

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

**Кафедра инфокоммуникаций
Отчет по лабораторной работе № 1.3
«Основы ветвления Git»
по дисциплине «Основы программной инженерии»**

Выполнил студент группы

ПИЖ-б-о-21-1

Зиберов Александр

« » октября 2022 г.

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.

Выполнение работы:

Создать общедоступный репозиторий на GitHub, в котором будет использована лицензия MIT, рисунок 1.

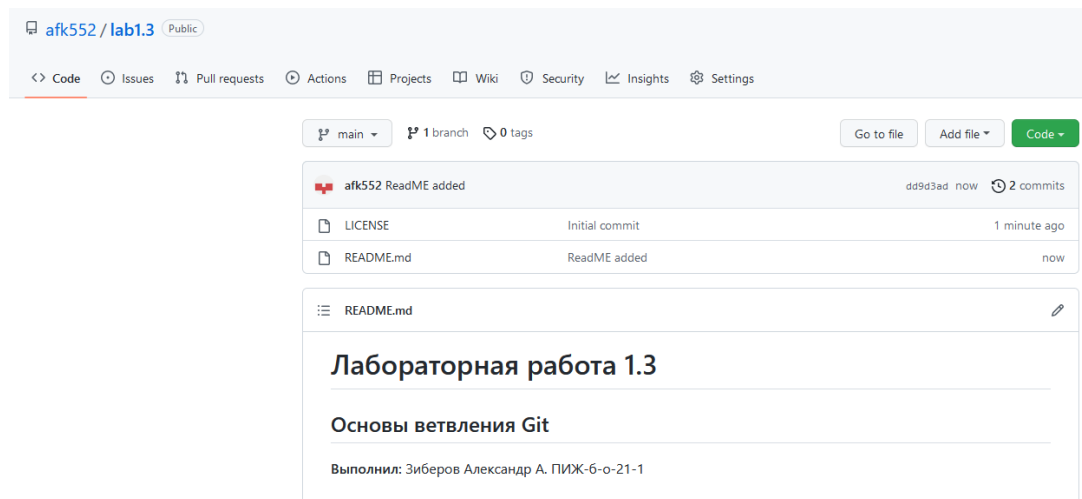


Рисунок 1 Репозиторий GitHub

Создать три файла: 1.txt, 2.txt, 3.txt. Проиндексировать первый файл и сделать коммит с комментарием "add 1.txt file", рисунок 2.

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

C:\git\lab1.3>git add 1.txt

C:\git\lab1.3>git status
On branch main
Your branch is up to date with 'origin/main'.

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

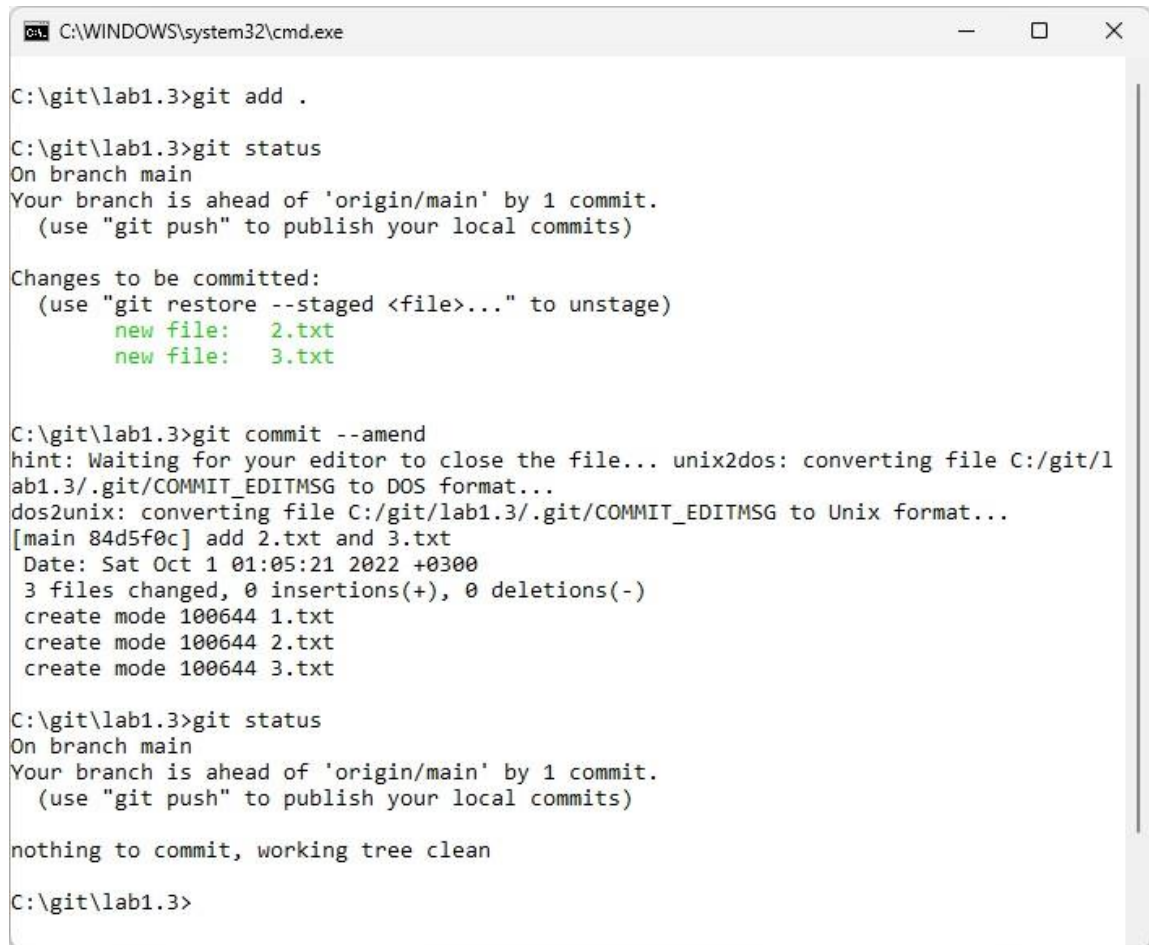
Untracked files:
  (use "git add <file>..." to include in what will be committed)
    2.txt
    3.txt

C:\git\lab1.3>git commit -m "add 1.txt file"
[main b800e63] add 1.txt file
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 1.txt

C:\git\lab1.3>
```

Рисунок 2 Окно командной строки

Проиндексировать второй и третий файлы. Переписать уже сделанный коммит с новым комментарием "add 2.txt and 3.txt.", рисунки 3-4.



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

C:\git\lab1.3>git add .

C:\git\lab1.3>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:\git\lab1.3>git commit --amend
hint: Waiting for your editor to close the file... unix2dos: converting file C:/git/lab1.3/.git/COMMIT_EDITMSG to DOS format...
dos2unix: converting file C:/git/lab1.3/.git/COMMIT_EDITMSG to Unix format...
[main 84d5f0c] add 2.txt and 3.txt
Date: Sat Oct 1 01:05:21 2022 +0300
3 files changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 1.txt
 create mode 100644 2.txt
 create mode 100644 3.txt

C:\git\lab1.3>git status
On branch main
Your branch is ahead of 'origin/main' by 1 commit.
(use "git push" to publish your local commits)

nothing to commit, working tree clean

C:\git\lab1.3>
```

