

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:

Мизин Глеб Егорович

2 курс, группа ПИЖ-б-о-21-1,

09.03.04 «Программная

инженерия», направленность

(профиль) «Разработка и

сопровождение программного

обеспечения», очная форма

обучения

—

(подпись)

Проверил:

—

(подпись)

Отчет защищен с оценкой _____ Дата защиты _____

Ставрополь, 2022 г.

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

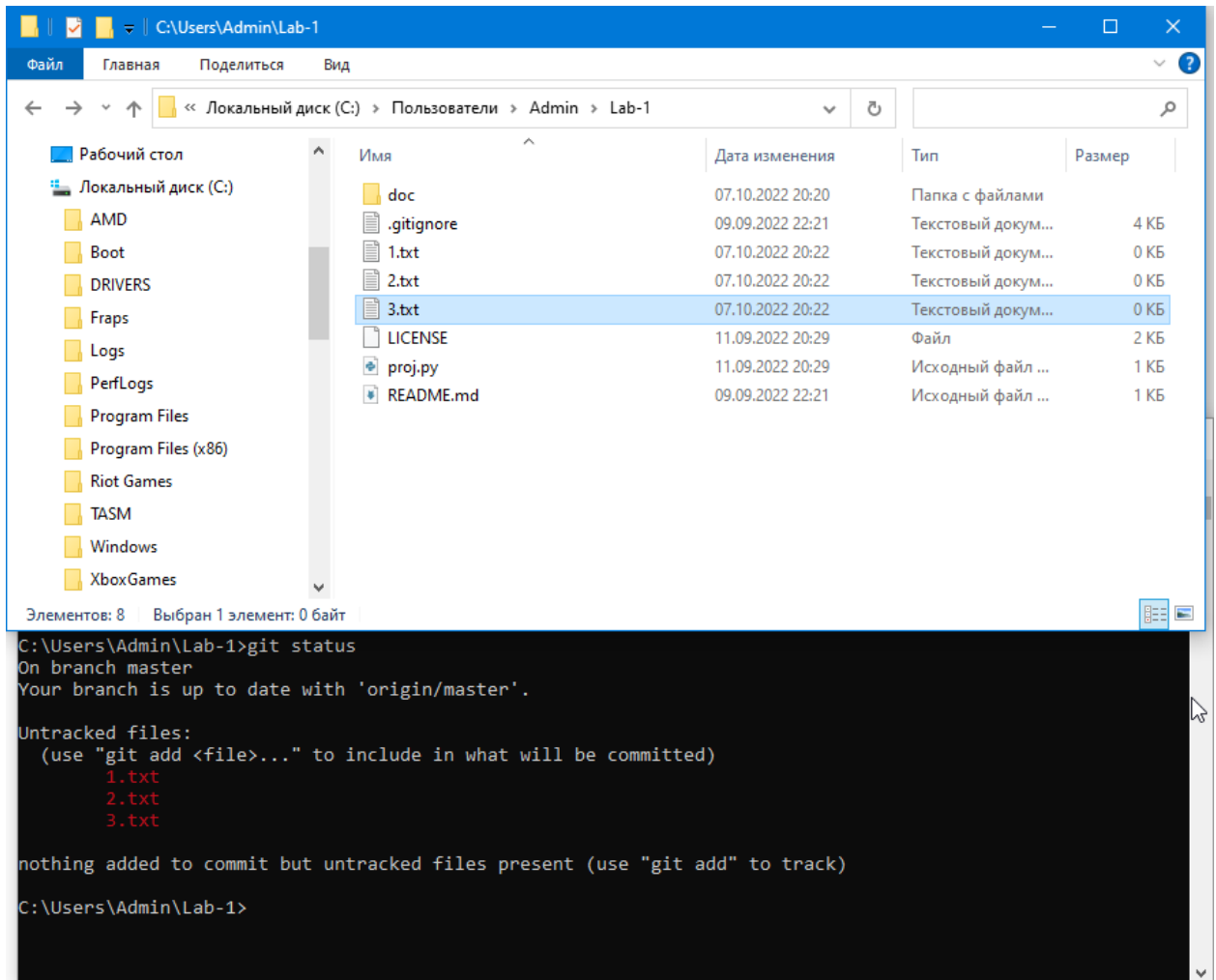


Рисунок 1 – создание файлов

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

```

C:\Users\Admin\Lab-1>git status
On branch master
Your branch is up to date with 'origin/master'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    new file:   1.txt

Untracked files:
  (use "git add <file>..." to include in what will be committed)
    2.txt
    3.txt

C:\Users\Admin\Lab-1>git commit -m "add 1.txt file"
[master df6391c] add 1.txt file
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 1.txt

```

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:\Users\Admin\Lab-1>git add 2.txt 3.txt

C:\Users\Admin\Lab-1>git status
On branch master
Your branch is ahead of 'origin/master' 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\Admin\Lab-1>git commit -m "add 2.txt and 3.txt"
[master 4025d49] add 2.txt and 3.txt
2 files changed, 0 insertions(+), 0 deletions(-)
create mode 100644 2.txt
create mode 100644 3.txt

```

Рисунок 3 – индексация второго и третьего файла

4. Создать новую ветку my_first_branch

```

C:\Users\Admin\Lab-1>git branch my_first_branch

C:\Users\Admin\Lab-1>git checkout my_first_branch
Switched to branch 'my_first_branch'

```

Рисунок 4 – создание новой ветки «my_first_branch»

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

```
