

**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

Сумин Никита Сергеевич.

« » __ 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.

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Owner * Repository name *

BrandooDi / mywork3

Great repository names are short and memorable. Need inspiration? How about [bug-free-waffle?](#)

Description (optional)

☒ **Public**
Anyone on the internet can see this repository. You choose who can commit.

☐ **Private**
You choose who can see and commit to this repository.

Initialize this repository with:
Skip this step if you're importing an existing repository.

☐ **Add a README file**
This is where you can write a long description for your project. [Learn more.](#)

Add .gitignore
Choose which files not to track from a list of templates. [Learn more.](#)

.gitignore template: None ▾

Choose a license
A license tells others what they can and can't do with your code. [Learn more.](#)

License: MIT License ▾

You are creating a public repository in your personal account.

[Create repository](#)

Рисунок 1 – Создание репозитория

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

компьютер > Nikita (D:) > www > mywork3 >

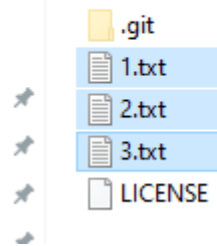


Рисунок 2 - Создание текстовых файлов

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

```
MINGW64:/d/www/mywork3
user@MINGW64 /d/www/mywork3 (main)
$ git add 1.txt

user@MINGW64 /d/www/mywork3 (main)
$ git commit --m "add 1.txt file"
[main ea2996b] add 1.txt file
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 1.txt

user@MINGW64 /d/www/mywork3 (main)
$
```

Рисунок 3 - Коммит 1

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

```
MINGW64:/d/www/mywork3
user@MINGW64 /d/www/mywork3 (main)
$ git add 2.txt

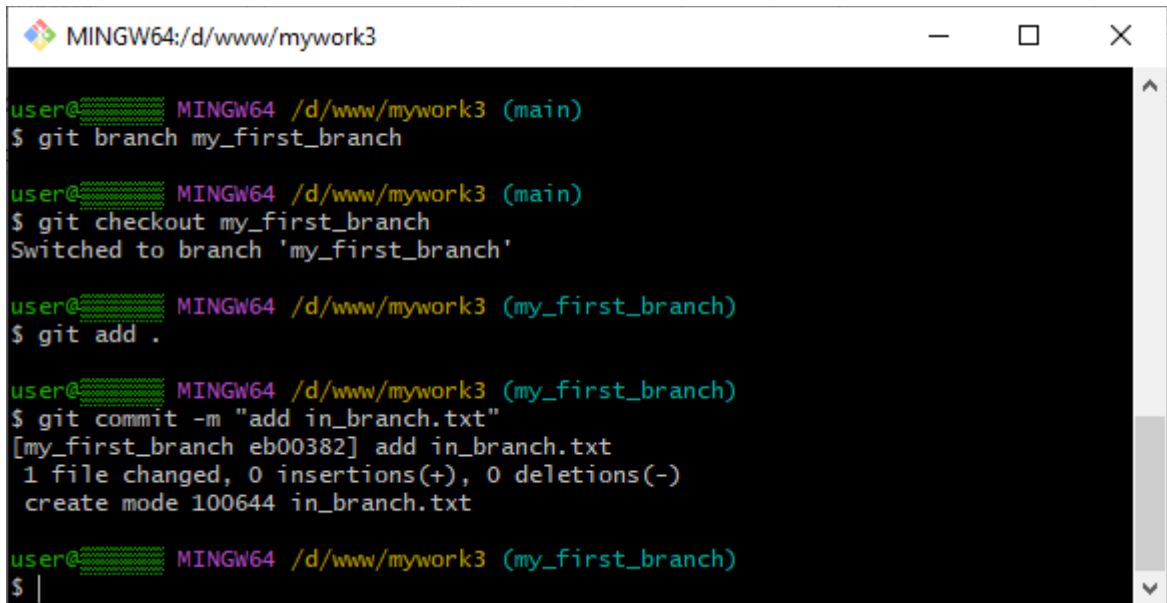
user@MINGW64 /d/www/mywork3 (main)
$ git add 3.txt

user@MINGW64 /d/www/mywork3 (main)
$ git commit --amend -m "add 2.txt and 3.txt"
[main cd2b880] add 2.txt and 3.txt
Date: Sat Jun 4 23:42:05 2022 +0300
3 files changed, 0 insertions(+), 0 deletions(-)
create mode 100644 1.txt
create mode 100644 2.txt
create mode 100644 3.txt

user@MINGW64 /d/www/mywork3 (main)
$
```

