

Лабораторная работа №3

«Ветвление и слияние версий проекта *Git*»

Цель работы: научиться использовать операции ветвления и слияния версий проекта *Git*, рассмотреть примеры использования веток в рабочей среде.

Практическая часть

1. Операции ветвления

Ветвление в *Git* может использоваться для работы над какой-то новой функцией программы отдельно от той версии, которая идёт в массы. Используя ветвление, Вы отклоняетесь от основной линии разработки и продолжаете работу независимо от неё, не вмешиваясь в основную линию. Рассмотрим несколько основных команд, используемых при работе с ветвлением в *Git*.

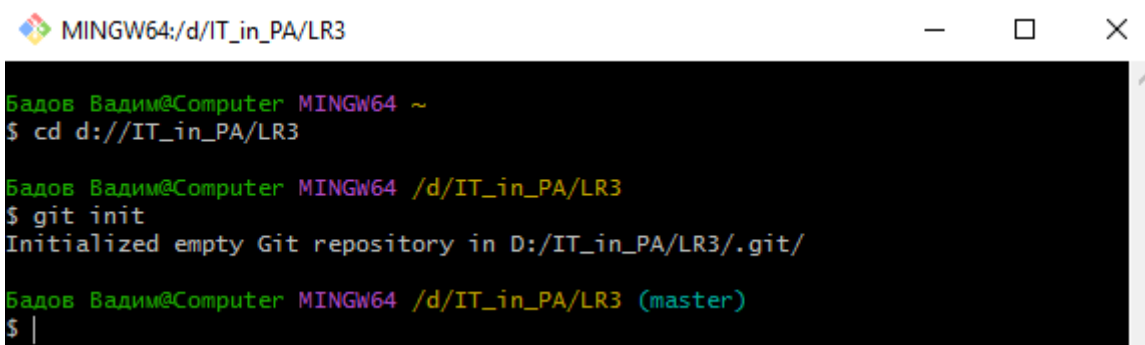
`$ git branch <имя_ветки>` – создание новой ветки.

`$ git checkout <имя_ветки>` – переключение между уже существующими ветками.

Задание:

1. Создайте в папке вашей бригады папку для выполнения данной лабораторной работы.
2. Перейдите в эту папку с помощью команды `$ cd <путь>`.

Инициализируйте репозиторий.

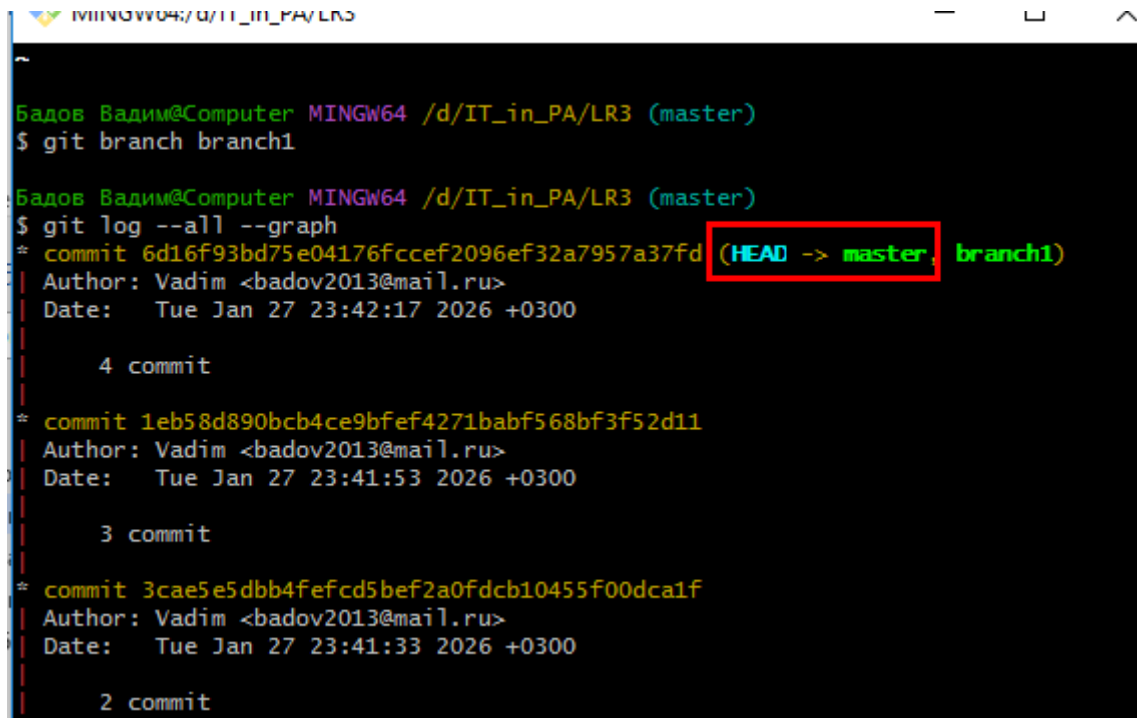


```
MINGW64:/d/IT_in_PA/LR3
Бадов Вадим@Computer MINGW64 ~
$ cd d://IT_in_PA/LR3
Бадов Вадим@Computer MINGW64 /d/IT_in_PA/LR3
$ git init
Initialized empty Git repository in D:/IT_in_PA/LR3/.git/
Бадов Вадим@Computer MINGW64 /d/IT_in_PA/LR3 (master)
$ |
```

