

**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 программной инженерии»

Выполнил студент группы ПИЖ-б-о-21-1

Образцова М. Д. .«08» октября 2022г.

Подпись студента _____

Работа защищена « » _____ 20__ г.

Проверил Воронкин Р.А. _____

(подпись)

Ставрополь 2022

Работа с консолью Git

1) Создание файлов 1,2,3, индексация первого файла и коммит с комментарием "add 1.txt file", индексация второго и третьего файла, перезапись уже сделанного коммита с новым комментарием "add 2.txt and 3.txt." показана на рисунках 1,2.

```
student-09-525@vdi-progr-024 MINGW64 /F/3 семестр/ОПИ/ОПИ_LR_3 (main)
$ git add 1.txt
git
student-09-525@vdi-progr-024 MINGW64 /F/3 семестр/ОПИ/ОПИ_LR_3 (main)
$ git commit -m "add 1.txt file"
[main f4e0e3e] add 1.txt file
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 1.txt
```

Рисунок 1– индексация первого файла

```
student-09-525@vdi-progr-024 MINGW64 /F/3 семестр/ОПИ/ОПИ_LR_3 (main)
$ git add 3.txt

student-09-525@vdi-progr-024 MINGW64 /F/3 семестр/ОПИ/ОПИ_LR_3 (main)
$ git commit -m "add 1.txt file"
[main 5f16ac8] add 1.txt file
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 3.txt

student-09-525@vdi-progr-024 MINGW64 /F/3 семестр/ОПИ/ОПИ_LR_3 (main)
$ git status
On branch main
Your branch is ahead of 'origin/main' by 1 commit.
(use "git push" to publish your local commits)

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

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

student-09-525@vdi-progr-024 MINGW64 /F/3 семестр/ОПИ/ОПИ_LR_3 (main)
$ git add 2.txt
git comm
student-09-525@vdi-progr-024 MINGW64 /F/3 семестр/ОПИ/ОПИ_LR_3 (main)
$ git commit --amend
[main d445d45] add 2.txt and 3.txt
Date: Sat Oct 8 10:30:41 2022 +0300
2 files changed, 0 insertions(+), 0 deletions(-)
create mode 100644 2.txt
create mode 100644 3.txt
```

Рисунок 2– индексация 2 и 3 файлов

2) Создание новой ветки `my_first_branch` показано на рисунке 3.

```
student-09-307@vdi-progr-012 MINGW64 /F/3 семестр/ОПИ/ОПИ_LR_3 (main)
$ git branch my_first_branch

student-09-307@vdi-progr-012 MINGW64 /F/3 семестр/ОПИ/ОПИ_LR_3 (main)
$ git log
commit f4e0e3ed3f19fd92e8566c053f971852a8fb43b4 (HEAD -> main, my_first_branch)
Author: obrMaria <midsummer.madness@mail.ru>
Date: Sat Oct 8 10:48:31 2022 +0300

    add 1.txt file

commit d445d455d22cc11bc7ab5993dfca6d46db1f9132
Author: obrMaria <midsummer.madness@mail.ru>
Date: Sat Oct 8 10:30:41 2022 +0300

    add 2.txt and 3.txt

commit 29b2ac81cef087614418db9bbf72ca36754c1a1f (origin/main, origin/HEAD)
Author: obrMaria <112749225+obrMaria@users.noreply.github.com>
Date: Sat Oct 8 10:05:17 2022 +0300

    Initial commit

student-09-307@vdi-progr-012 MINGW64 /F/3 семестр/ОПИ/ОПИ_LR_3 (main)
$ git log --oneline --decorate
f4e0e3e (HEAD -> main, my_first_branch) add 1.txt file
d445d45 add 2.txt and 3.txt
29b2ac8 (origin/main, origin/HEAD) Initial commit
```

Рисунок 3 – Результат создания новой ветки `my_first_branch`;
просмотр указателей на ветки `Git log --oneline --decorate`

3) Переход на ветку и создание нового файла `in_branch.txt`, коммит изменений показан на рисунке 4.

```
student-09-307@vdi-progr-012 MINGW64 /F/3 семестр/ОПИ/ОПИ_LR_3 (main)
$ git checkout my_first_branch
Switched to branch 'my_first_branch'
```