Рисунок 4 - Перезаписанный коммит

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



```
MINGW64:/d/www/mywork3

user@MINGW64 /d/www/mywork3 (main)
$ git branch my_first_branch

user@MINGW64 /d/www/mywork3 (main)
$ git checkout my_first_branch
Switched to branch 'my_first_branch'

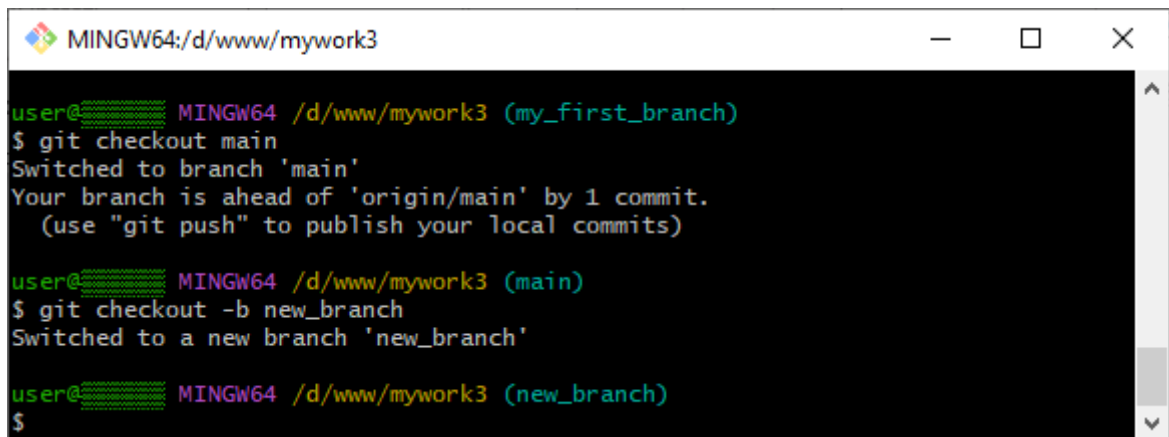
user@MINGW64 /d/www/mywork3 (my_first_branch)
$ git add .

user@MINGW64 /d/www/mywork3 (my_first_branch)
$ git commit -m "add in_branch.txt"
[my_first_branch eb00382] add in_branch.txt
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 in_branch.txt

user@MINGW64 /d/www/mywork3 (my_first_branch)
$ |
```

Рисунок 5 - Новая ветка my_first_branch

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



```
MINGW64:/d/www/mywork3

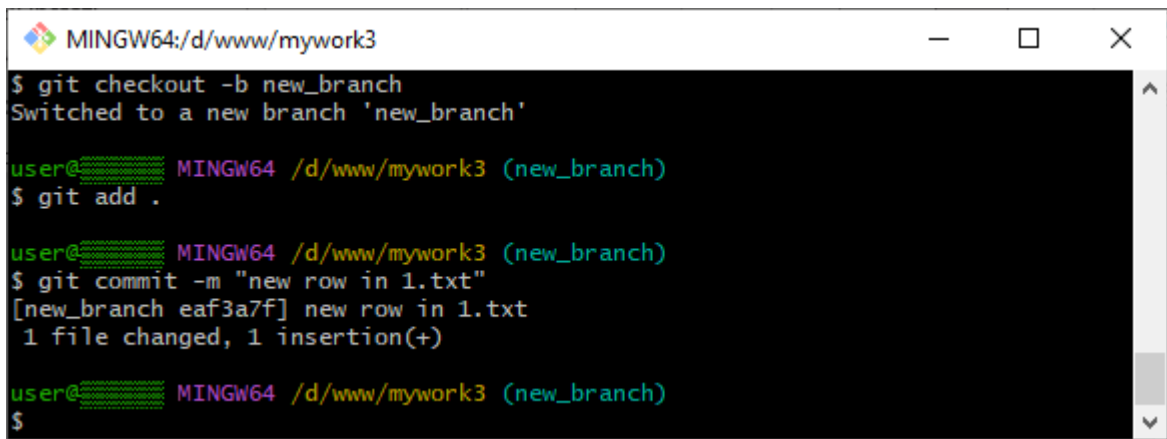
user@MINGW64 /d/www/mywork3 (my_first_branch)
$ git checkout main
Switched to branch 'main'
Your branch is ahead of 'origin/main' by 1 commit.
(use "git push" to publish your local commits)

user@MINGW64 /d/www/mywork3 (main)
$ git checkout -b new_branch
Switched to a new branch 'new_branch'

user@MINGW64 /d/www/mywork3 (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.



```
MINGW64:/d/www/mywork3
$ git checkout -b new_branch
Switched to a new branch 'new_branch'

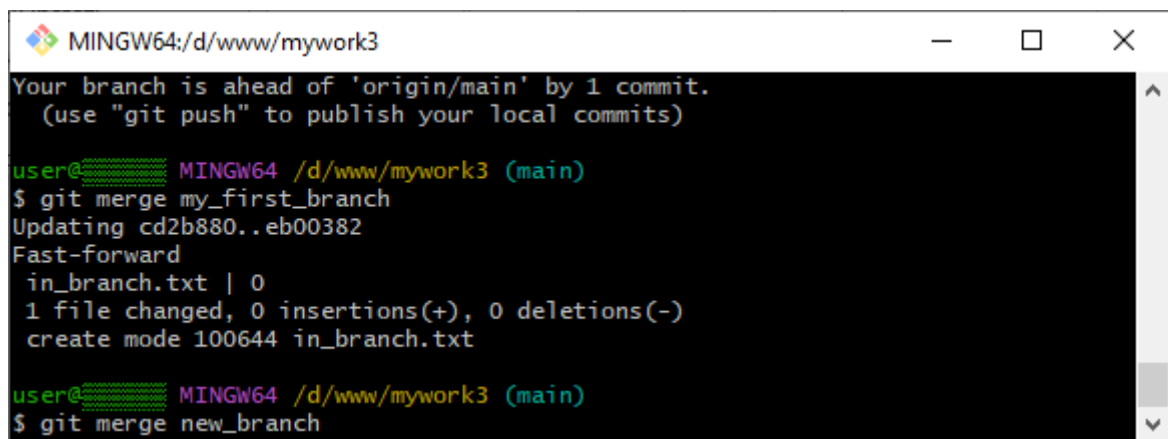
user@MINGW64 /d/www/mywork3 (new_branch)
$ git add .