3. Создайте текстовый файл и добавьте в него текст, сделайте коммит.
4. Измените этот файл несколько раз и создайте таким образом несколько коммитов (3 – 5). Проверьте с помощью `$ git log --all --graph`.

5. Теперь создайте новую ветвь, используя `$ git branch <имя_ветки>`.

Проверьте с помощью `$ git log --all --graph`.



```
Бадов Вадим@Computer MINGW64 /d/IT_in_PA/LR3 (master)
$ git branch branch1

Бадов Вадим@Computer MINGW64 /d/IT_in_PA/LR3 (master)
$ git log --all --graph
* commit 6d16f93bd75e04176fccef2096ef32a7957a37fd (HEAD -> master, branch1)
| Author: Vadim <badov2013@mail.ru>
| Date: Tue Jan 27 23:42:17 2026 +0300
|
| 4 commit
|
* commit 1eb58d890bcb4ce9bfef4271babf568bf3f52d11
| Author: Vadim <badov2013@mail.ru>
| Date: Tue Jan 27 23:41:53 2026 +0300
|
| 3 commit
|
* commit 3cae5e5dbb4fefcd5bef2a0fdcb10455f00dca1f
| Author: Vadim <badov2013@mail.ru>
| Date: Tue Jan 27 23:41:33 2026 +0300
|
| 2 commit
```

Обратите внимание на то, что при создании новой ветви мы всё ещё находимся на главной ветке (**HEAD -> master**).

6. Переключитесь на только что созданную ветку, используя

`$ git checkout <имя_ветки>`.

7. Проверяя историю версий теперь, вы должны увидеть

HEAD -> <имя_ветки>.

8. Находясь на новой ветке, создайте новый текстовый файл и добавьте в него текст. Сделайте коммит и проверьте историю версий с помощью `$ git log --all --graph`.

```

Бадов Вадим@Computer MINGW64 /d/IT_in_PA/LR3 (branch1)
$ git log --all --graph
* commit c3f2912d271c5cc9494eb181ae7671d253696d56 (HEAD -> branch1)
| Author: Vadim <badov2013@mail.ru>
| Date:   Wed Jan 28 00:20:06 2026 +0300
|
|     test2
|
* commit 6d16f93bd75e04176fccef2096ef32a7957a37fd (master)
| Author: Vadim <badov2013@mail.ru>
| Date:   Tue Jan 27 23:42:17 2026 +0300
|
|     4 commit
|
* commit 1eb58d890bcb4ce9bfef4271babf568bf3f52d11
| Author: Vadim <badov2013@mail.ru>
| Date:   Tue Jan 27 23:41:53 2026 +0300
|
|     3 commit
|
* commit 3cae5e5dbb4fefcd5bef2a0fdcb10455f00dca1f
| Author: Vadim <badov2013@mail.ru>
| Date:   Tue Jan 27 23:41:33 2026 +0300
|
|     2 commit

```

9. Допустим, что данная ветвь вам была нужна для работы над условной новой функцией вашей программы, а теперь вам срочно необходимо исправить баг, существующий на главной ветке (**master**). Для этого переключитесь на главную ветвь, используя `$ git checkout master`, измените первый текстовый файл и сделайте коммит.

```

linode@linode-pc MINGW64 /c/it_v_pd/laba_8 (branch1)
$ git checkout master
Switched to branch 'master'
M       file.txt

linode@linode-pc MINGW64 /c/it_v_pd/laba_8 (master)
$ git add .

linode@linode-pc MINGW64 /c/it_v_pd/laba_8 (master)
$ git commit -m "version4"
[master c31ac9f] version4
1 file changed, 1 insertion(+), 1 deletion(-)