C:\Users\Admin\Lab-1>copy con in_branch.txt
Скопировано файлов:      1.

C:\Users\Admin\Lab-1>git add in_branch.txt

C:\Users\Admin\Lab-1>git commit -m "add in_branch.txt to my_first_branch"
[my_first_branch bea8523] add in_branch.txt to my_first_branch
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 in_branch.txt
```

Рисунок 5 – создание, индексация и коммит файла

6. Вернуться на ветку master, 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:\Users\Admin\Lab-1>git checkout master
Switched to branch 'master'
Your branch is ahead of 'origin/master' by 2 commits.
(use "git push" to publish your local commits)

C:\Users\Admin\Lab-1>git checkout -b new_branch
Switched to a new branch 'new_branch'

C:\Users\Admin\Lab-1>git add 1.txt

C:\Users\Admin\Lab-1>git commit -m "Изменил 1.txt"
[new_branch 89e9db5] Изменил 1.txt
1 file changed, 1 insertion(+)
```

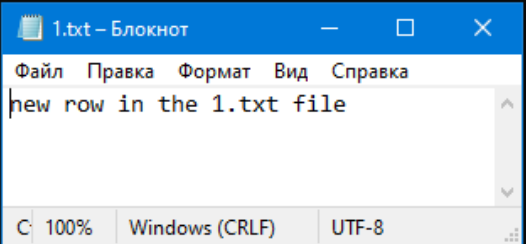


Рисунок 6 – возврат к ветке master создание новой ветки с переходом на неё и добавление изменений в файл с последующим коммитом

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

```

C:\Users\Admin\Lab-1>git checkout master
Switched to branch 'master'
Your branch is ahead of 'origin/master' by 2 commits.
  (use "git push" to publish your local commits)

C:\Users\Admin\Lab-1>git merge my_first_branch
Updating 4025d49..bea8523
Fast-forward
 in_branch.txt | 0
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 in_branch.txt

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

```

Рисунок 7 – слияние веток

8. Удалить ветки my_first_branch и new_branch.

```

C:\Users\Admin\Lab-1>git branch -d my_first_branch
Deleted branch my_first_branch (was bea8523).

C:\Users\Admin\Lab-1>git branch -d new_branch
Deleted branch new_branch (was 89e9db5).