user@MINGW64 /d/www/mywork3 (new_branch)
$ git commit -m "new row in 1.txt"
[new_branch eaf3a7f] new row in 1.txt
1 file changed, 1 insertion(+)

user@MINGW64 /d/www/mywork3 (new_branch)
$
```

Рисунок 7 - Новая строка в файле 1

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



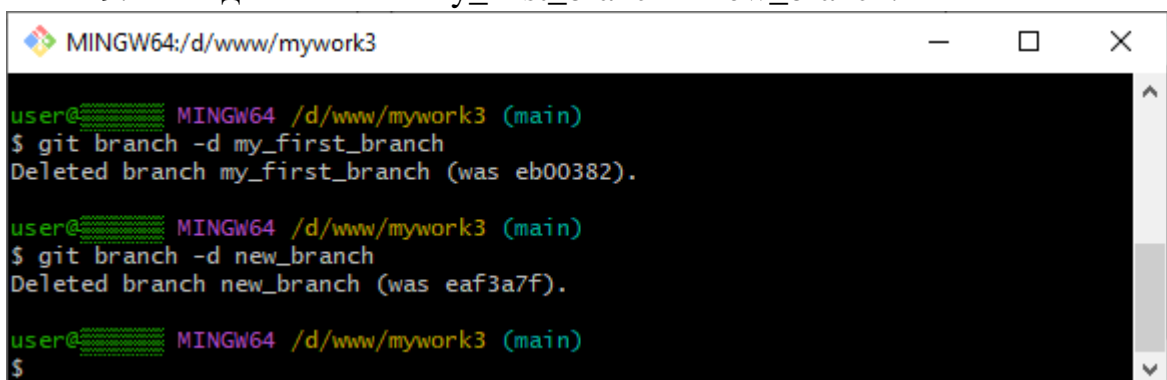
```
MINGW64:/d/www/mywork3
Your branch is ahead of 'origin/main' by 1 commit.
(use "git push" to publish your local commits)

user@MINGW64 /d/www/mywork3 (main)
$ git merge my_first_branch
Updating cd2b880..eb00382
Fast-forward
 in_branch.txt | 0
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 in_branch.txt

user@MINGW64 /d/www/mywork3 (main)
$ git merge new_branch
```

Рисунок 8 - Слияние веток

9. Удалил ветки my_first_branch и new_branch.



```
MINGW64:/d/www/mywork3

user@MINGW64 /d/www/mywork3 (main)
$ git branch -d my_first_branch
Deleted branch my_first_branch (was eb00382).

user@MINGW64 /d/www/mywork3 (main)
$ git branch -d new_branch
Deleted branch new_branch (was eaf3a7f).