Рисунок 4–Переход на новую ветку с помощью команды `git checkout`

```

student-09-307@vdi-progr-012 MINGW64 /F/3 семестр/ОПИ/ОПИ_LR_3 (my_first_branch)
$ touch in_branch.txt

student-09-307@vdi-progr-012 MINGW64 /F/3 семестр/ОПИ/ОПИ_LR_3 (my_first_branch)
$ ls
1.txt 2.txt 3.txt in_branch.txt LICENSE README.md

student-09-307@vdi-progr-012 MINGW64 /F/3 семестр/ОПИ/ОПИ_LR_3 (my_first_branch)
$ git add in_branch
fatal: pathspec 'in_branch' did not match any files

student-09-307@vdi-progr-012 MINGW64 /F/3 семестр/ОПИ/ОПИ_LR_3 (my_first_branch)
$ git add in_branch.txt

student-09-307@vdi-progr-012 MINGW64 /F/3 семестр/ОПИ/ОПИ_LR_3 (my_first_branch)
$ git commit -a -m "first comiit in new branch"
[my_first_branch eddef0f] first comiit in new branch
Committer: student-09-307 <student-09-307@ncfu.net>
Your name and email address were configured automatically based
on your username and hostname. Please check that they are accurate.
You can suppress this message by setting them explicitly. Run the
following command and follow the instructions in your editor to edit
your configuration file:

    git config --global --edit

After doing this, you may fix the identity used for this commit with:

    git commit --amend --reset-author

1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 in_branch.txt

```

Рисунок 5— Создание нового файла в новой ветке

```

1@DESKTOP-A8I4D4M MINGW64 ~/desktop/OPI_LR_3 (my_first_branch)
$ 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)

1@DESKTOP-A8I4D4M MINGW64 ~/desktop/OPI_LR_3 (main)

```

Рисунок 6 – Результат возвращения на ветку main

4) Создание и переход сразу на ветку new_branch показан на рисунке 7.

```

1@DESKTOP-A8I4D4M MINGW64 ~/desktop/OPI_LR_3 (main)
$ git checkout -b new_branch
Switched to a new branch 'new_branch'

```

Рисунок 7— Результат создания и перехода на новую ветку
с помощью команды Git branch -b

5) Сделанные изменения в файле 1.txt, добавление строки “new row in the 1.txt file” показано на рисунке 8

```
l@DESKTOP-A8I4D4M MINGW64 ~/desktop/OPI_LR_3 (new_branch)
$ git commit -m "commit in new branch"
[new_branch 10d51de] commit in new branch
1 file changed, 1 insertion(+)
```

Рисунок 8 – Результат создания и перехода на ветку new_branch

6) Переход на ветку main и слияние ветки main и my_first_branch, после чего слияние ветки main и new_branch показано на рисунках 9,10.

```
$ git merge my_first_branch
Updating 4d318a9..c92443f
Fast-forward
 in_branch.txt | 0
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 in_branch.txt
```

Рисунок 9 – Результат слияния веток main и my_first_branch

```
$ git merge new_branch
Merge made by the 'recursive' strategy.
1.txt | 1 +
1 file changed, 1 insertion(+)
```

Рисунок 10 – Результат слияния веток main и new_branch

7) Удаление веток my_first_branch и new_branch показано на рисунке 11.

```
$ git branch -d my_first_branch
Deleted branch my_first_branch
```

Рисунок 11 – Результат удаления веток

8) Создание веток branch_1 и branch_2 показано на рисунке 10.

```
git branch branch_1
git branch branch_2
```

Рисунок 10 – Результат создания веток

9) Переход на ветки и изменения файлов с последующим коммитом показаны на рисунках 12,13.

```
M@DESKTOP-UVM9NOL MINGW64 ~/desktop/OPI_LR_3 (branch_1)
$ git add .

M@DESKTOP-UVM9NOL MINGW64 ~/desktop/OPI_LR_3 (branch_1)
$ git commit -m "изменила 1 и 3 файлы"
[branch_1 a650efa] изменила 1 и 3 файлы
2 files changed, 2 insertions(+), 1 deletion(-)
```

Рисунок 12– результатов коммитов на ветке branch_1

```
M@DESKTOP-UVM9NOL MINGW64 ~/desktop/OPI_LR_3 (branch_2)
$ git add .

M@DESKTOP-UVM9NOL MINGW64 ~/desktop/OPI_LR_3 (branch_2)
$ git commit -m "изменены 1.txt и 3.txt"
[branch_2 17a1431] изменены 1.txt и 3.txt
2 files changed, 2 insertions(+), 1 deletion(-)
```

Рисунок 13– результатов коммитов на ветке branch_2

10) Слияние изменения ветки branch_2 в ветку branch_1 показано на рисунке 14.

```
M@DESKTOP-UVM9NOL MINGW64 ~/desktop/OPI_LR_3 (branch_2)
$ 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.

M@DESKTOP-UVM9NOL MINGW64 ~/desktop/OPI_LR_3 (branch_2|MERGING)
$
```

