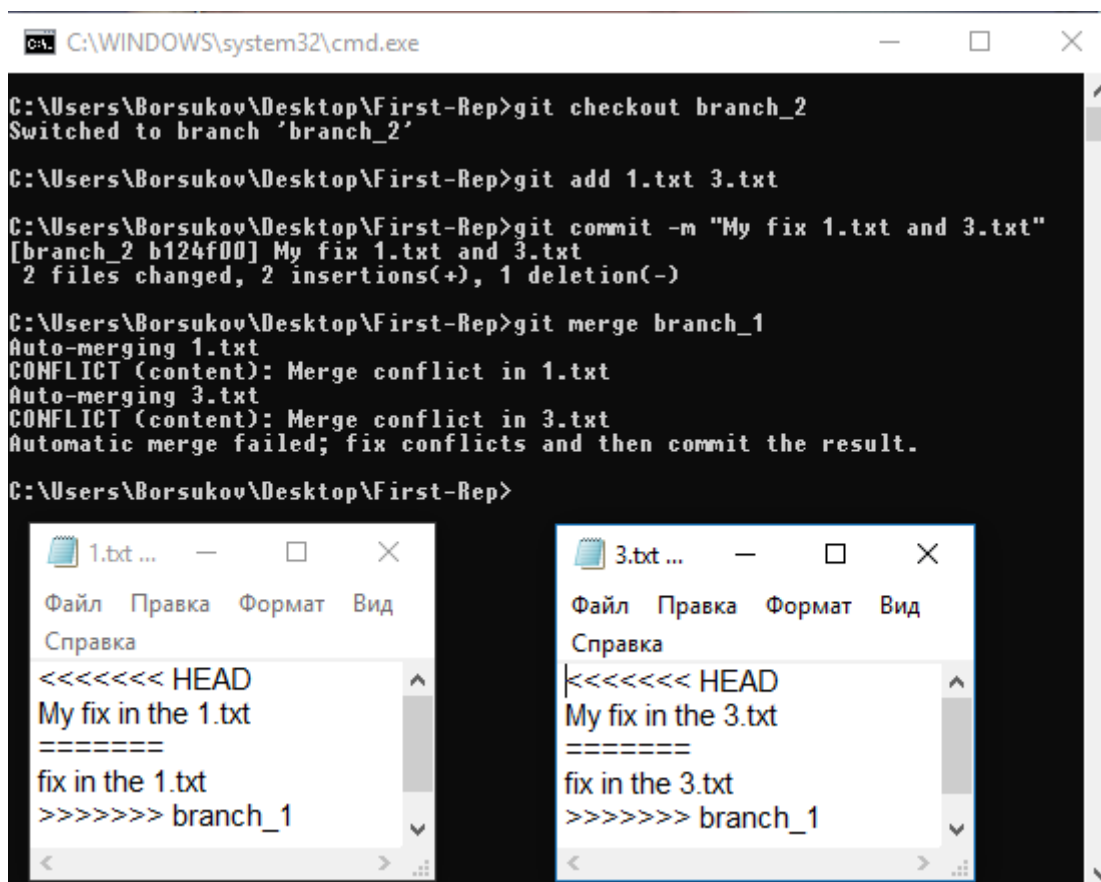


Рисунок 11 – Переход на ветку `branch_2` и изменение файлов

12. Слить изменения ветки branch_2 в ветку branch_1.



The screenshot shows a Windows command prompt window titled "C:\WINDOWS\system32\cmd.exe" with the following commands and output:

```
C:\Users\Borsukov\Desktop\First-Rep>git checkout branch_2
Switched to branch 'branch_2'

C:\Users\Borsukov\Desktop\First-Rep>git add 1.txt 3.txt

C:\Users\Borsukov\Desktop\First-Rep>git commit -m "My fix 1.txt and 3.txt"
[branch_2 b124f00] My fix 1.txt and 3.txt
2 files changed, 2 insertions(+), 1 deletion(-)

C:\Users\Borsukov\Desktop\First-Rep>git merge branch_1
Auto-merging 1.txt
CONFLICT (content): Merge conflict in 1.txt
Auto-merging 3.txt
CONFLICT (content): Merge conflict in 3.txt
Automatic merge failed; fix conflicts and then commit the result.

C:\Users\Borsukov\Desktop\First-Rep>
```