user@MINGW64 /d/www/mywork3 (main)
$
```

Рисунок 9 - Удаление веток 10.

Создал ветки branch_1 и branch_2.

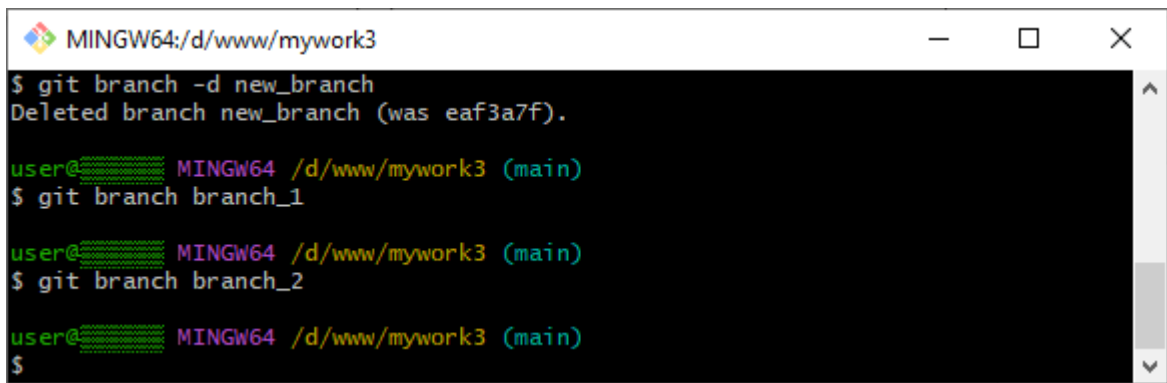
A screenshot of a terminal window titled 'MINGW64:/d/www/mywork3'. The terminal shows the following commands and output: '\$ git branch -d new_branch' followed by 'Deleted branch new_branch (was eaf3a7f).', then 'user@MINGW64 /d/www/mywork3 (main)' followed by '\$ git branch branch_1', then 'user@MINGW64 /d/www/mywork3 (main)' followed by '\$ git branch branch_2', and finally 'user@MINGW64 /d/www/mywork3 (main)' followed by '\$'.

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.

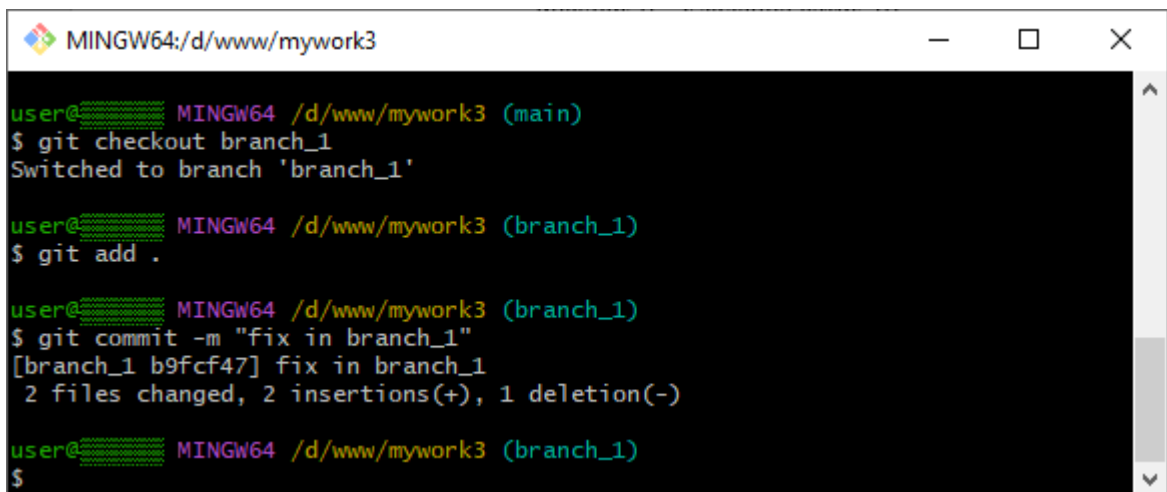
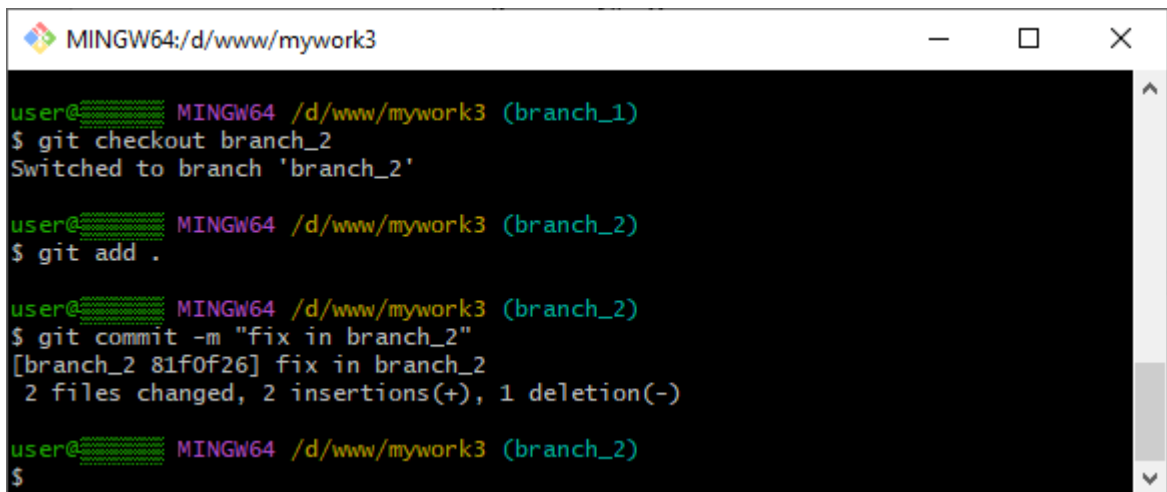
A screenshot of a terminal window titled 'MINGW64:/d/www/mywork3'. The terminal shows the following commands and output: 'user@MINGW64 /d/www/mywork3 (main)' followed by '\$ git checkout branch_1' and 'Switched to branch 'branch_1'', then 'user@MINGW64 /d/www/mywork3 (branch_1)' followed by '\$ git add .', then 'user@MINGW64 /d/www/mywork3 (branch_1)' followed by '\$ git commit -m "fix in branch_1"', then '[branch_1 b9fcf47] fix in branch_1' followed by '2 files changed, 2 insertions(+), 1 deletion(-)', and finally 'user@MINGW64 /d/www/mywork3 (branch_1)' followed by '\$'.

Рисунок 11 - Изменение файлов на ветке branch_1

12. Перешел на ветку branch_2 и также изменил файл 1.txt, удалил все содержимое и добавил 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.