Рисунок 14 – Результат слияния веток

11) Решение конфликтов при слиянии веток показано на рисунках 15-17

```

M@DESKTOP-UVM9NOL MINGW64 ~/desktop/OPI_LR_3 (branch_2|MERGING)
$ git status
On branch branch_2
You have unmerged paths.
  (fix conflicts and run "git commit")
  (use "git merge --abort" to abort the merge)

Unmerged paths:
  (use "git add <file>..." to mark resolution)
    both modified:   1.txt
    both modified:   3.txt

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

M@DESKTOP-UVM9NOL MINGW64 ~/desktop/OPI_LR_3 (branch_2|MERGING)
$ git add 1.txt

M@DESKTOP-UVM9NOL MINGW64 ~/desktop/OPI_LR_3 (branch_2|MERGING)
$ git status
On branch branch_2
You have unmerged paths.
  (fix conflicts and run "git commit")
  (use "git merge --abort" to abort the merge)

Changes to be committed:
  modified:   1.txt

Unmerged paths:
  (use "git add <file>..." to mark resolution)
    both modified:   3.txt

```

Рисунок 15 – Результат решения конфликтов git merge

```

M@DESKTOP-UVM9NOL MINGW64 ~/desktop/OPI_LR_3 (branch_2|MERGING)
$ 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:
opendiff kdiff3 tkdiff xxdiff meld tortoisemerge gvimdiff diffuse diffmerge ecme
rge p4merge araxis bc codecompare smerge 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– команда git mergetool

```
MINGW64:/c/Users/M/desktop/OPI_LR_3

my fix in the 3.txt
fix in the 3.txt

<(01:52 11/10/2022)1,1 A11 <(01:52 11/10/2022)0,1 A11 <(01:52 11/10/2022)1,1 A11

my fix in the 3.txt
fix in the 3.txt

3.txt[+] [dos] (01:52 11/10/2022) 5,1 A11
-- INSERT --

M@DESKTOP-UVM9NOL MINGW64 /C/users/M/desktop/OPI_LR_3 (branch_2|MERGING)
$ git commit -m "add 1.txt and 3.txt"
[branch_2 087c0ec] add 1.txt and 3.txt

M@DESKTOP-UVM9NOL MINGW64 /C/users/M/desktop/OPI_LR_3 (branch_2)
$
```

Рисунок 17 – Результат решения конфликтов

12) Отправление ветки branch_1 на GitHub показано на рисунке 15.


```

M@DESKTOP-UVM9NOL MINGW64 /C:/users/M/desktop/OPI_LR_3 (branch_2)
$ git push origin branch_2
Enumerating objects: 18, done.
Counting objects: 100% (16/16), done.
Delta compression using up to 4 threads
Compressing objects: 100% (6/6), done.
Writing objects: 100% (9/9), 972 bytes | 486.00 KiB/s, done.
Total 9 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (2/2), done.
remote:
remote: Create a pull request for 'branch_2' on GitHub by visiting:
remote:   https://github.com/obrMaria/OPI_LR_3/pull/new/branch_2
remote:
To https://github.com/obrMaria/OPI_LR_3.git
 * [new branch]      branch_2 -> branch_2

```

Рисунок 18 – Результат отправления ветки

13) Создание средствами GitHub удаленной ветки branch_3 показано на рисунке 19.

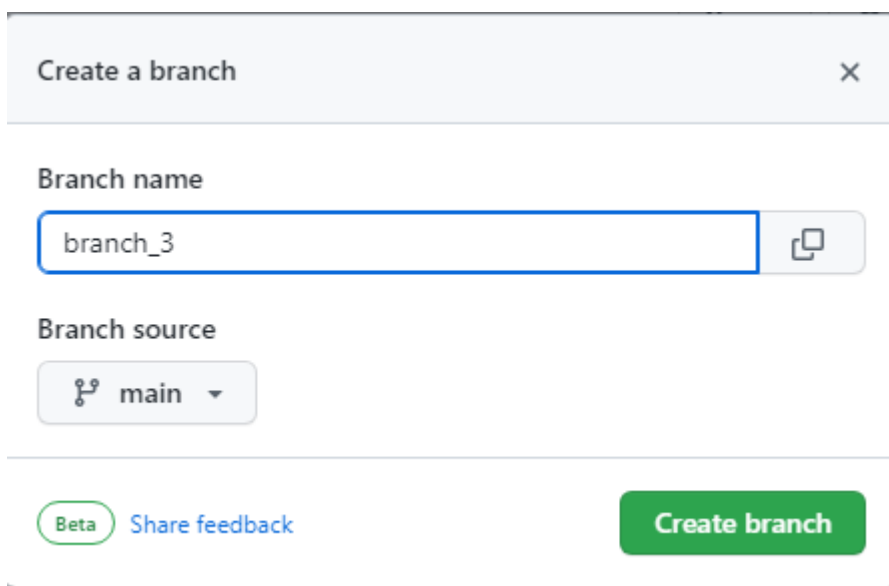


Рисунок 19 – Результат создания ветки