Рисунок 3 Окно командной строки

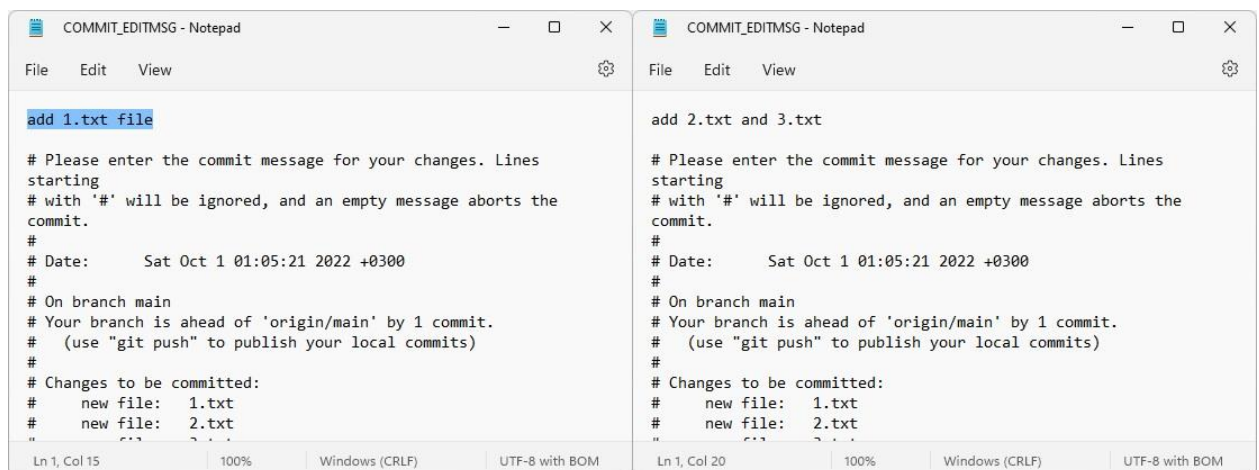
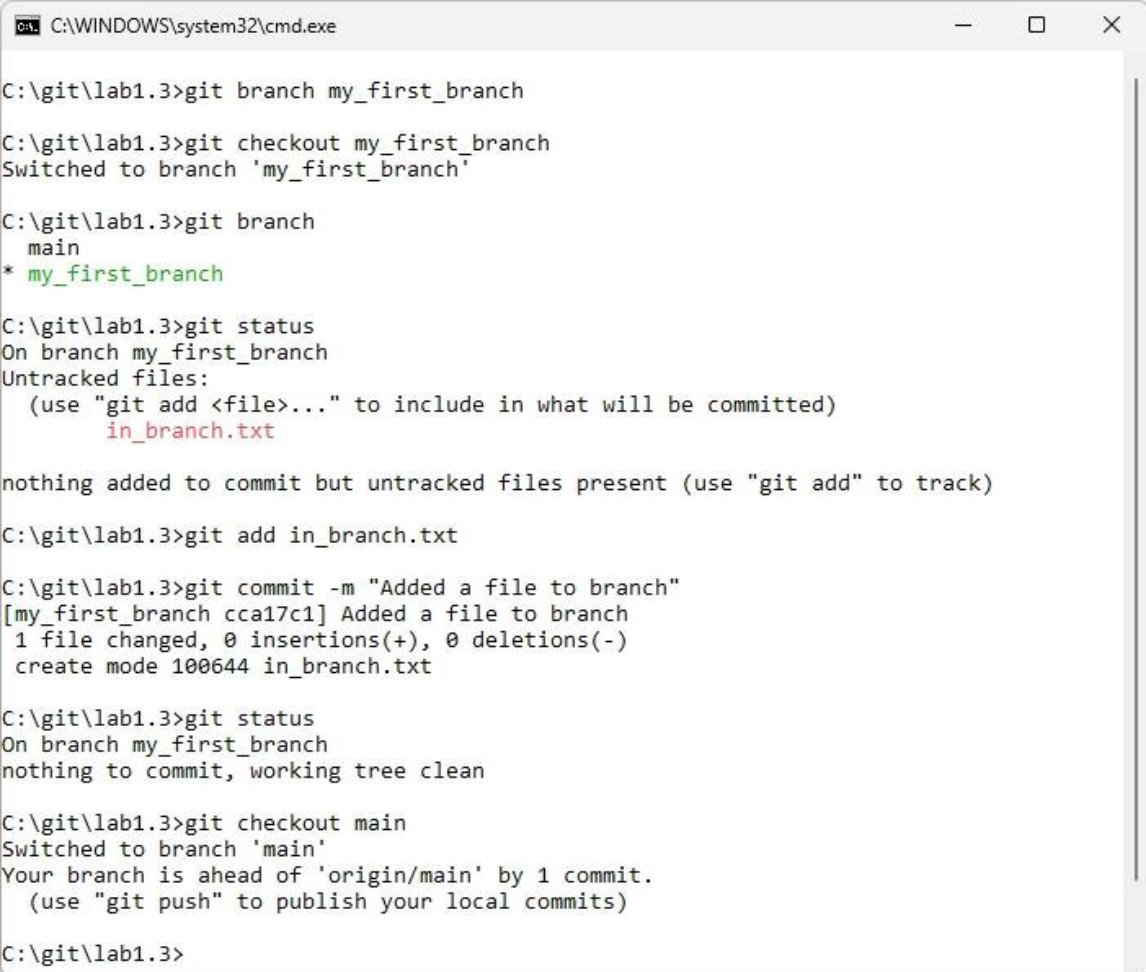


Рисунок 4 Окна блокнота

Создать новую ветку `my_first_branch`. Перейти на ветку и создать новый файл `in_branch.txt`, закоммитить изменения. Вернуться на ветку `master`.
Рисунок 5.



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

C:\git\lab1.3>git branch my_first_branch

C:\git\lab1.3>git checkout my_first_branch
Switched to branch 'my_first_branch'

C:\git\lab1.3>git branch
  main
* my_first_branch

C:\git\lab1.3>git status
On branch my_first_branch
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        in_branch.txt

nothing added to commit but untracked files present (use "git add" to track)

C:\git\lab1.3>git add in_branch.txt

C:\git\lab1.3>git commit -m "Added a file to branch"
[my_first_branch cca17c1] Added a file to branch
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 in_branch.txt

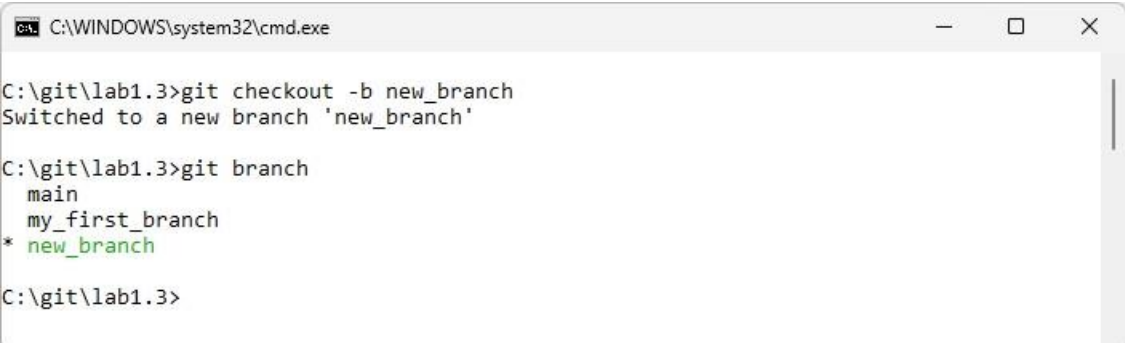
C:\git\lab1.3>git status
On branch my_first_branch
nothing to commit, working tree clean