```
MINGW64:/d/www/mywork3

user@MINGW64 /d/www/mywork3 (branch_1)
$ git checkout branch_2
Switched to branch 'branch_2'

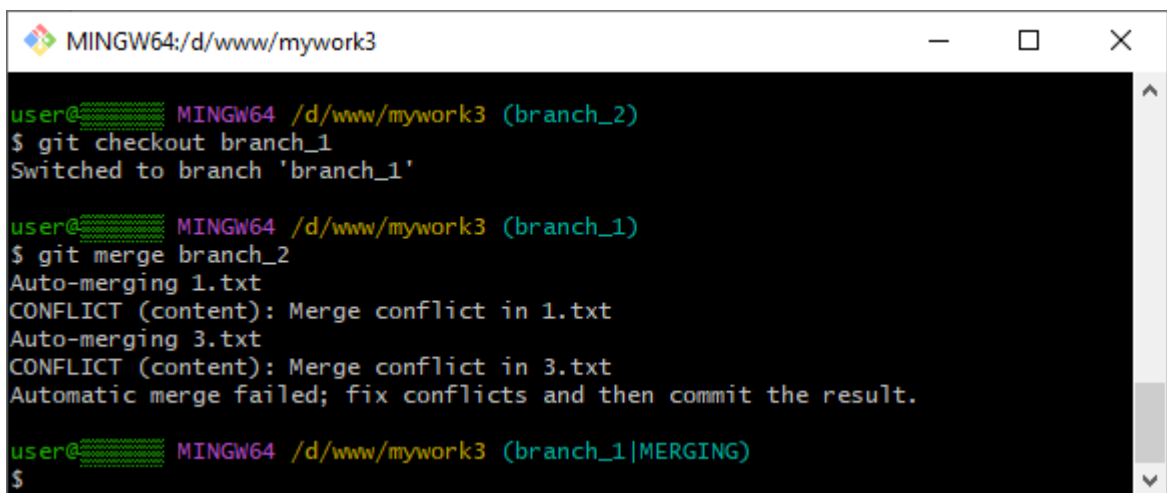
user@MINGW64 /d/www/mywork3 (branch_2)
$ git add .

user@MINGW64 /d/www/mywork3 (branch_2)
$ git commit -m "fix in branch_2"
[branch_2 81f0f26] fix in branch_2
2 files changed, 2 insertions(+), 1 deletion(-)

user@MINGW64 /d/www/mywork3 (branch_2)
$
```

Рисунок 12 - Изменение файлов на ветке branch_2 13.

Слил изменения ветки branch_2 в ветку branch_1.



```
MINGW64:/d/www/mywork3

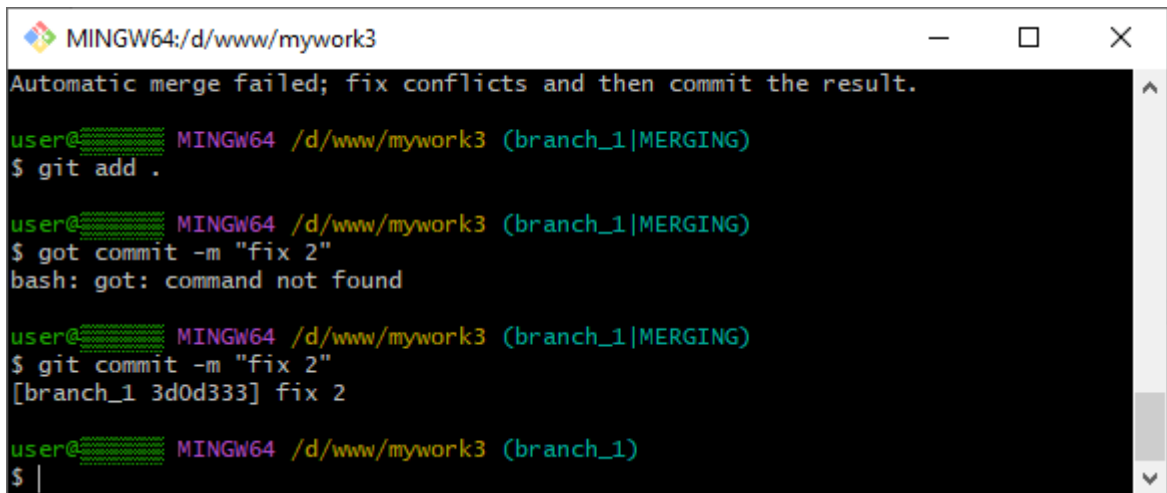
user@MINGW64 /d/www/mywork3 (branch_2)
$ git checkout branch_1
Switched to branch 'branch_1'

user@MINGW64 /d/www/mywork3 (branch_1)
$ 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.

user@MINGW64 /d/www/mywork3 (branch_1|MERGING)
$
```

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.



```
MINGW64:/d/www/mywork3
Automatic merge failed; fix conflicts and then commit the result.

user@MINGW64 /d/www/mywork3 (branch_1|MERGING)
$ git add .