```

10. Проверьте с помощью `$ git log --all --graph`.

```

linode@linode-pc MINGW64 /c/it_v_pd/laba_8 (master)
$ git log --all --graph
* commit c31ac9f3e1205aa497c8e92ab26f080fd37876cf (HEAD -> master)
  Author: Iskenderov Andrei <linode@duck.com>
  Date: Sat Apr 15 11:27:14 2023 +0300

    version4

* commit 04d2004379b44845cb080fdec8804e51a84eac0e (branch1)
  Author: Iskenderov Andrei <linode@duck.com>
  Date: Sat Apr 15 11:18:32 2023 +0300

    branch1 version1

* commit f7a68d1dd4d03aff8244ead7847a07d9976f6ce9
  Author: Iskenderov Andrei <linode@duck.com>
  Date: Sat Apr 15 10:51:27 2023 +0300

    version3

* commit b11b1e44a73b33c53ff7707b83ecd4f9c6efalae
  Author: Iskenderov Andrei <linode@duck.com>
  Date: Sat Apr 15 10:51:17 2023 +0300

    version2

* commit 7cbc857246b5ccddf1b8b165a0fccaa53029ab4f
  Author: Iskenderov Andrei <linode@duck.com>
  Date: Sat Apr 15 10:51:05 2023 +0300

    version1

```



Должно получиться так, что созданная ранее вами ветка как бы ушла в сторону от главной ветки.

2. Операция слияния

Слияние в *Git* используется для того, чтобы добавить все изменения из одной ветки в другую.

`$ git merge <имя_ветки> -m "сообщение"` — слияние указанной ветки с той, на которой вы находитесь.

Задание:

Примечание: данное задание выполняется в проекте задания по операциям ветвления.

1. Переключитесь на ранее созданную ветку. Чтобы узнать то, на какой ветке вы находитесь, достаточно посмотреть на бирюзовую надпись справа от пути.

```
linode@linode-pc MINGW64 /c/it_v_pd/lab8 (master)
$ git checkout branch1
Switched to branch 'branch1'

linode@linode-pc MINGW64 /c/it_v_pd/lab8 (branch1)
$ |
```

2. Измените текстовый файл, который вы создавали ранее на новой ветке и сделайте ещё один коммит.
3. Допустим, что вы закончили работать над той функцией, для которой вы создавали эту ветвь, и теперь вы хотите соединить её с главной веткой. Так как команда слияния переносит изменения из другой ветки в ту, над которой мы работаем сейчас, вам необходимо перейти назад на ветвь `master` с помощью `$ git checkout master`. После напишите `$ git merge <имя_ветки> -m "сообщение"` и проверьте с помощью `$ git log --all --graph`.

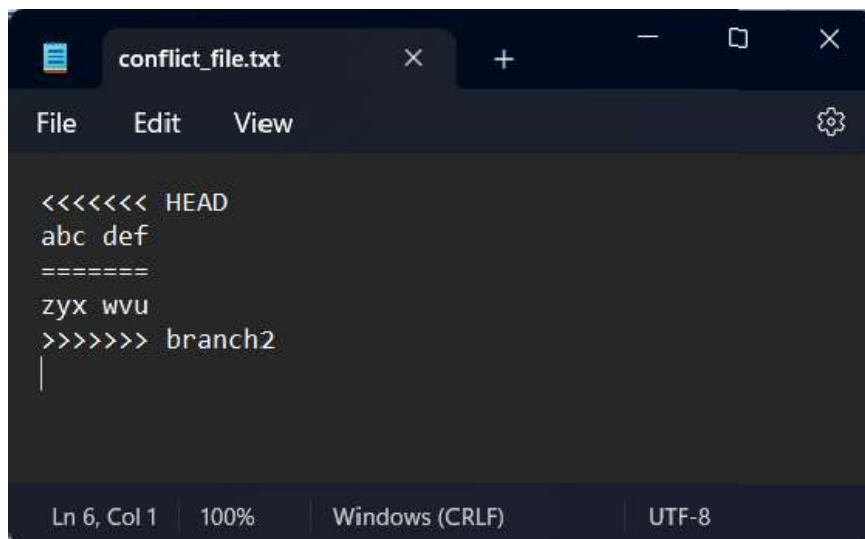
```
linode@linode-pc MINGW64 /c/it_v_pd/lab8 (master)
$ git merge branch1 -m "merge branch1"
Merge made by the 'ort' strategy.
 branch1_file.txt | 1 +
 1 file changed, 1 insertion(+)
 create mode 100644 branch1_file.txt

linode@linode-pc MINGW64 /c/it_v_pd/lab8 (master)
$ git log --all --graph
*   commit c800e393e14d2307cb0241d4f923888fa06a3730 (HEAD -> master)
| \ Merge: c31ac9f 0747417
|  | Author: Iskenderov Andrei <linode@duck.com>
|  | Date:   Sat Apr 15 16:33:41 2023 +0300
|  |
|  |     merge branch1
|
| *   commit 0747417f030c558f6bbd94dca8bf16402177fcc7 (branch1)
|  | Author: Iskenderov Andrei <linode@duck.com>
|  | Date:   Sat Apr 15 12:24:47 2023 +0300
|  |
|  |     branch1 version2
|
| *   commit 04d2004379b44845cb080fdec8804e51a84eac0e
|  | Author: Iskenderov Andrei <linode@duck.com>
|  | Date:   Sat Apr 15 11:18:32 2023 +0300
|  |
|  |     branch1 version1
|
| *   commit c31ac9f3e1205aa497c8e92ab26f080fd37876cf
|  | Author: Iskenderov Andrei <linode@duck.com>
|  | Date:   Sat Apr 15 11:27:14 2023 +0300
|  |
|  |     version4
|
| *   commit f7a68d1dd4d03aff8244ead7847a07d9976f6ce9
|  | Author: Iskenderov Andrei <linode@duck.com>
```