C:\git\lab1.3>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)

C:\git\lab1.3>
```

Рисунок 5 Окно командной строки

Создать и сразу перейти на ветку `new_branch`, рисунок 6.



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

C:\git\lab1.3>git checkout -b new_branch
Switched to a new branch 'new_branch'

C:\git\lab1.3>git branch
  main
  my_first_branch
* new_branch

C:\git\lab1.3>
```

Рисунок 6 Окно командной строки

Сделать изменения в файле 1.txt, добавить строčku “new row in the 1.txt file”, закоммитить изменения, рисунок 7.

```
1.txt - Notepad
File Edit View
new row in the 1.txt file

C:\WINDOWS\system32\cmd.exe
C:\git\lab1.3>git branch
main
my_first_branch
* new_branch

C:\git\lab1.3>git status
On branch new_branch
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   1.txt

no changes added to commit (use "git add" and/or "git commit -a")

C:\git\lab1.3>git add 1.txt

C:\git\lab1.3>git commit -m "New row in 1.txt"
[new_branch 02ae7e1] New row in 1.txt
1 file changed, 1 insertion(+)
```

Рисунок 7 Окно командной строки

Перейти на ветку master и слить ветки master и my_first_branch, после чего слить ветки master и new_branch. Удалить ветки my_first_branch и new_branch. Рисунок 8.

```
C:\WINDOWS\system32\cmd.exe
C:\git\lab1.3>git branch
main
my_first_branch
* new_branch

C:\git\lab1.3>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)

C:\git\lab1.3>git merge my_first_branch
Updating 84d5f0c..cca17c1
Fast-forward
 in_branch.txt | 0
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 in_branch.txt

C:\git\lab1.3>git merge new_branch
Merge made by the 'ort' strategy.
 1.txt | 1 +
1 file changed, 1 insertion(+)

C:\git\lab1.3>git branch -d my_first_branch
Deleted branch my_first_branch (was cca17c1).

C:\git\lab1.3>git branch -d new_branch
Deleted branch new_branch (was 02ae7e1).
```

Рисунок 8 Окно командной строки

Создать ветки branch_1 и branch_2, рисунок 9.



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

C:\git\lab1.3>git branch branch_1

C:\git\lab1.3>git branch branch_2

C:\git\lab1.3>git branch
  branch_1
  branch_2
* main
```

Рисунок 9 Окно командной строки

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. Рисунки 10-11.

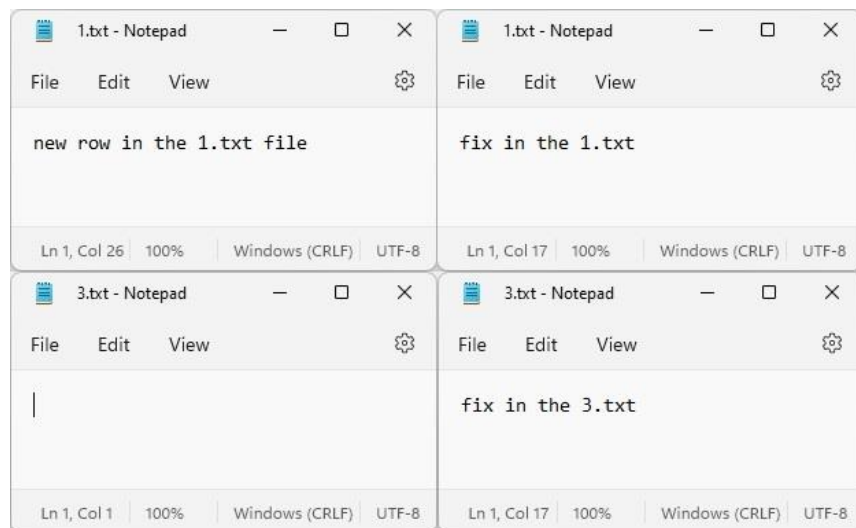
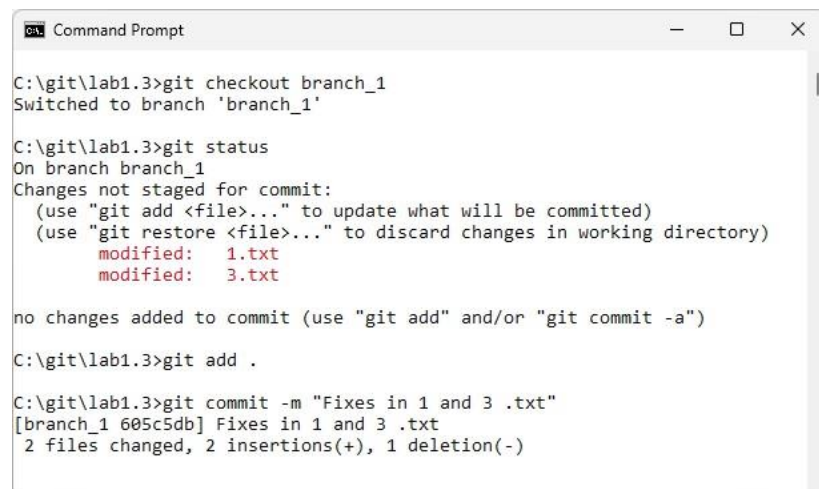


Рисунок 10 Окна блокнота



```
Command Prompt

C:\git\lab1.3>git checkout branch_1
Switched to branch 'branch_1'

C:\git\lab1.3>git status
On branch branch_1
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   1.txt
        modified:   3.txt

no changes added to commit (use "git add" and/or "git commit -a")

C:\git\lab1.3>git add .