```

Рисунок 8 – удаление веток

9. Создать ветки branch_1 и branch_2

```

C:\Users\Admin\Lab-1>git branch branch_1
C:\Users\Admin\Lab-1>git branch branch_2

```

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.

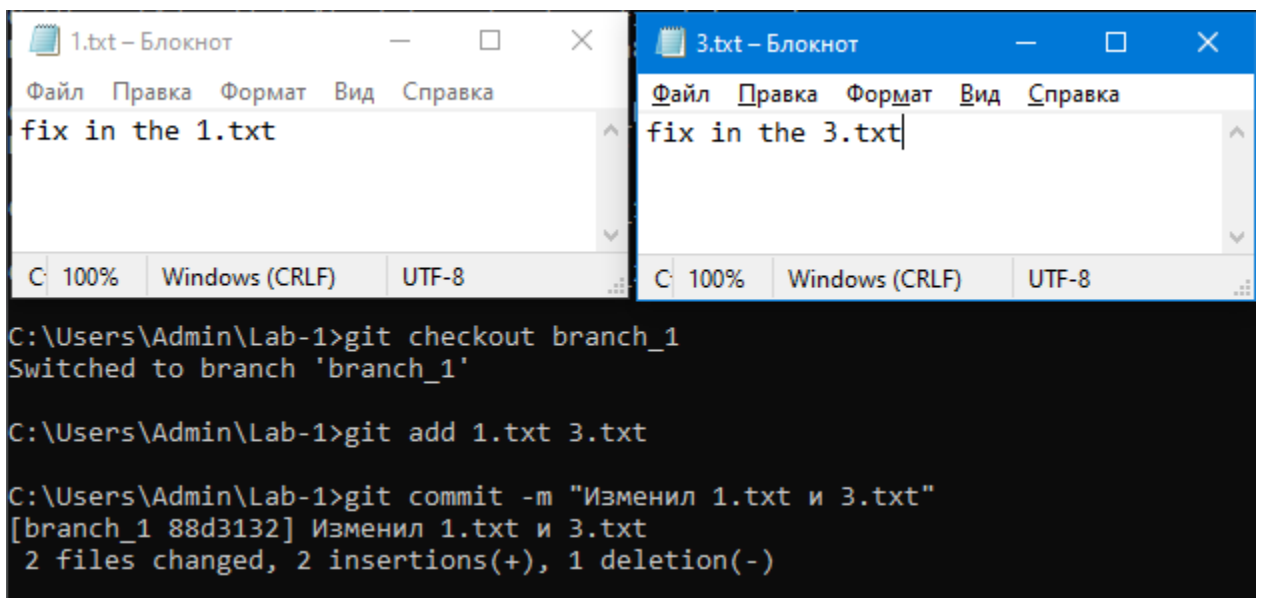


Рисунок 10 – переход на ветку branch_1, изменение файлов

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.