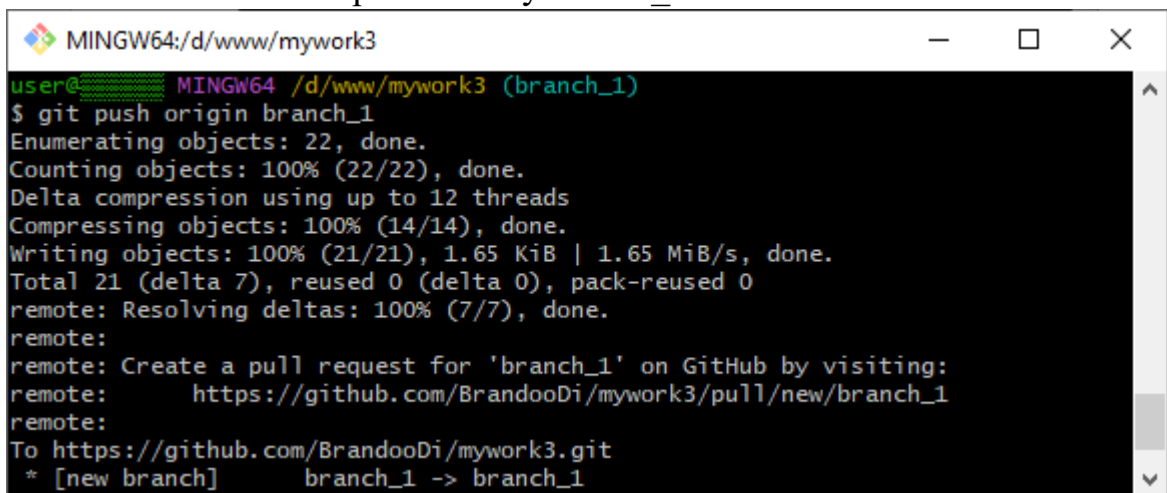
user@MINGW64 /d/www/mywork3 (branch_1|MERGING)
$ got commit -m "fix 2"
bash: got: command not found

user@MINGW64 /d/www/mywork3 (branch_1|MERGING)
$ git commit -m "fix 2"
[branch_1 3d0d333] fix 2

user@MINGW64 /d/www/mywork3 (branch_1)
$ |
```

Рисунок 14 - Разрешение конфликта

15. Отправил ветку branch_1 на GitHub.



```
MINGW64:/d/www/mywork3
user@MINGW64 /d/www/mywork3 (branch_1)
$ git push origin branch_1
Enumerating objects: 22, done.
Counting objects: 100% (22/22), done.
Delta compression using up to 12 threads
Compressing objects: 100% (14/14), done.
Writing objects: 100% (21/21), 1.65 KiB | 1.65 MiB/s, done.
Total 21 (delta 7), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (7/7), done.
remote:
remote: Create a pull request for 'branch_1' on GitHub by visiting:
remote:   https://github.com/BrandooDi/mywork3/pull/new/branch_1
remote:
To https://github.com/BrandooDi/mywork3.git
* [new branch]      branch_1 -> branch_1
```

Рисунок 15 - Отправление ветки branch_1

16. Создал средствами GitHub удаленную ветку branch_3.

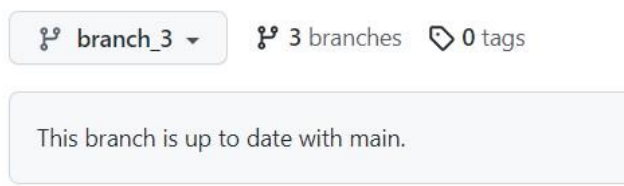
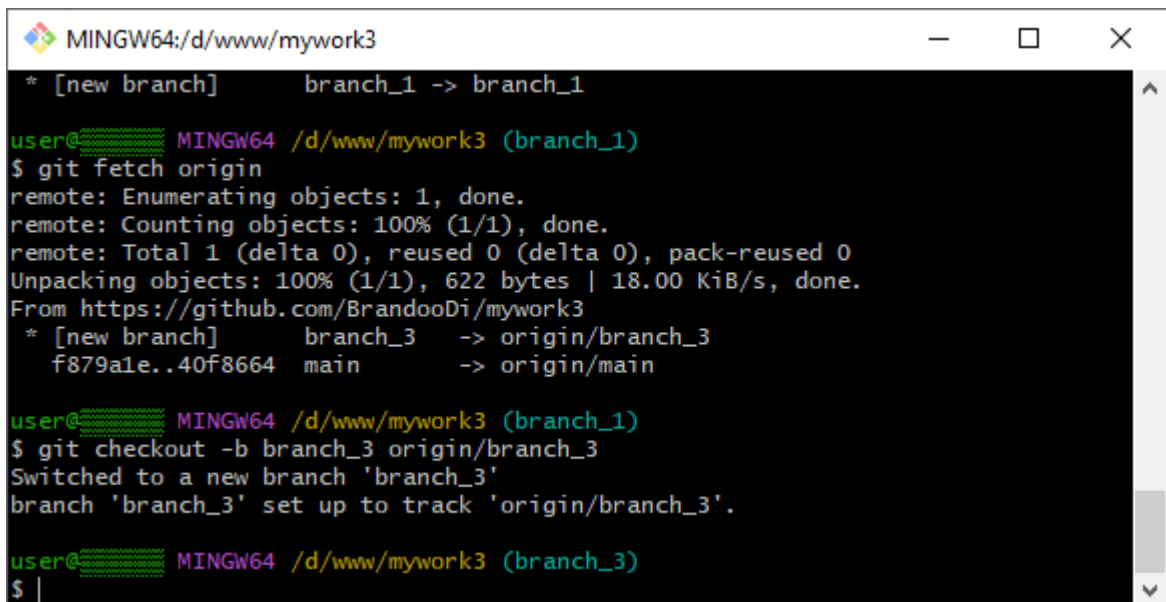


Рисунок 16 - Создание ветки branch_3

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