C:\git\lab1.3>git commit -m "Fixes in 1 and 3 .txt"
[branch_1 605c5db] Fixes in 1 and 3 .txt
2 files changed, 2 insertions(+), 1 deletion(-)
```

Рисунок 11 Окно командной строки

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. Рисунки 12-13.

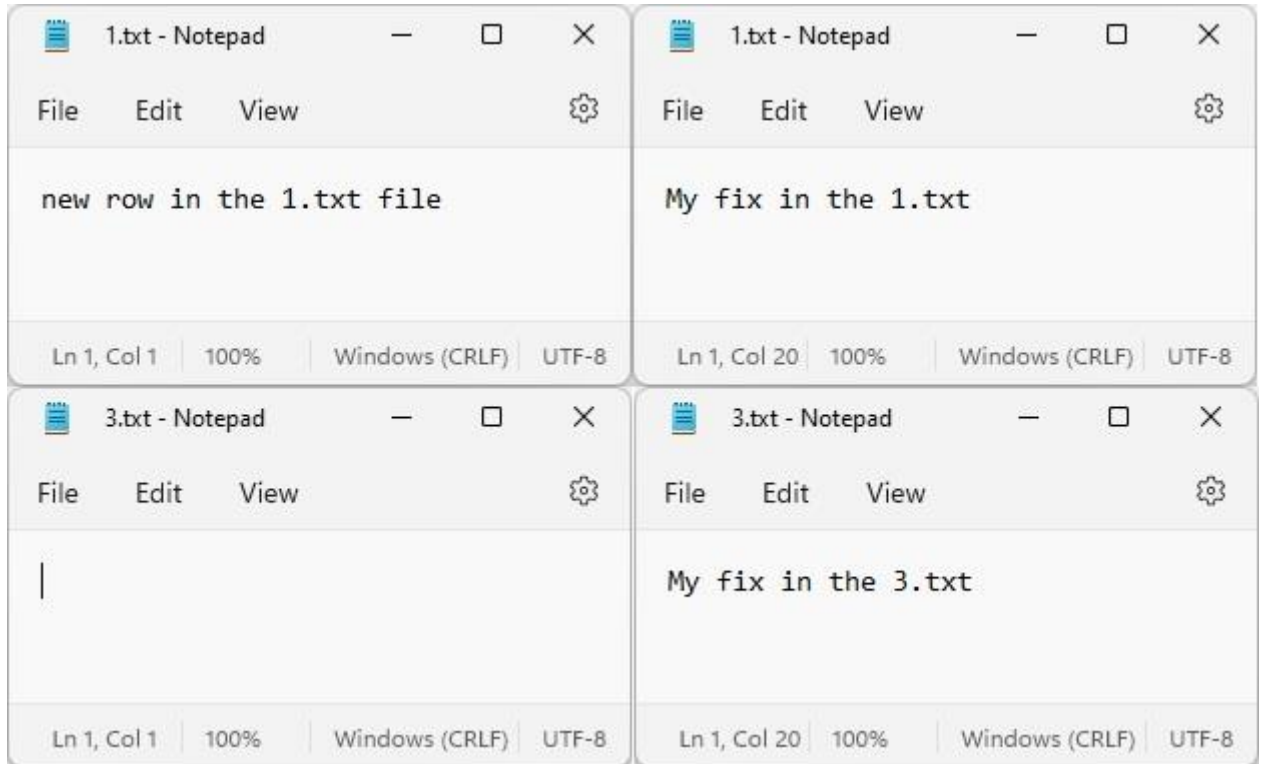


Рисунок 12 Окна блокнота

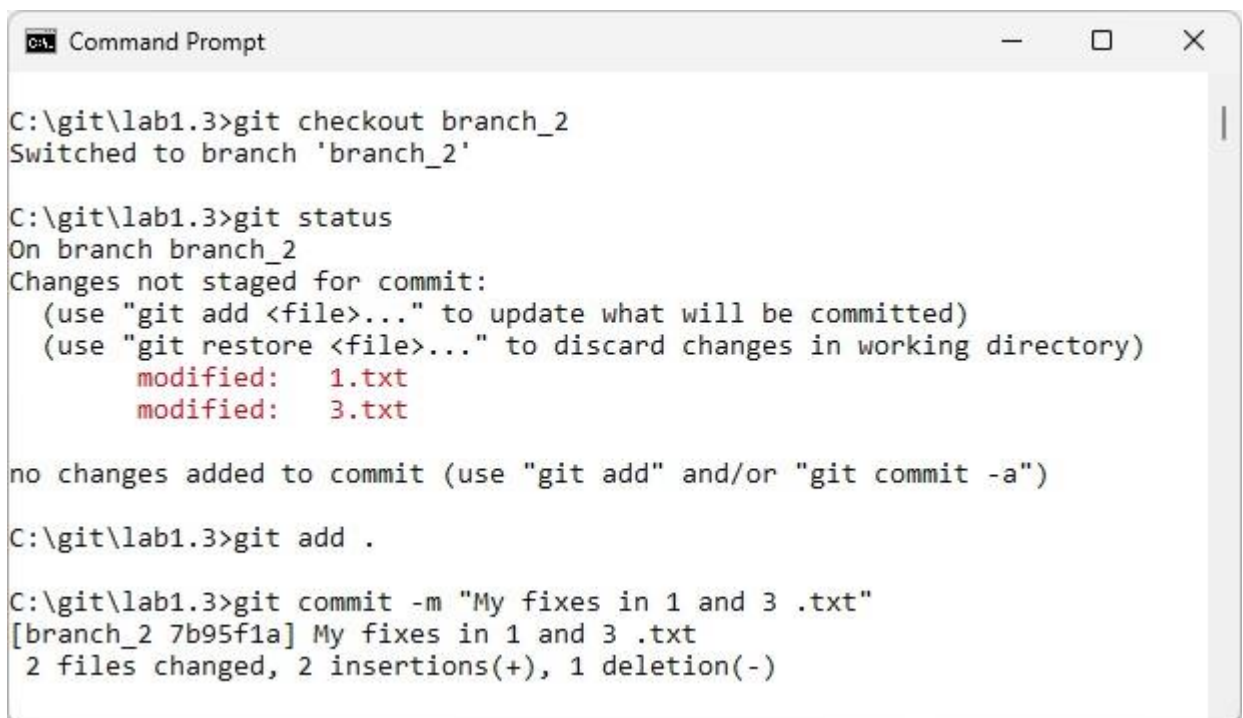
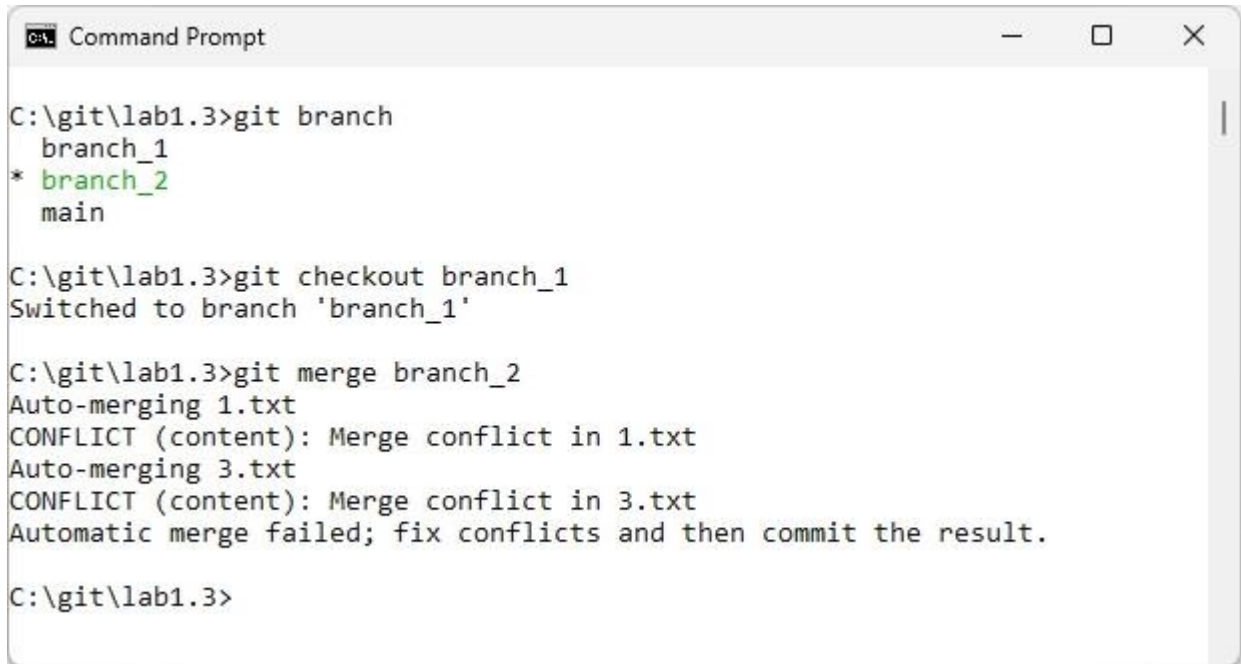


Рисунок 13 Окно командной строки

Слить изменения ветки branch_2 в ветку branch_1, рисунок 14.



