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

**Кафедра инфокоммуникаций**

# «Основы ветвления в Git»

**Отчёт по лабораторной работе № 1.3**

Выполнил студент группы ПИЖ-б-о-21-1

Гребенкин Е. А.____. «22» октября 2022г.

Подпись студента_____

Работа защищена «  »_____20__г.

Ставрополь 2022

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."

```
C:\Users\Kacca\Desktop\ОПИ\OPI_lw_3>git add 1.txt

C:\Users\Kacca\Desktop\ОПИ\OPI_lw_3>git commit -m "add 1.txt file"
[main d5cd09d] add 1.txt file
 Committer: KSAMU <KSAMU@rgd.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 1.txt

C:\Users\Kacca\Desktop\ОПИ\OPI_lw_3>
```

Рисунок 1 – индексация файла

```
C:\Users\Kacca\Desktop\ОПИ\OPI_lw_3>git add 2.txt 3.txt

C:\Users\Kacca\Desktop\ОПИ\OPI_lw_3>git commit -m "add 2.txt and 3.txt"
[main d1f8ce2] add 2.txt and 3.txt
 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

```
C:\Users\Kacca\Desktop\ОПИ\OPI_lw_3>git checkout -b my_first_branch
Switched to a new branch 'my_first_branch'

C:\Users\Kacca\Desktop\ОПИ\OPI_lw_3>git add in_branch.txt

C:\Users\Kacca\Desktop\ОПИ\OPI_lw_3>git -commit "add in_branch.txt file"
unknown option: -commit
usage: git [-v | --version] [-h | --help] [-C <path>] [-c <name>=<value>]
           [--exec-path[=<path>]] [--html-path] [--man-path] [--info-path]
           [-p | --paginate | -P | --no-pager] [--no-replace-objects] [--bare]
           [--git-dir=<path>] [--work-tree=<path>] [--namespace=<name>]
           [--super-prefix=<path>] [--config-env=<name>=<envvar>]
           <command> [<args>]

C:\Users\Kacca\Desktop\ОПИ\OPI_lw_3>git commit -m "add in_branch.txt file"
[my_first_branch c84ce24] add in_branch.txt file
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 in_branch.txt
```

Рисунок 3 – создание новой ветки и применение изменений в файле «in_branch.txt»

4) Создание и переход на ветку new_branch

```
C:\Users\Kacca\Desktop\ОПИ\OPI_lw_3>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)

C:\Users\Kacca\Desktop\ОПИ\OPI_lw_3>git checkout -b new_branch
Switched to a new branch 'new_branch'
```

Рисунок 4 – создание и переход на новую ветку

```
C:\Users\Kacca\Desktop\ОПИ\OPI_lw_3>git add 1.txt

C:\Users\Kacca\Desktop\ОПИ\OPI_lw_3>git commit -m "edit 1.txt file"
[new_branch a4cffd3] edit 1.txt file
 1 file changed, 1 insertion(+)

C:\Users\Kacca\Desktop\ОПИ\OPI_lw_3>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)
```

Рисунок 5 – коммит файла в ветку new_branch и переключение на ветку main

5) Слияние веток main, my_first_branch и new_branch

```
C:\Users\Kacca\Desktop\ОПИ\OPI_lw_3>git merge my_first_branch
Updating d1f8ce2..c84ce24
Fast-forward
 in_branch.txt | 0
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 in_branch.txt

C:\Users\Kacca\Desktop\ОПИ\OPI_lw_3>git merge new_branch
Merge made by the 'ort' strategy.
 1.txt | 1 +
 1 file changed, 1 insertion(+)
```

Рисунок 6 – слияние веток

```
C:\Users\Kacca\Desktop\ОПИ\OPI_lw_3>git branch --merged
* main
  my_first_branch
  new_branch

C:\Users\Kacca\Desktop\ОПИ\OPI_lw_3>git branch -d my_first_branch
Deleted branch my_first_branch (was c84ce24).