```

M@DESKTOP-UVM9NOL MINGW64 /C:/users/M/desktop/OPI_LR_3 (branch_2)
$ git fetch --all
From https://github.com/obrMaria/OPI_LR_3
 * [new branch]      branch_3 -> origin/branch_3

```

Рисунок 20–получение данных со всех удалённых серверов

14) Переход на ветку branch_3 и добавить файл файл 2.txt строку "the final fantasy in the 4.txt file" показано на рисунке 22.

```
M@DESKTOP-UVM9NOL MINGW64 /C:/users/M/desktop/OPI_LR_3 (main)
$ git merge branch_2
Updating f8a92c0..087c0ec
Fast-forward
 1.txt | 3 ++-
 3.txt | 1 +
 2 files changed, 3 insertions(+), 1 deletion(-)
```

Рисунок 21 –Перемещение ветки мастер на branch_2

```
M@DESKTOP-UVM9NOL MINGW64 /C:/users/M/desktop/OPI_LR_3 (main)
$ git push origin main
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/obrMaria/OPI_LR_3.git
   29b2ac8..087c0ec  main -> main

M@DESKTOP-UVM9NOL MINGW64 /C:/users/M/desktop/OPI_LR_3 (main)
$ git push origin branch_3
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 4 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 375 bytes | 375.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/obrMaria/OPI_LR_3.git
   29b2ac8..090ec16  branch_3 -> branch_3
```

Рисунок 22– последний коммит на ветке branch_3

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Что такое ветка?

Ветка в Git — это простой перемещаемый указатель на один из таких коммитов. По умолчанию, имя основной ветки в Git — master..

2. Что такое HEAD?

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

3. Способы создания веток.

С помощью команды `git branch` и команды `git checkout -b`.

4. Как узнать текущую ветку?

Вы можете легко это увидеть при помощи простой команды `git log`, которая покажет вам куда указывают указатели веток.

5. Как переключаться между ветками?

Для переключения на существующую ветку выполните команду `git checkout <>`.

6. Что такое удаленная ветка?

Удалённые ссылки — это ссылки (указатели) в ваших удалённых репозиториях, включая ветки, теги и так далее.

7. Что такое ветка отслеживания?

Ветки отслеживания — это ссылки на определённое состояние удалённых веток. Это локальные ветки, которые нельзя перемещать.

8. Как создать ветку отслеживания?

Для синхронизации `git fetch origin`, а затем `git checkout --track origin/<название ветки>`.

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

Команда `git push origin <branch>`.

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

Команда `git fetch` получает с сервера все изменения, которых у вас ещё нет, но не будет изменять состояние вашей рабочей директории. Эта команда

просто получает данные и позволяет вам самостоятельно сделать слияние. Тем не менее, существует команда `git pull`, которая в большинстве случаев является командой `git fetch`, за которой непосредственно следует команда `git merge`.

11. Как удалить локальную и удаленную ветки?

Вы можете удалить ветку на удалённом сервере используя параметр `--delete` для команды `git push`. Для удаления ветки на сервере, выполните следующую команду: `git push origin --delete <branch>`. Для локальной `git branch -d <branch>`

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. В этом рабочем процессе для регистрации истории проекта вместо одной ветки `main` используются две ветки. В главной ветке `main` хранится официальная история релиза, а ветка разработки `develop` предназначена для объединения всех функций.

Когда в ветке `develop` оказывается достаточно функций для выпуска (или приближается назначенная дата релиза), от ветки `develop` создается ветка `release`. Создание этой ветки запускает следующий цикл релиза, и с этого момента новые функции добавить больше нельзя — допускается лишь исправление багов, создание документации и решение других задач, связанных с релизом. Когда подготовка к поставке завершается, ветка `release` сливается с `main` и ей присваивается номер версии. Кроме того, нужно выполнить ее слияние с веткой `develop`, в которой с момента создания ветки релиза могли возникнуть изменения.

Ветки сопровождения или исправления (hotfix) используются для быстрого внесения исправлений в рабочие релизы.