Below the command prompt, two text editor windows are shown side-by-side. The left window is titled "1.txt ..." and the right window is titled "3.txt ...". Both windows show a merge conflict with the following content:

```
<<<<<<< HEAD
My fix in the 1.txt
=====
fix in the 1.txt
>>>>>>> branch_1
```

Рисунок 12 – Результат попытки слияния веток

13. Решить конфликт файла 1.txt в ручном режиме, а конфликт 3.txt используя команду git mergetool с помощью одной из доступных утилит, например Meld.

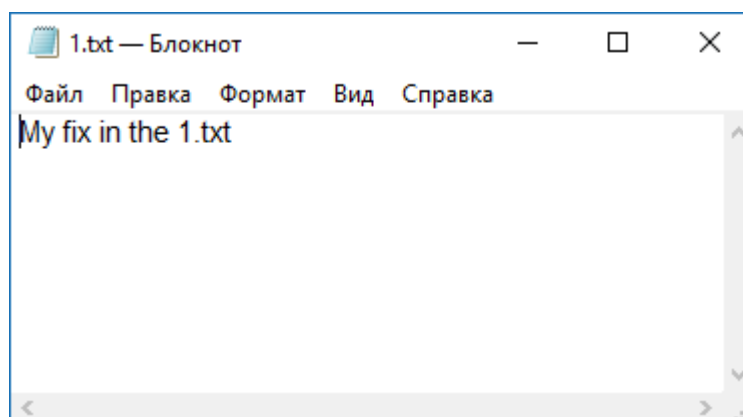


Рисунок 13.1 – Решение конфликта файлов вручную

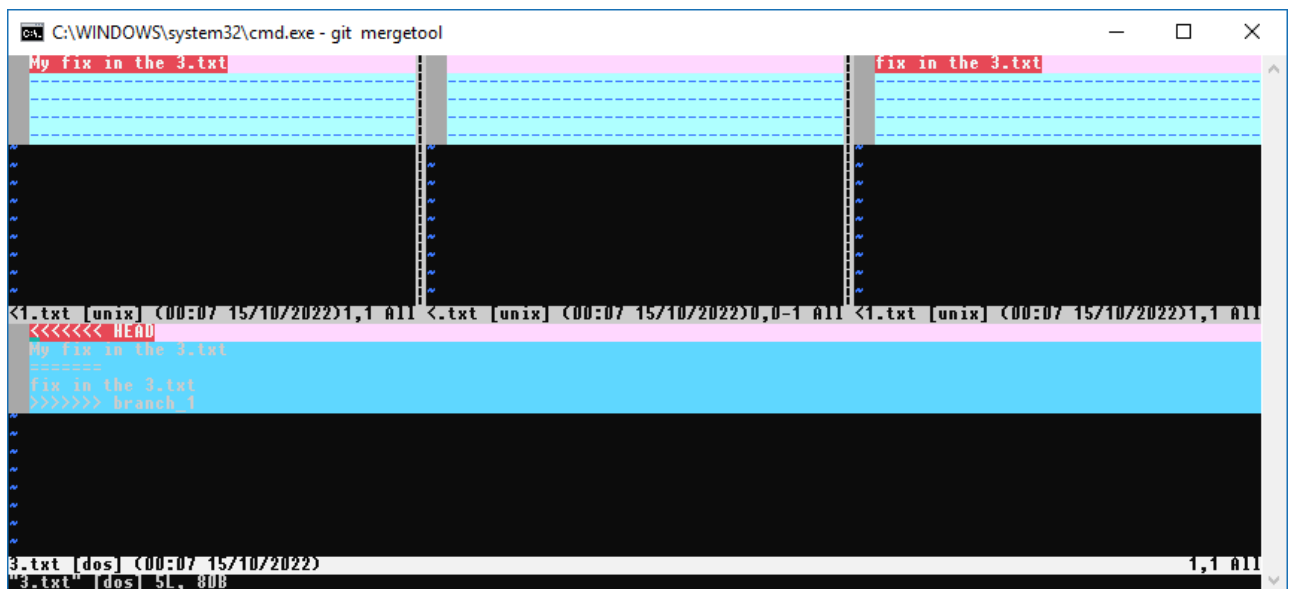


Рисунок 13.2 – Решение конфликта файлов через git mergetool с использованием vimdiff