```
MINGW64:/d/www/mywork3
* [new branch]      branch_1 -> branch_1

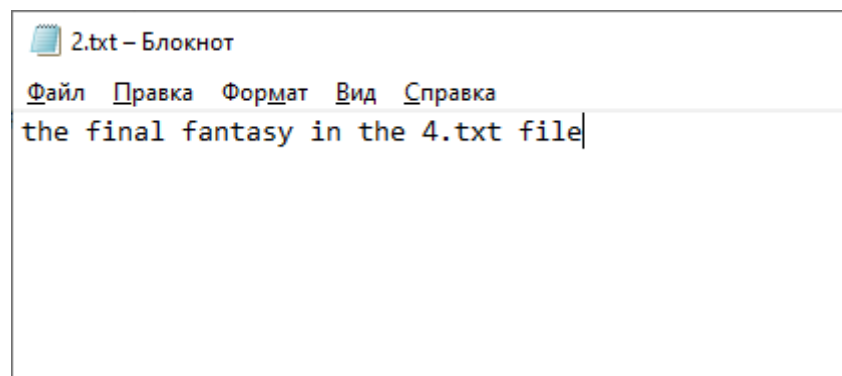
user@MINGW64 /d/www/mywork3 (branch_1)
$ git fetch origin
remote: Enumerating objects: 1, done.
remote: Counting objects: 100% (1/1), done.
remote: Total 1 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (1/1), 622 bytes | 18.00 KiB/s, done.
From https://github.com/BrandooDi/mywork3
* [new branch]      branch_3 -> origin/branch_3
f879a1e..40f8664  main -> origin/main

user@MINGW64 /d/www/mywork3 (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'.

user@MINGW64 /d/www/mywork3 (branch_3)
$
```

Рисунок 17 - Создание ветки отслеживания

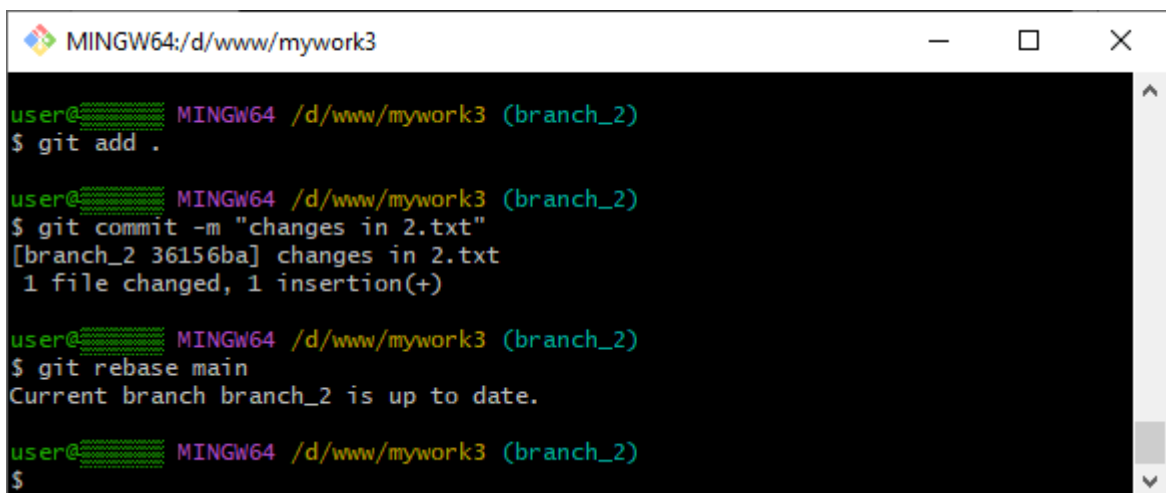
18. Перешел на ветку branch_2 и добавил в файл 2.txt строку "the final fantasy in the 4.txt file".



```
2.txt - Блокнот
Файл  Правка  Формат  Вид  Справка
the final fantasy in the 4.txt file
```

Рисунок 18 - Изменение в файле 2

19. Выполнил перемещение ветки master на ветку branch_2.



```
MINGW64:/d/www/mywork3

user@MINGW64 /d/www/mywork3 (branch_2)
$ git add .