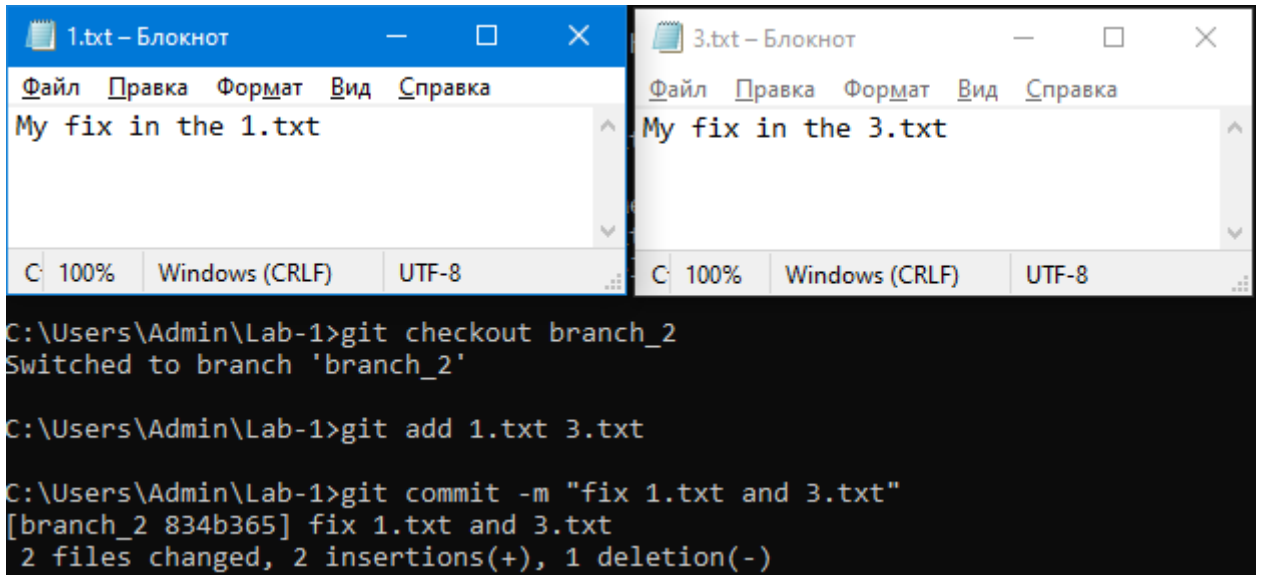


Рисунок 11 – Переход на ветку branch_2 и изменение файлов

12. Слить изменения ветки branch_2 в ветку branch_1.

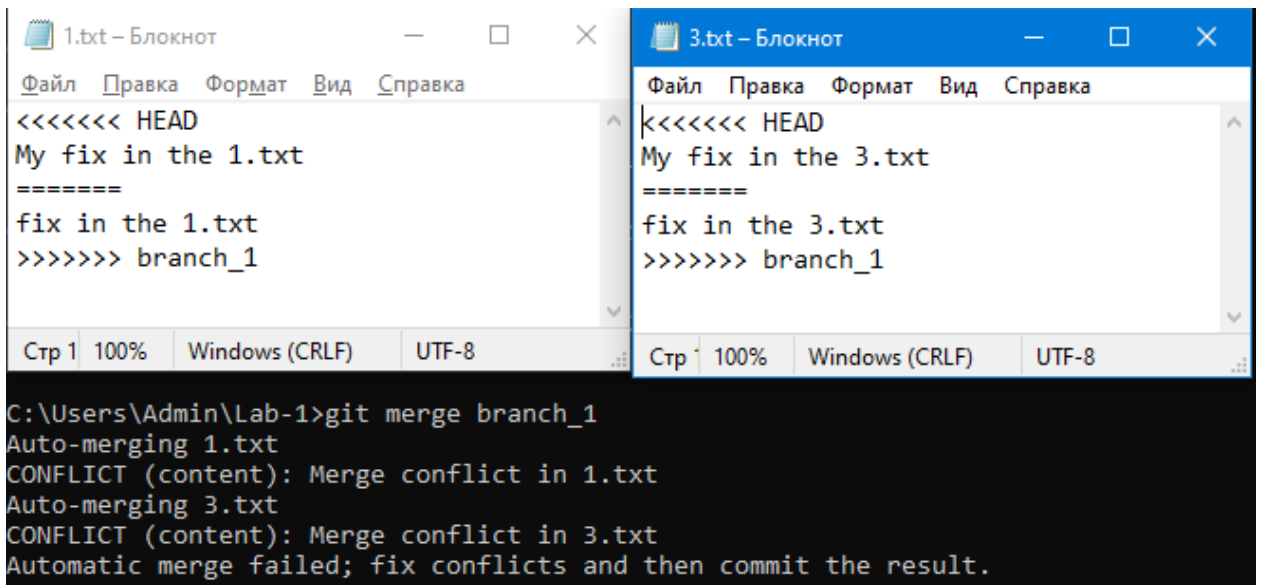


Рисунок 12 –Результат попытки слияния веток

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.