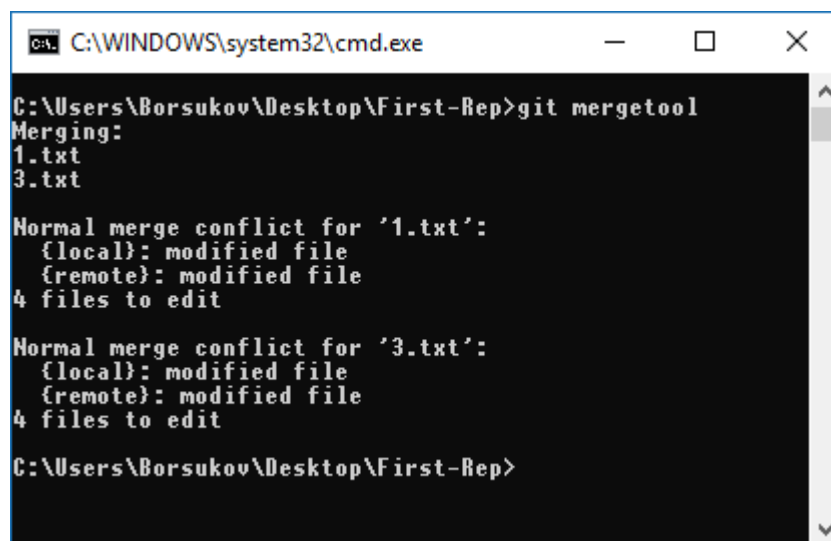
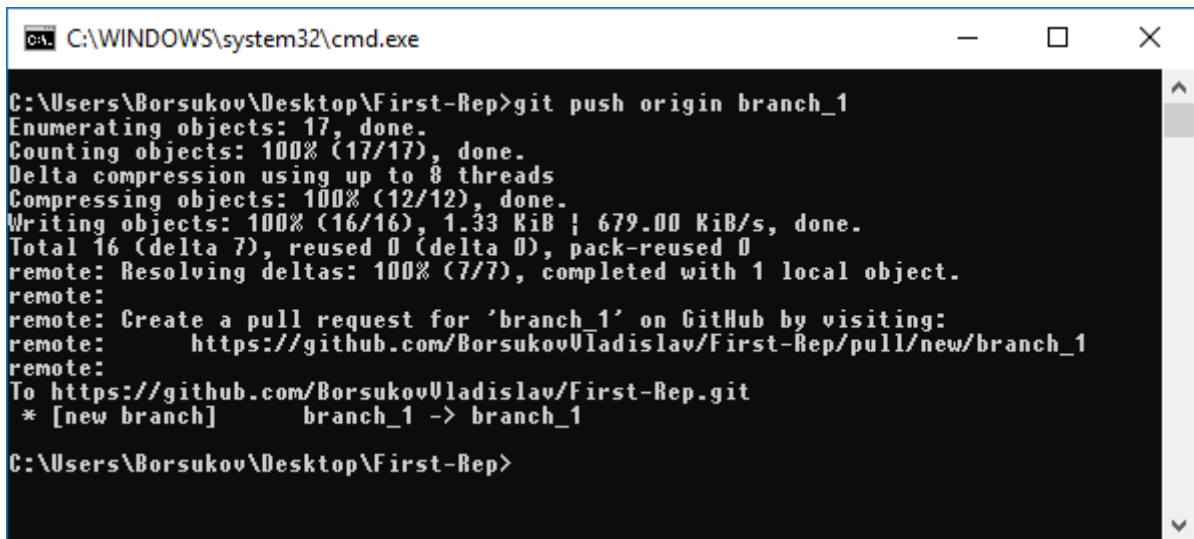