```
C:\git\lab1.3>git branch
  branch_1
* branch_2
  main

C:\git\lab1.3>git checkout branch_1
Switched to branch 'branch_1'

C:\git\lab1.3>git merge branch_2
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:\git\lab1.3>
```

Рисунок 14 Окно командной строки

Решить конфликт файла 1.txt в ручном режиме, рисунок 15.

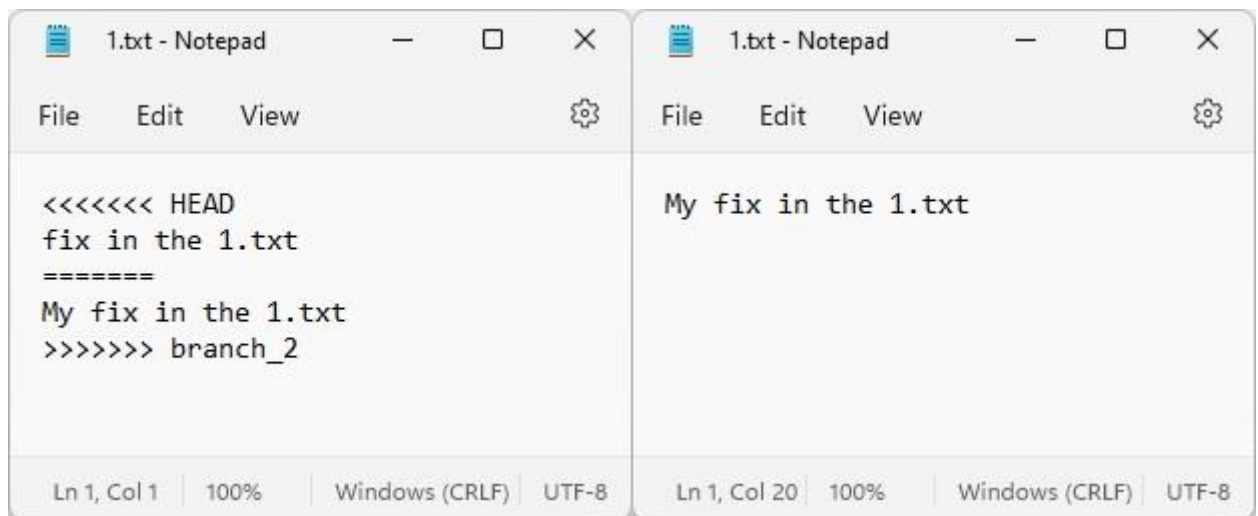


Рисунок 15 Окно командной строки

Решить конфликт файла .txt используя команду `git mergetool` с помощью одной из доступных утилит (`vimdiff`), рисунок 16.



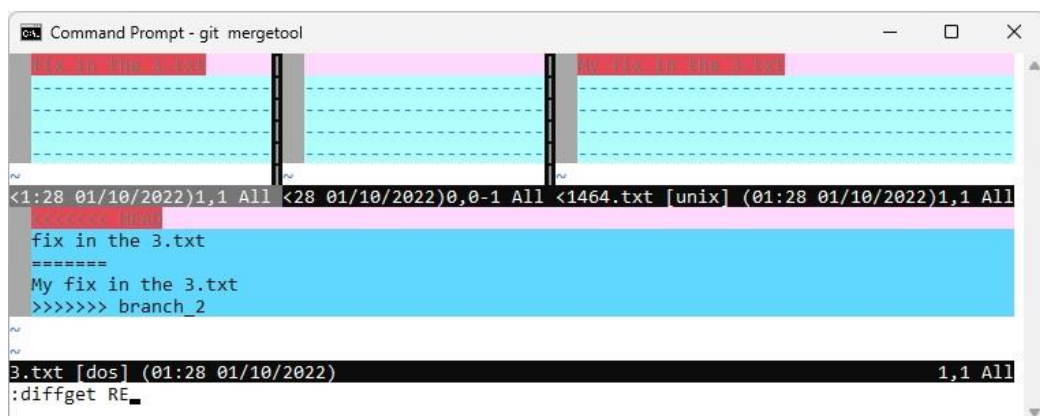
```
C:\git\lab1.3>git mergetool

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:
3.txt

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

Рисунок 16 Окно командной строки

Появилось окно инструмента `vimdiff`. Нажатием `CTRL + W` переходим в нижнее окно, рисунок 17.



```
vimdiff
<1:28 01/10/2022>1,1 All <28 01/10/2022>0,0-1 All <1464.txt [unix] (01:28 01/10/2022)1,1 All
fix in the 3.txt
=====
My fix in the 3.txt
>>>>>> branch_2

3.txt [dos] (01:28 01/10/2022) 1,1 All
:diffget RE
```

Рисунок 17 Окно редактора vimdiff

Нажимаем сочетание клавиш `SHIFT + :` и вводим команду `:diffget RE` для выбора варианта файла из ветки `branch_2` (`RE` – `Remote`), рисунок 18.



```
vimdiff
<28 01/10/2022>1,0-1 All <28 01/10/2022>0,0-1 All <1464.txt [unix] (01:28 01/10/2022)1,1 All
fix in the 3.txt
=====
My fix in the 3.txt
>>>>>> branch_2

3.txt[+] [dos] (01:28 01/10/2022) 1,1 All
:diffget RE
```

Рисунок 18 Окно редактора vimdiff

Сохраняем изменения и выходим командой :wqa, рисунок 19.



Рисунок 19 Окно редактора vimdiff

Очищаем мусор от редактора, рисунок 20.