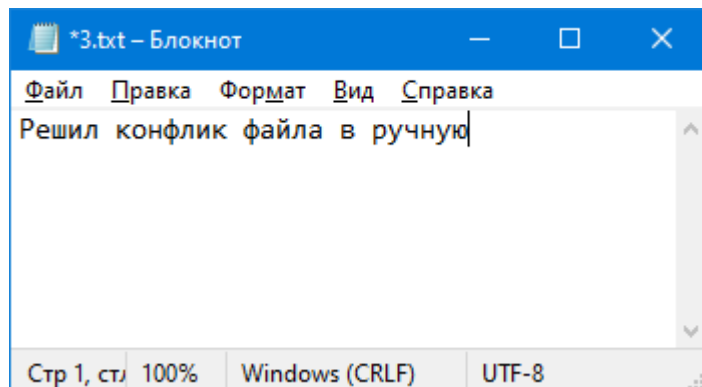


Рисунок 13.1 – Решение конфликта файлов вручную

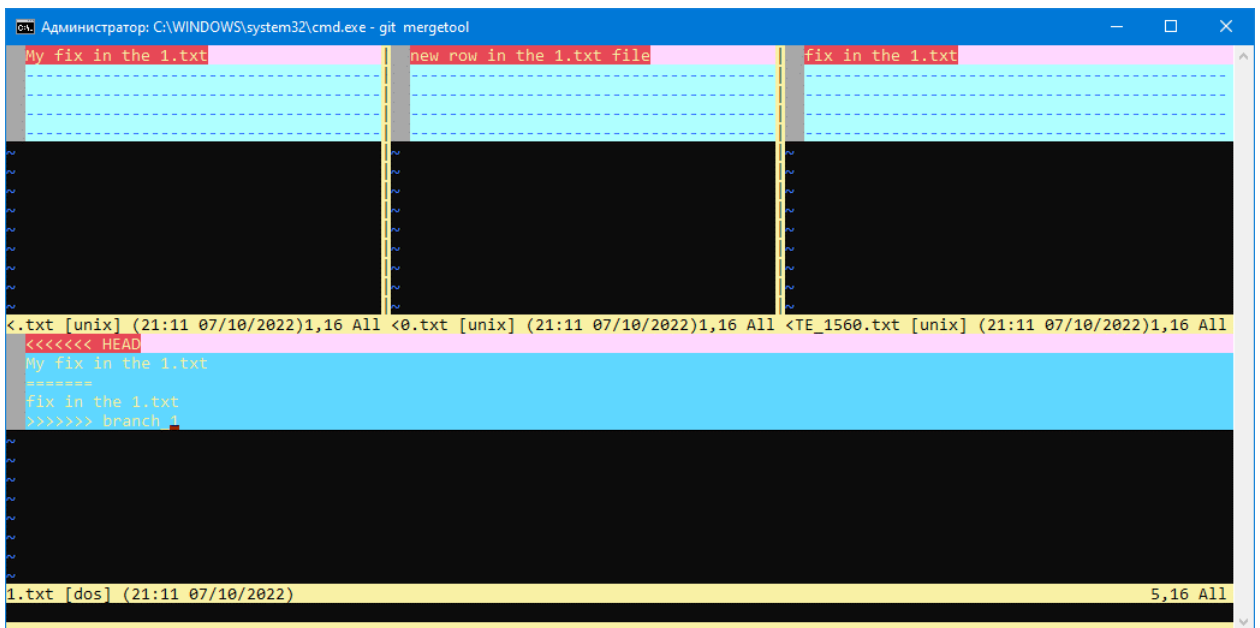


Рисунок 13.2 – решение конфликта файлов через git mergetool с использованием vimdiff

```
C:\Users\Admin\Lab-1>git mergetool

This message is displayed because 'merge.tool' is not configured.
See 'git mergetool --tool-help' or 'git help config' for more details.
'git mergetool' will now attempt to use one of the following tools:
tortoisemerge emerge vimdiff nvimdiff
Merging:
1.txt
3.txt

Normal merge conflict for '1.txt':
  {local}: modified file
  {remote}: modified file
Hit return to start merge resolution tool (vimdiff):
4 files to edit

Normal merge conflict for '3.txt':
  {local}: modified file
  {remote}: modified file
Hit return to start merge resolution tool (vimdiff):
4 files to edit
```

Рисунок 14 – результат работы git mergetool

14. Отправить ветку branch_1 на GitHub.


```
C:\Users\Admin\Lab-1>git push origin branch_1
Enumerating objects: 17, done.
Counting objects: 100% (17/17), done.
Delta compression using up to 12 threads
Compressing objects: 100% (12/12), done.
Writing objects: 100% (16/16), 1.52 KiB | 1.52 MiB/s, done.
Total 16 (delta 5), reused 1 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (5/5), done.
remote:
remote: Create a pull request for 'branch_1' on GitHub by visiting:
remote:   https://github.com/GlebMizin/Lab-1/pull/new/branch_1
remote:
To https://github.com/GlebMizin/Lab-1.git
 * [new branch]      branch_1 -> branch_1
```

Рисунок 15 – Отправка ветки на GitHub

15. Создать средствами GitHub удаленную ветку branch_3.