Рисунок 14 – Результат работы git mergetool

14. Отправить ветку branch_1 на GitHub.



```
C:\WINDOWS\system32\cmd.exe

C:\Users\Borsukov\Desktop\First-Rep>git push origin branch_1
Enumerating objects: 17, done.
Counting objects: 100% (17/17), done.
Delta compression using up to 8 threads
Compressing objects: 100% (12/12), done.
Writing objects: 100% (16/16), 1.33 KiB | 679.00 KiB/s, done.
Total 16 (delta 7), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (7/7), completed with 1 local object.
remote:
remote: Create a pull request for 'branch_1' on GitHub by visiting:
remote:   https://github.com/BorsukovVladislav/First-Rep/pull/new/branch_1
remote:
To https://github.com/BorsukovVladislav/First-Rep.git
 * [new branch]      branch_1 -> branch_1

C:\Users\Borsukov\Desktop\First-Rep>
```

Рисунок 15 – Отправка ветки на GitHub

15. Создать средствами GitHub удаленную ветку branch_3.

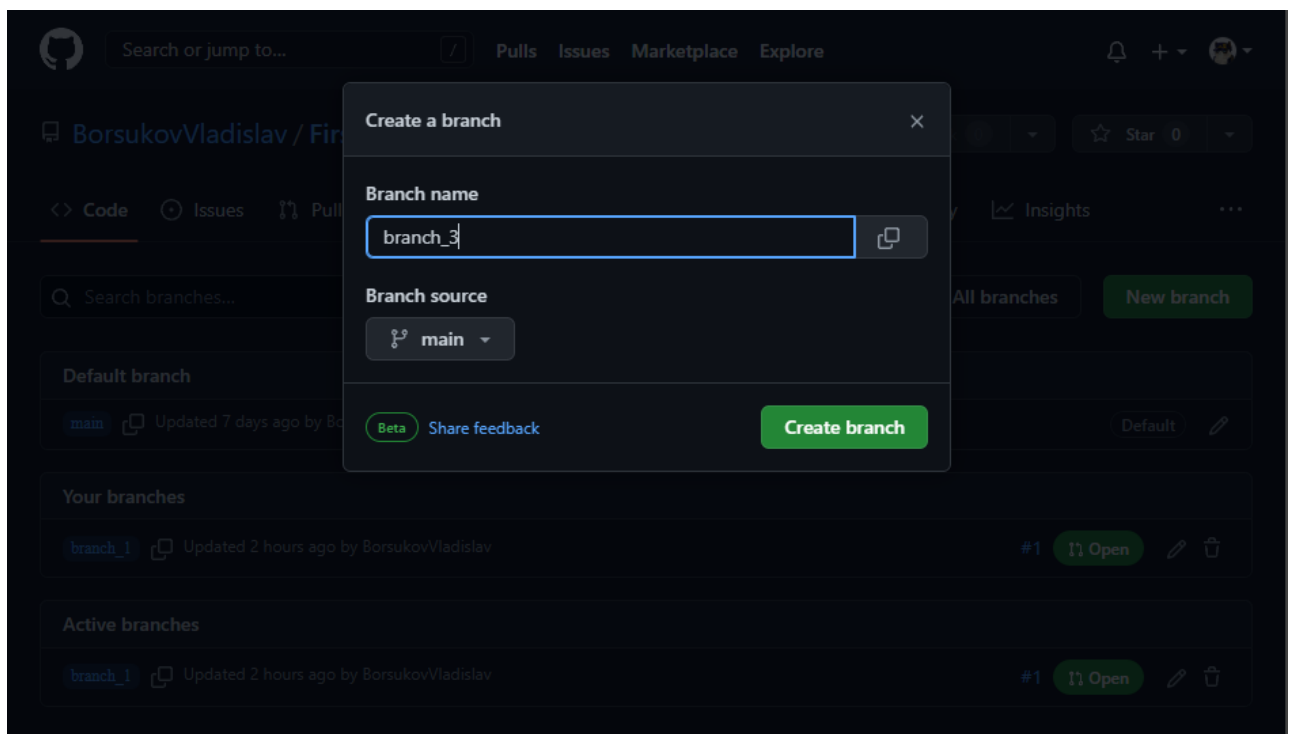


Рисунок 16 – Создание удалённой ветки через GitHub

16. Создать в локальном репозитории ветку отслеживания удаленной ветки branch_3.

```
C:\Users\Borsukov\Desktop\First-Rep>git checkout --track 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

17. Перейти на ветку branch_3 и добавить в файл 2.txt строку "the final fantasy in the 4.txt file".

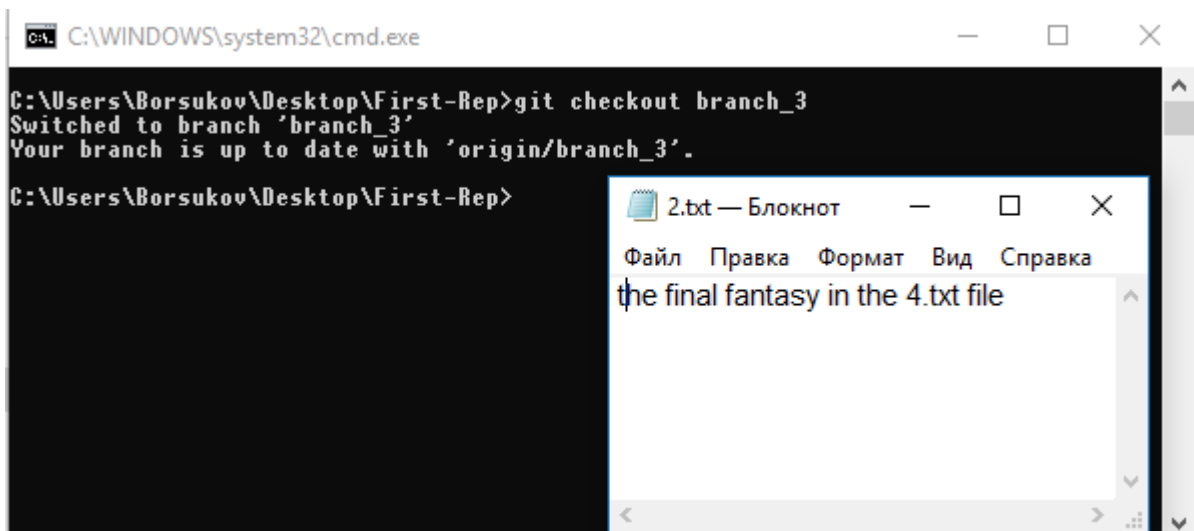
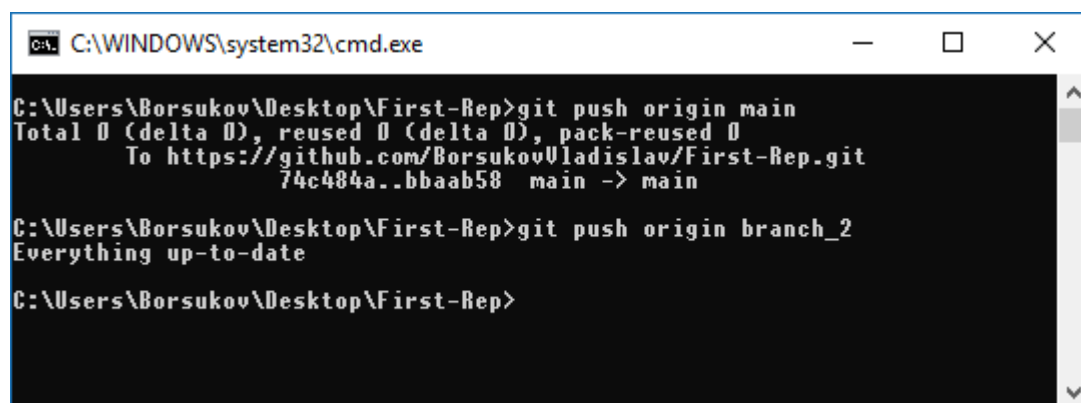


Рисунок 18 – Переход на ветку и добавление файла

18. Отправить изменения веток master и branch_2 на GitHub.