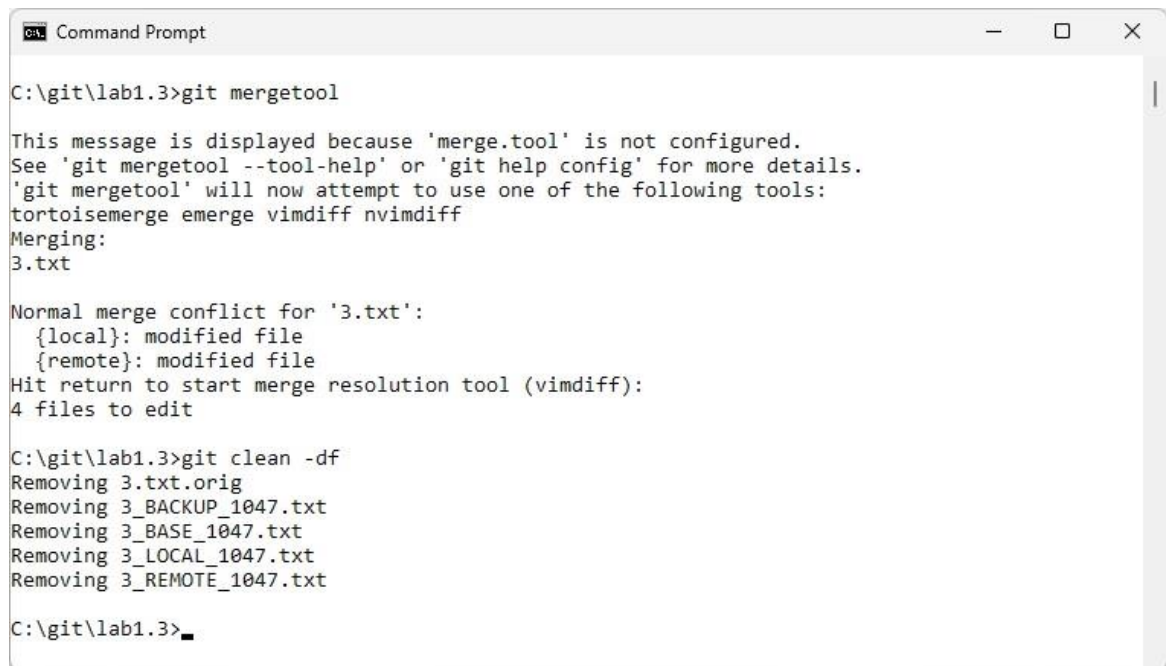


Рисунок 20 Окно командной строки

Отправить ветку branch_1 на GitHub, рисунок 21.

```
Command Prompt

C:\git\lab1.3>git branch
* branch_1
  branch_2
  main

C:\git\lab1.3>git status
On branch branch_1
Your branch is up to date with 'origin/branch_1'.

All conflicts fixed but you are still merging.
(use "git commit" to conclude merge)

Changes to be committed:
  modified:   1.txt
  modified:   3.txt

C:\git\lab1.3>git add .

C:\git\lab1.3>git commit -m "Resolved conflict when merging"
[branch_1 45fd079] Resolved conflict when merging

C:\git\lab1.3>git push --set-upstream origin branch_1
Enumerating objects: 13, done.
Counting objects: 100% (13/13), done.
Delta compression using up to 8 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (7/7), 682 bytes | 682.00 KiB/s, done.
Total 7 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), done.
To https://github.com/afk552/lab1.3
   605c5db..45fd079  branch_1 -> branch_1
branch 'branch_1' set up to track 'origin/branch_1'.

C:\git\lab1.3>
```

Рисунок 21 Окно командной строки

Получаем ветку branch_1 на сайте, рисунок 22.

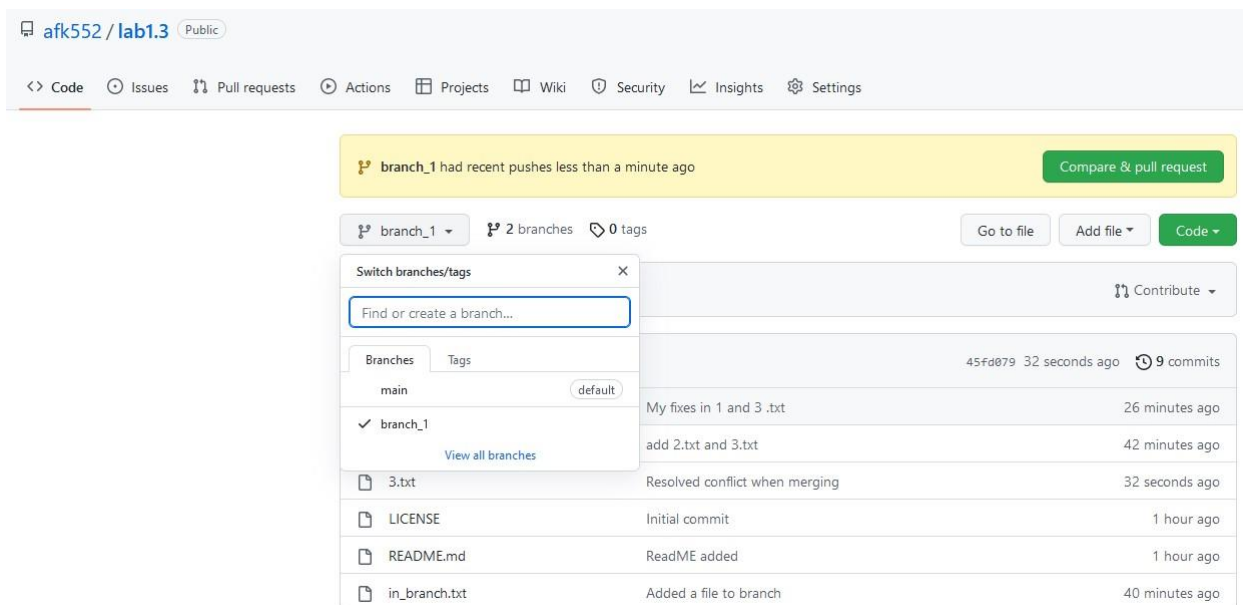


Рисунок 22 Окно репозитория GitHub

Создать средствами GitHub удаленную ветку branch_3, рисунок 23.

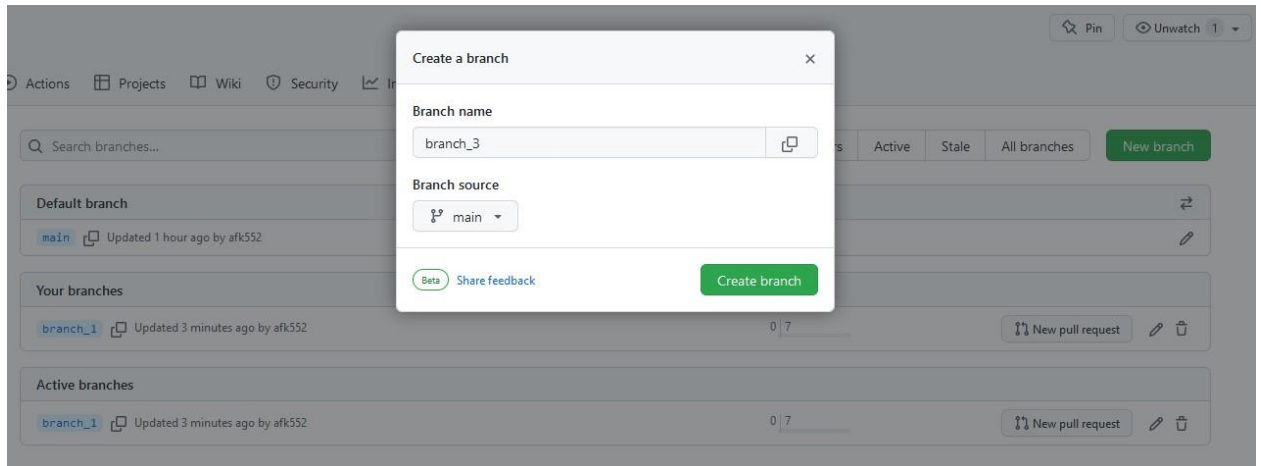


Рисунок 23 Меню создания новой ветки GitHub

Создать в локальном репозитории ветку отслеживания удаленной ветки branch_3, рисунок 24.