C:\Users\Kacca\Desktop\ОПИ\OPI_lw_3>git branch -d new_branch
Deleted branch new_branch (was a4cffd3).
```

Рисунок 7 – удаление слитых веток

6) Создание веток branch_1 и branch_2

```
C:\Users\Kacca\Desktop\ОПИ\OPI_lw_3>git branch branch_1

C:\Users\Kacca\Desktop\ОПИ\OPI_lw_3>git branch branch_2
```

Рисунок 8 – создание веток

```
C:\Users\Kacca\Desktop\ОПИ\OPI_lw_3>git checkout branch_1
Switched to branch 'branch_1'

C:\Users\Kacca\Desktop\ОПИ\OPI_lw_3>git add 1.txt 3.txt

C:\Users\Kacca\Desktop\ОПИ\OPI_lw_3>git commit -m "edit 1.txt and 3.txt files"
[branch_1 eabd413] edit 1.txt and 3.txt files
 2 files changed, 2 insertions(+), 1 deletion(-)
```

Рисунок 9 – коммит изменений в файлах 1 и 3на ветке branch_1

7) Слияние и исправление конфликтов

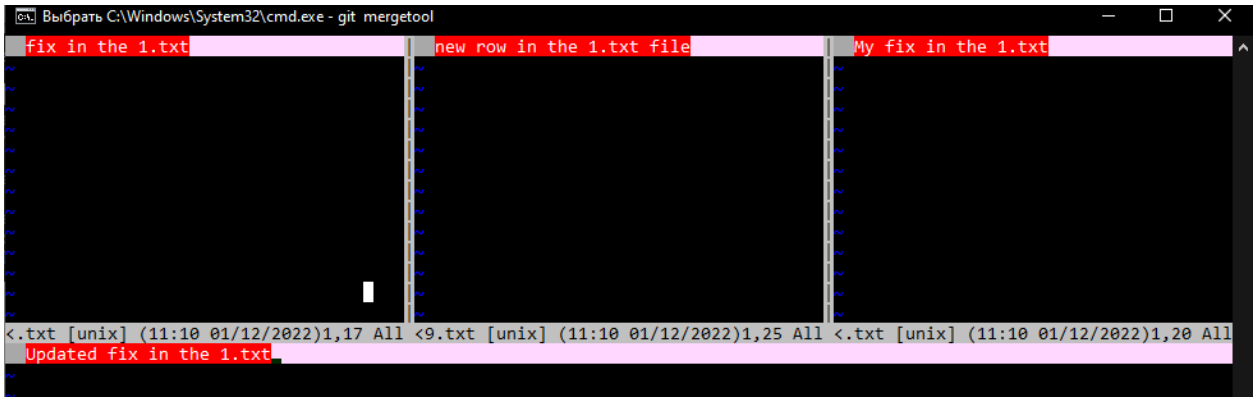


Рисунок 10 – исправление конфликта в файле 1

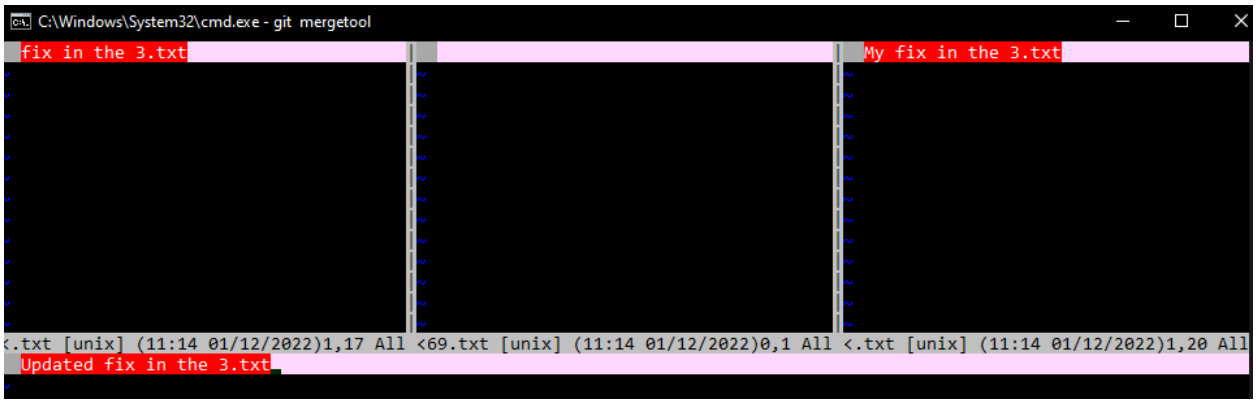


Рисунок 11 – исправление конфликта в файле 3