```
C:\WINDOWS\system32\cmd.exe

C:\Users\Borsukov\Desktop\First-Rep>git push origin main
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/BorsukovUladislav/First-Rep.git
74c484a..bbaab58  main -> main

C:\Users\Borsukov\Desktop\First-Rep>git push origin branch_2
Everything up-to-date

C:\Users\Borsukov\Desktop\First-Rep>
```

Рисунок 19 – отправление изменения веток на GitHub

1. Что такое ветка?

Ветка в Git — это просто легковесный подвижный указатель на один из коммитов.

2. Что такое HEAD?

HEAD – это указатель, задача которого ссылаться на определенный коммит в репозитории.

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

3. Способы создания веток.

Командой `git branch`

Командой `git checkout -b`

4. Как узнать текущую ветку?

Командой `git branch`

5. Как переключаться между ветками?

`Git checkout «имя ветки»`

6. Что такое удаленная ветка?

Удалённые ветки — это ссылки на состояние веток в удаленных репозиториях.

7. Что такое ветка отслеживания?

Ветки слежения — это ссылки на определённое состояние удалённых веток.

8. Как создать ветку отслеживания?

Для синхронизации ваших изменений с удаленным сервером выполним команду `git fetch origin`. Далее прописать `git checkout --track origin/ «имя ветки»`

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

Командой `git push «имя ветки»`

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

`Git pull` — это, по сути, команда `git fetch`, после которой сразу же следует `git merge`. `git fetch` получает изменения с сервера и сохраняет их в каталог. А `git merge` уже вливает все эти изменения в локальную копию.

11. Как удалить локальную и удаленную ветки?

Удаление удаленной ветки производится при помощи команды: `git push origin --delete «имя ветки»`

Удаление локальной ветки производится при помощи команды: `git branch -d «имя ветки»`

12. Изучить модель ветвления `git-flow` (использовать материалы статей

[https://www.atlassian.c](https://www.atlassian.com/ru/git/tutorials/comparing-workflows/gitflow-workflow)

[om/ru/git/tutorials/comparing-workflows/gitflow-workflow](https://www.atlassian.com/ru/git/tutorials/comparing-workflows/gitflow-workflow),

<https://habr.com/ru/post/106912/>). Какие основные типы веток присутствуют в модели `git-flow`? Как организована работа с ветками в модели `git-flow`? В чем недостатки `git-flow`?

`Git-flow` — альтернативная модель ветвления `Git`, в которой используются функциональные ветки и несколько основных веток.

Под каждую новую функцию нужно выделить собственную ветку, которую можно отправить в центральный репозиторий для создания резервной копии или совместной работы команды. Ветки `feature` создаются не

на основе main, а на основе develop. Когда работа над функцией завершается, соответствующая ветка сливается с веткой develop. Функции не следует отправлять напрямую в ветку main.

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.

Первая проблема: авторам приходится использовать ветку develop вместо master, поскольку master зарезервирован для кода, который отправляется в продакшен. Существует сложившийся обычай называть рабочую ветвь по умолчанию master, и делать ответвления и слияния с ней. Большинство инструментов по умолчанию используют это название для основной ветки и по умолчанию выводят именно ее, и бывает неудобно постоянно переключаться вручную на другую ветку.

Вторая проблема процесса git flow – сложности, возникающие из-за веток для патчей и для релиза. Подобная структура может подойти некоторым организациям, но для абсолютного большинства она просто убийственно излишня. На сегодняшний день большинство компаний практикуют непрерывное развертывание (continuous delivery), что подразумевает, что основная ветвь по умолчанию может быть задеплоена (deploy). А значит, можно избежать использования веток для релиза и патчей, и всех связанных с ними хлопот, например, обратного слияния из веток релизов.

13. На прошлой лабораторной работе было задание выбрать одно из программных средств с GUI для работы с Git. Необходимо в рамках этого вопроса привести описание инструментов для работы с ветками Git, предоставляемых этим средством.

 Current branch
main

 **Fetch origin**
Last fetched 1 hour a

Branches

Pull requests


New branch

Default branch


✓ main

3 hours ago

Recent branches


 branch_2

3 hours ago

 branch_3

1 hour ago


Other branches

 origin/branch_2

2 hours ago

Merge into **branch_2** ×

Default branch


 main

3 hours ago

Recent branches


✓ branch_2

3 hours ago

 branch_3

1 hour ago

Other branches

 origin/branch_2

2 hours ago

✓

This branch is up to date with **main**

Create a merge commit ▼