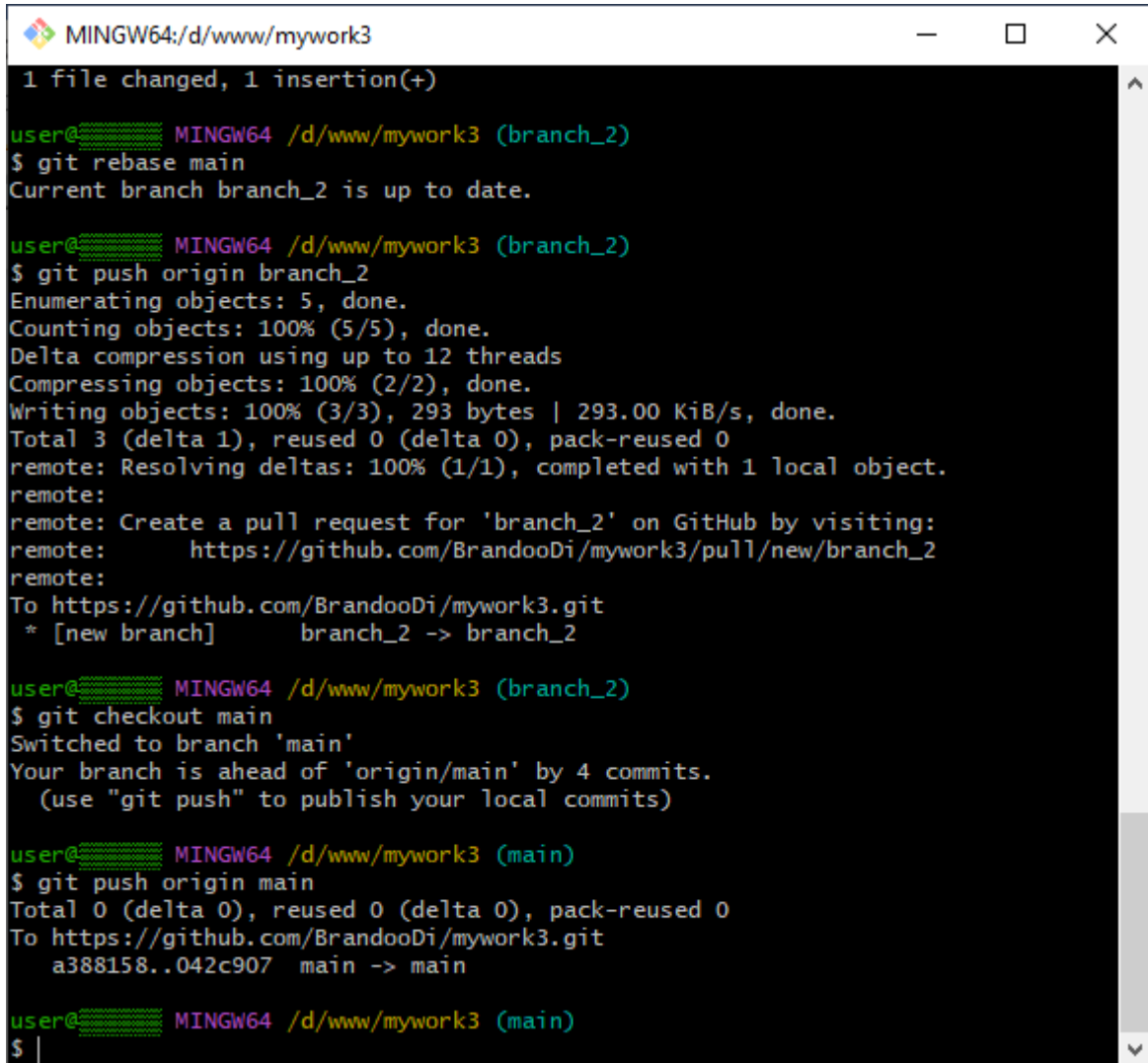
user@MINGW64 /d/www/mywork3 (branch_2)
$ git commit -m "changes in 2.txt"
[branch_2 36156ba] changes in 2.txt
1 file changed, 1 insertion(+)

user@MINGW64 /d/www/mywork3 (branch_2)
$ git rebase main
Current branch branch_2 is up to date.

user@MINGW64 /d/www/mywork3 (branch_2)
$
```

Рисунок 19 - Перемещение ветки

20. Отправил изменения веток master и branch_2 на GitHub.



```
1 file changed, 1 insertion(+)

user@MINGW64 /d/www/mywork3 (branch_2)
$ git rebase main
Current branch branch_2 is up to date.

user@MINGW64 /d/www/mywork3 (branch_2)
$ git push origin branch_2
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 12 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 293 bytes | 293.00 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/BrandooDi/mywork3/pull/new/branch_2
remote:
To https://github.com/BrandooDi/mywork3.git
 * [new branch]      branch_2 -> branch_2

user@MINGW64 /d/www/mywork3 (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)

user@MINGW64 /d/www/mywork3 (main)
$ git push origin main
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/BrandooDi/mywork3.git
 a388158..042c907  main -> main

user@MINGW64 /d/www/mywork3 (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, относительно 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>` или на 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 удалённым 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/gitflowworkflow>, <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?

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.