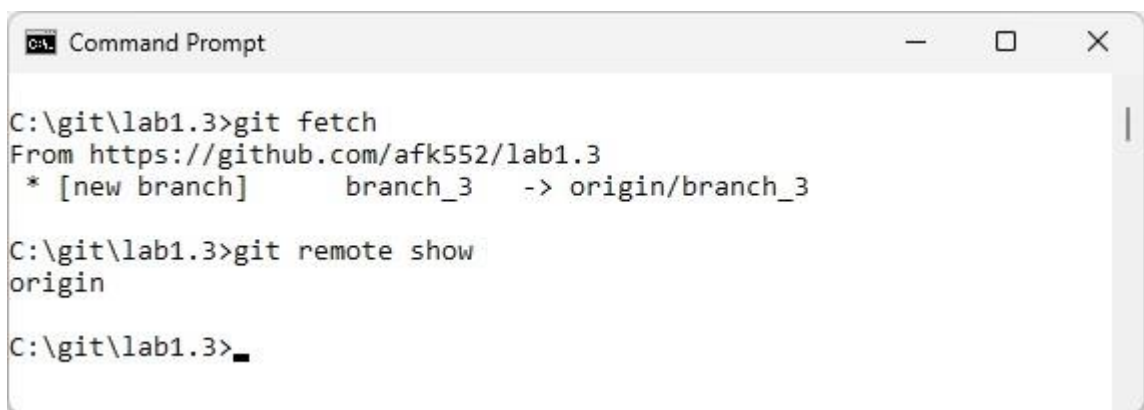


Рисунок 24 Окно командной строки

Перейти на ветку branch_3, рисунок 25.

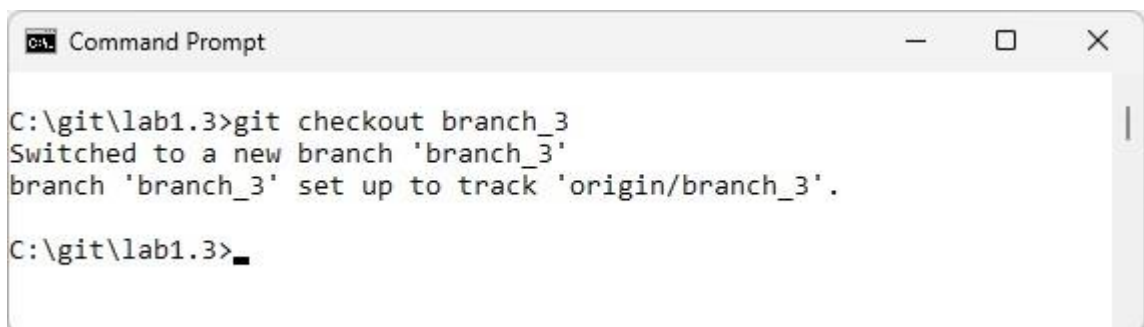


Рисунок 25 Окно командной строки

Добавить файл 2.txt строку "the final fantasy in the 4.txt file", рисунок 26.

```
C:\git\lab1.3>git status
On branch branch_3
Your branch is up to date with 'origin/branch_3'.

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

nothing added to commit but untracked files present (use "git add" to track)

C:\git\lab1.3>git add 2.txt
C:\git\lab1.3>git status
On branch branch_3
Your branch is up to date with 'origin/branch_3'.

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

C:\git\lab1.3>git commit -m "Added a text line"
[branch_3 82e5b9c] Added a text line
1 file changed, 1 insertion(+)
create mode 100644 2.txt

C:\git\lab1.3>
```

Рисунок 26 Окно командной строки

Выполнить перемещение ветки master на ветку branch_2, рисунок 27.

```
C:\git\lab1.3>git branch
branch_1
* branch_2
branch_3
main

C:\git\lab1.3>git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.

C:\git\lab1.3>git rebase branch_2
Successfully rebased and updated refs/heads/main.

C:\git\lab1.3>
```

Рисунок 27 Окно командной строки

Отправить изменения веток master и branch_2 на GitHub, рисунок 28.

```
Command Prompt

C:\git\lab1.3>git push --set-upstream origin main
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/afk552/lab1.3
   3040aa8..7b95f1a  main -> main
branch 'main' set up to track 'origin/main'.

C:\git\lab1.3>git push --set-upstream origin branch_2
Everything up-to-date
branch 'branch_2' set up to track 'origin/branch_2'.

C:\git\lab1.3>
```

Рисунок 28 Окно командной строки

Полученные ветки на GitHub, рисунок 29.



Рисунок 29 Окно командной строки

Результат в ветке main, рисунок 30.

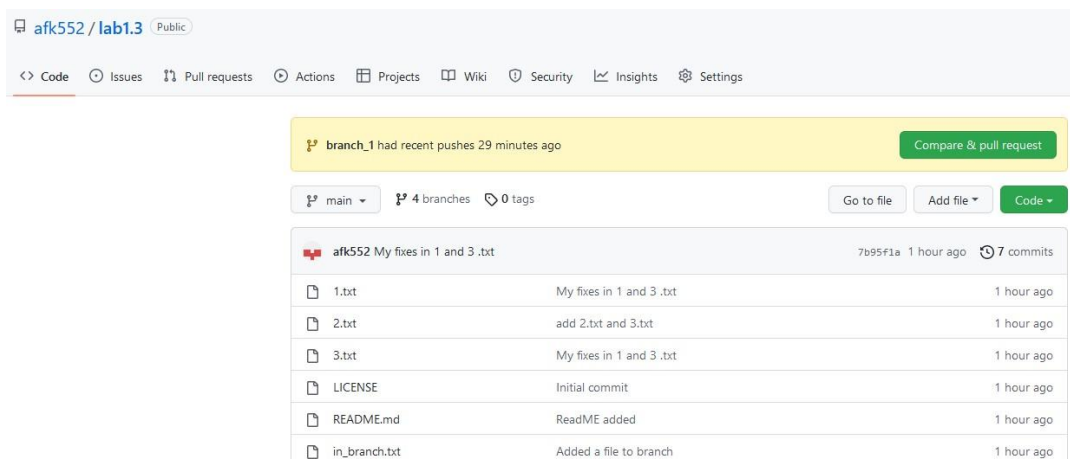


Рисунок 30 Окно командной строки

Вывод: В ходе выполнения работы были изучены основы ветвления в СКВ git, а именно: создание ветвей локально и удаленно, их слияние и перемещение, решения конфликтов файлов при слиянии веток.

Контрольные вопросы:

1. Что такое ветка?

Ветка в Git – перемещаемый указатель на один из коммитов (история коммитов) с целью воссоздания снимков для отслеживания изменений и возможного отката. По умолчанию ветка называется main, но можно создать свои. Ветки в основном используются для внесения изменений в проект, не затрагивая текущую версию, а при готовности к релизу такую ветку сливают в основную для применения изменений.

2. Что такое HEAD?