4. Рассмотрим ситуацию конфликта при слиянии двух веток, когда *Git* не знает, какое изменение ему принять. Такая ситуация может случиться, когда в одном файле одна и та же строка была изменена по разному в двух ветвях. Для этого создайте новый текстовый файл, напишите в него текст и сделайте коммит.
5. Создайте новую ветку. Пока вы ещё не перешли на неё, измените этот файл на ветке **master** и сделайте коммит, после этого перейдите на новую ветку. Измените тот же файл на той же строке и снова сделайте коммит.
6. Перейдём назад на ветку **master** и попробуем соединить другую ветвь с ней с помощью `$ git merge <имя_ветки> -m "сообщение"`. Вы должны получить вот такое сообщение:

```
linode@linode-pc MINGW64 /c/it_v_pd/laba_8 (master)
$ git merge branch2 -m "merge conflict"
Auto-merging conflict_file.txt
CONFLICT (content): Merge conflict in conflict_file.txt
Automatic merge failed; fix conflicts and then commit the result.
```

Если вы откроете тот файл, с которым у вас появились проблемы, то увидите следующее:



Тут *Git* показывает вам то, где именно образовался конфликт, и в чём он состоит. В данном случае “abc def” – версия этой строки из данной ветки (HEAD), а “zyx wvu” – версия этой строки из ветки, с которой вы пытаетесь слиться (branch2).

7. Чтобы решить этот конфликт, сотрите сообщение *Git'a* в текстовом файле и напишите на данной строке то, что вы хотели бы оставить.

Разрешите конфликт: Вам нужно удалить все маркеры **Git** (<<<<<<, =====, >>>>>>) и оставить в файле тот вариант текста, который должен быть итоговым. У вас есть три основных варианта:

- Оставить вариант из HEAD (верхний).
- Оставить вариант из ветки branch_name (нижний).
- Написать совершенно новый текст, который объединяет смысл или является третьим вариантом.

Это может быть или “abc def”, или “zyx wvu”, или даже какой-то новый текст. Сделайте коммит и проверьте историю версий.

```
linode@linode-pc MINGW64 /c/it_v_pd/laba_8 (master|MERGING)
$ git add .

linode@linode-pc MINGW64 /c/it_v_pd/laba_8 (master|MERGING)
$ git commit -m "merge conflict resolving"
[master ef3189c] merge conflict resolving

linode@linode-pc MINGW64 /c/it_v_pd/laba_8 (master)
$ git log --all --graph
*   commit ef3189c4cc864d62c2863c7d7c5af776997aa06a (HEAD -> master)
  | Merge: labe038 63947ef
  | Author: Iskenderov Andrei <linode@duck.com>
  | Date: Sat Apr 15 17:15:56 2023 +0300
  |
  |     merge conflict resolving
  |
  | *   commit 63947ef3ccf580350169607c89403a193ca6316e (branch2)
  |   | Author: Iskenderov Andrei <linode@duck.com>
  |   | Date: Sat Apr 15 17:04:55 2023 +0300
  |   |
  |   |     aa1
```

Как вы можете видеть по истории версий, слияние завершилось само с собой после коммита. Вы вручную приняли решение о том, какое содержимое должно остаться в проекте, тем самым **разрешив конфликт**. Это стандартная и очень важная процедура в командной работе.

Требования к отчету:

Титульный лист

Отчет должен содержать все введенные команды, ответы системы, пояснения, а также то, что было сделано между пунктами (например, отредактирован файл).