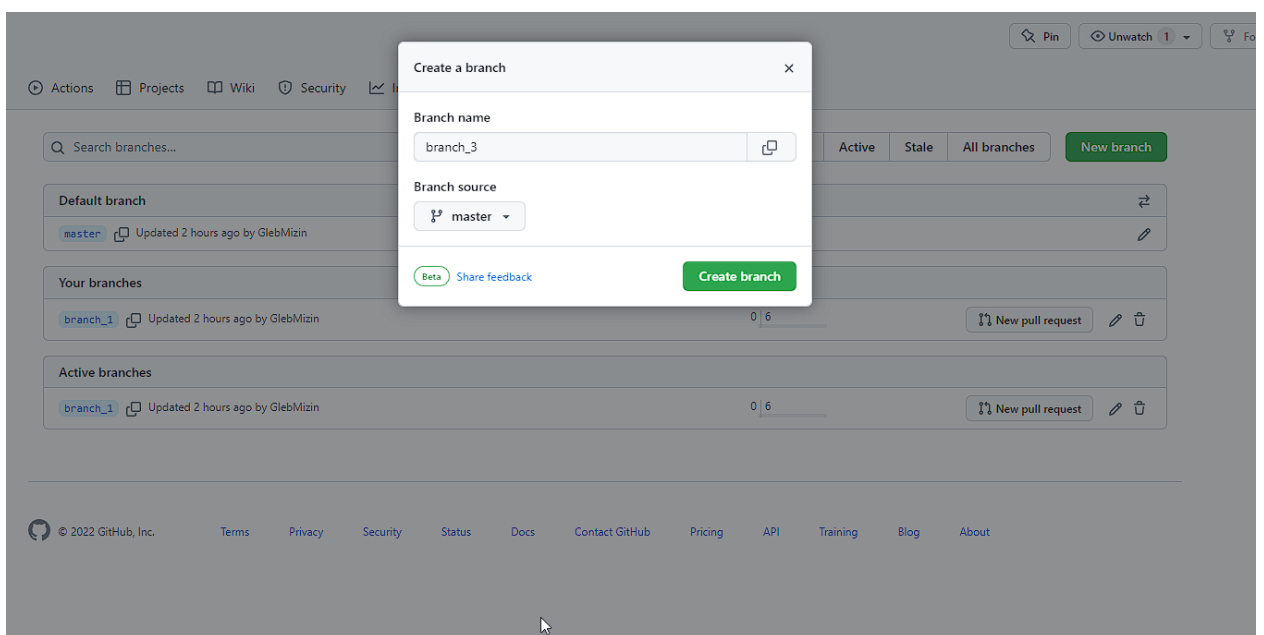


Рисунок 16 – создание удалённой ветки через GitHub

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

```
C:\Users\Admin\Lab-1>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 файл файл 2.txt строку "the final fantasy in the 4.txt file".

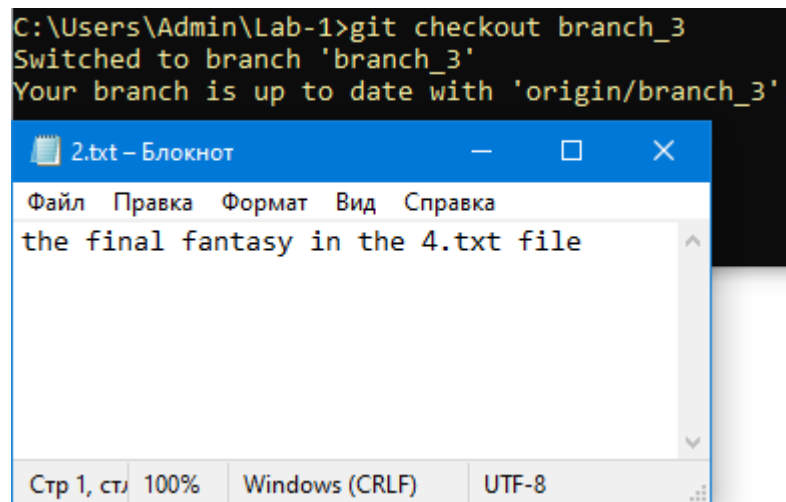


Рисунок 18 – переход на ветку и изменение файла

18. Выполнить перемещение ветки master на ветку branch_2

```
C:\Users\Admin\Lab-1>git rebase --continue
[detached HEAD 70b97e2] Изменил 1.txt и 3.txt
2 files changed, 2 insertions(+), 2 deletions(-)
Successfully rebased and updated refs/heads/branch_2.
```

Рисунок 18 – перемещение ветки master на ветку branch_2

19. Отправить изменения веток master и branch_2 на GitHub.

```
C:\Users\Admin\Lab-1>git push origin master
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/GlebMizin/Lab-1.git
    bf4d198..91800da  master -> master

C:\Users\Admin\Lab-1>git push origin branch_2
Enumerating objects: 15, done.
Counting objects: 100% (14/14), done.
Delta compression using up to 12 threads
Compressing objects: 100% (6/6), done.
Writing objects: 100% (11/11), 909 bytes | 909.00 KiB/s, done.
Total 11 (delta 3), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (3/3), completed with 1 local object.
remote:
remote: Create a pull request for 'branch_2' on GitHub by visiting:
remote:   https://github.com/GlebMizin/Lab-1/pull/new/branch_2
remote:
To https://github.com/GlebMizin/Lab-1.git
 * [new branch]      branch_2 -> branch_2
```

Рисунок 19 – отправление изменения веток на GitHub

Контрольные вопросы

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 fetch` это команда, которая обновляет локальную копию удаленного репозитория. А `git pull` перенесит изменения в удаленный репозиторий.

11. Как удалить локальную и удаленную ветки?

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

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

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 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

Filter

New branch

Default branch

✓ main3 hours ago

Recent branches

branch_23 hours ago

branch_31 hour ago

Other branches

origin/branch_22 hours ago

Merge into **branch_2**

Filter

Default branch

main3 hours ago

Recent branches

✓ branch_23 hours ago

branch_31 hour ago

Other branches

origin/branch_22 hours ago

✓

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

Create a merge commit