HEAD – указатель на коммит в вашей репозитории, который станет родителем следующего коммита; / специальный указатель на коммит, который показывает относительно какой ветки он будет внесен. При смене ветки на другую, этот указатель переключится между последними коммитами выбранной ветки.

3. Способы создания веток

Командой `git branch <название ветки>;`

Командой `git checkout -b <название ветки> //` в этом случае произойдет создание ветки с заданным именем и сразу осуществится переход на неё.

4. Как узнать текущую ветку?

Командой `git branch`, она отобразит зелёную звездочку напротив текущей ветки.

5. Как переключаться между ветками?

Командой `git checkout <название ветки>`.

6. Что такое удаленная ветка?

Удаленная ветка – это указатель в удалённом репозитории, включая ветки, теги и т.д. Полный список удалённых ссылок можно получить с помощью команды `git remote show <имя удаленного репозитория>` для получения удалённых веток и дополнительной информации.

7. Что такое ветка отслеживания?

Ветка отслеживания – это ссылка на определённое состояние удалённой ветки. Git перемещает их автоматически при любой коммуникации с удалённым репозиторием, чтобы гарантировать точное соответствие с ним.

8. Как создать ветку отслеживания?

Командой `git branch --track <название ветки> origin/<название ветки>`.

9. Как отправить изменения из локальной ветки в удаленную ветку

Отправить изменения командой `git push <имя удаленного репозитория, например, origin> <название ветки>`.

Командой `git fetch origin` можно получить обновления с сервера.

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

Команда `git fetch` получает с сервера все изменения, которых ещё нет локально, но не будет изменять состояние текущей рабочей копии. Эта команда просто получает данные и позволяет самостоятельно сделать слияние. По этой причине использовать её безопаснее.

Команда `git pull` в большинстве случаев является командой `git fetch`, за которой непосредственно следует команда `git merge`. `git pull` определит сервер и ветку, за которыми следит текущая ветка, получит данные с этого сервера и затем попытается слить удалённую ветку.

11. Как удалить локальную и удалённую ветки?

Удалить локальную ветку: `git branch -d <название ветки>;`

Удалить локальную ветку: `git push <имя удаленного репозитория> -d <название удаленной 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`?

Типы веток в `git-flow`:

Главные ветви:

- master
- develop

Вспомогательные ветки:

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

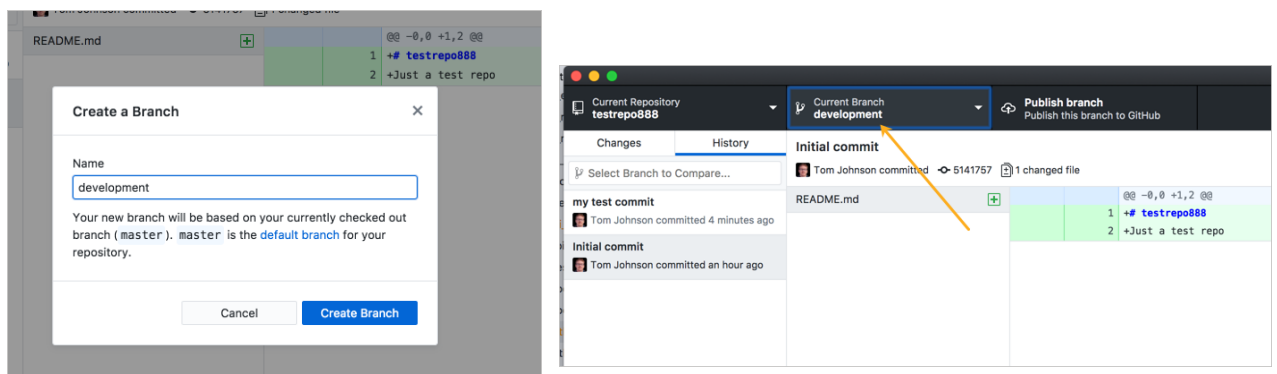
Работа с ветками в этой модели организовано разделением ветвей по задачам. Ветка master – основная ветка проекта. Вспомогательные ветки представляют из себя разграниченные области: релиз, хотфикс, фичи. Таким образом, мы вносим в одни из них необходимые изменения, а затем, как они оттестированы и становятся готовы к релизу, вспомогательные ветви вливаются в develop, а после – в master. При этом существуют определенные правила, например, если в данный момент существует ветвь релиза (release branch), то ветвь исправления (hotfix branch) должна вливаться в неё, а не в ветвь разработки (develop) и т.д.

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.

GitHub Desktop – официально приложение/интерфейс для работы с GitHub, работающий на основе git.

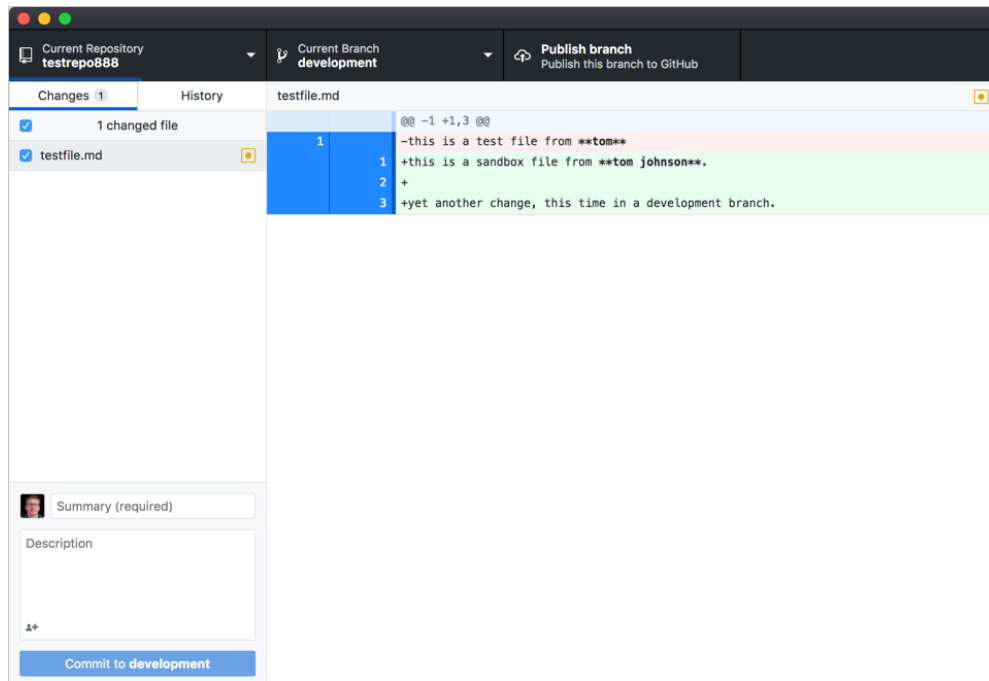
Создание ветки:

В GitHub Desktop переходим в Branch > New Branch и создаем новую ветвь и нажмем Create Branch. В меню можно переключить текущую ветку с которой будем работать.



Коммит изменения в ветку:

После внесения изменений закоммитим изменения в левом нижнем углу, кликнув на Commit to <название ветки>.



Слияние веток:

Переключимся на ветку, в которую хотим объединить другую ветку. Объединяем кнопкой «Merge into».