```
C:\Users\Kacca\Desktop\ОПИ\OPI_lw_3>git status
On branch 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

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        1.txt.orig
        3.txt.orig
        "~$\321\207\321\221\321\202 \320\277\320\276 3 \320\273\320\260\320\261\320\265.docx"
        "\320\236\321\202\321\207\321\221\321\202 \320\277\320\276 3 \320\273\320\260\320\261\320\265.docx"
```

Рисунок 12 – просмотр состояние после слияния

8) Создание ветки branch_3 средствами GitHub

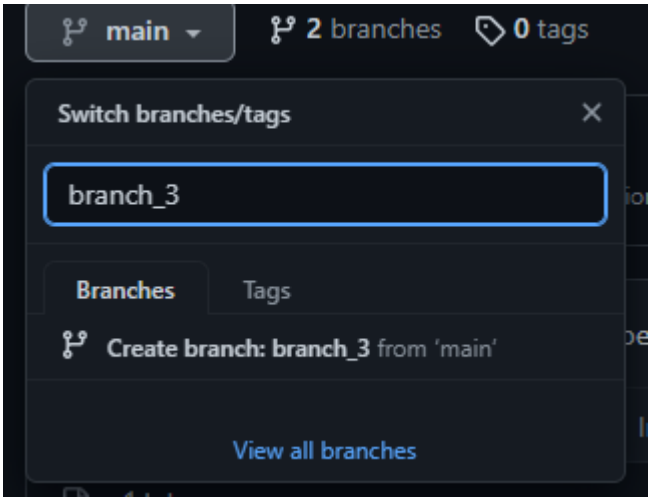


Рисунок 13 – создание новой ветки через пользовательский интерфейс

```
C:\Users\Kacca\Desktop\ОПИ\OPI_lw_3>git add .

C:\Users\Kacca\Desktop\ОПИ\OPI_lw_3>git commit -m "edit 2.txt file"
[branch_3 7fa6f36] edit 2.txt file

C:\Users\Kacca\Desktop\ОПИ\OPI_lw_3>git push --set-upstream origin branch_3
Enumerating objects: 17, 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), 1.02 KiB | 523.00 KiB/s, done.
Total 9 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/woeshiga/OPI_lw_3.git
   1260c7a..7fa6f36  branch_3 -> branch_3
branch 'branch_3' set up to track 'origin/branch_3'.
```

Рисунок 14 – коммит и пуш изменений в файле 2 на ветке branch_3

9) Пуш веток main и branch_2 на GitHub

```
C:\Users\Kacca\Desktop\ОПИ\OPI_lw_3>git push --set-upstream origin branch_2
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: Create a pull request for 'branch_2' on GitHub by visiting:
remote:      https://github.com/woeshiga/OPI_lw_3/pull/new/branch_2
remote:
To https://github.com/woeshiga/OPI_lw_3.git
 * [new branch]      branch_2 -> branch_2
branch 'branch_2' set up to track 'origin/branch_2'.
```

Рисунок 15 – отправка ветки branch_2 на GitHub

```
C:\Users\Kacca\Desktop\ОПИ\OPI_lw_3>git push
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 4 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 523 bytes | 523.00 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.
To https://github.com/woeshiga/OPI_lw_3.git
   5323b21..1f2c544  main -> main
```

Рисунок 15 – отправка ветки main на GitHub

## 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?

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

7. Что такое ветка отслеживания?

   Ветки отслежения — это ссылки на определѐнное состояние удалѐнных веток. Это локальные ветки, которые нельзя перемещать.

8. Как создать ветку отслеживания?

   Для синхранизации got fetch origin, а затем git checkout --track origin/.

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

   Команда git push origin .

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

    Команда git fetch получает с сервера все изменения, которых у вас ещѐ 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 удалѐнном сервере используя параметр --delete для команды git push . Для удаления ветки на сервере, выполните следующую команду: git push origin --delete . Для локальной git branch –d

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?

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.