

**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

Степанов Д.А. .«28» ноября 2022г.

Подпись студента _____

Работа защищена « » _____ 20__ г.

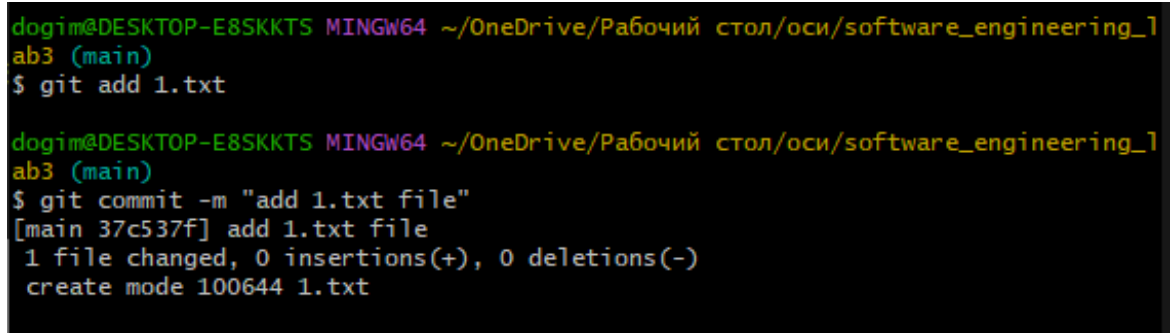
Проверил Воронкин Р.А. _____

(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.



```
dogim@DESKTOP-E8SKKTS MINGW64 ~/OneDrive/Рабочий стол/оси/software_engineering_1
ab3 (main)
$ git add 1.txt

dogim@DESKTOP-E8SKKTS MINGW64 ~/OneDrive/Рабочий стол/оси/software_engineering_1
ab3 (main)
$ git commit -m "add 1.txt file"
[main 37c537f] add 1.txt file
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 1.txt
```

Рисунок 1– индексация первого файла

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

```
dogim@DESKTOP-E8SKKTS MINGW64 ~/OneDrive/Рабочий стол/оси/software_engineering_lab3 (main)
$ git branch my_first_branch

dogim@DESKTOP-E8SKKTS MINGW64 ~/OneDrive/Рабочий стол/оси/software_engineering_lab3 (main)
$ git log
commit 0d85f6ca38a25530517b529cdf28a41d06350de3 (HEAD -> main, origin/main, origin/HEAD, my_first_branch)
Author: dima <dogima1@mail.ru>
Date: Mon Nov 28 20:29:34 2022 +0300

    add 1.txt file

commit 37c537f55b242b09d6c5cbff246ecfc893a38947
Author: dima <dogima1@mail.ru>
Date: Mon Nov 28 20:28:30 2022 +0300

    add 1.txt file

commit 8afb1fd2d6119afe511dcdbf203fe3546feaa67
Author: Dima <65311242+Lunyawa@users.noreply.github.com>
Date: Mon Nov 21 18:19:00 2022 +0300

    Update README.md

commit 96a0c9f81b6243f90b936b3870ed0daaf5e87a2f
Author: Dima <65311242+Lunyawa@users.noreply.github.com>
Date: Mon Nov 21 18:18:42 2022 +0300

    Initial commit

dogim@DESKTOP-E8SKKTS MINGW64 ~/OneDrive/Рабочий стол/оси/software_engineering_lab3 (main)
$ git log --oneline --decorate
0d85f6c (HEAD -> main, origin/main, origin/HEAD, my_first_branch) add 1.txt file
37c537f add 1.txt file
8afb1fd Update README.md
96a0c9f 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.

```
dogim@DESKTOP-E8SKKTS MINGW64 ~/OneDrive/Рабочий стол/оси/software_engineering_lab3 (main)
$ git checkout my_first_branch
Switched to branch 'my_first_branch'
```

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

```

dogim@DESKTOP-E8SKKTS MINGW64 ~/OneDrive/Рабочий стол/оси/software_engineering_lab
$ touch in_branch.txt

dogim@DESKTOP-E8SKKTS MINGW64 ~/OneDrive/Рабочий стол/оси/software_engineering_lab
$ ls
1.txt 2.txt 3.txt LICENSE README.md in_branch.txt

dogim@DESKTOP-E8SKKTS MINGW64 ~/OneDrive/Рабочий стол/оси/software_engineering_lab
$ git add in_branch.txt

dogim@DESKTOP-E8SKKTS MINGW64 ~/OneDrive/Рабочий стол/оси/software_engineering_lab
$ git commit -a -m "first commit in new branch"
[my_first_branch 02b57e4] first commit in new branch
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 in_branch.txt

dogim@DESKTOP-E8SKKTS MINGW64 ~/OneDrive/Рабочий стол/оси/software_engineering_lab
$ |

```

Рисунок 5– Создание нового файла в новой ветке

```

dogim@DESKTOP-E8SKKTS MINGW64 ~/OneDrive/Рабочий стол/оси/software_engineering_lab3 (my_first_branch)
$ git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.

```

Рисунок 6 – Результат возвращения на ветку main

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

```

dogim@DESKTOP-E8SKKTS MINGW64 ~/OneDrive/Рабочий стол/оси/software_engineering_lab3 (main)
$ git checkout -b new_branch
Switched to a new branch 'new_branch'

dogim@DESKTOP-E8SKKTS MINGW64 ~/OneDrive/Рабочий стол/оси/software_engineering_lab3 (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

```
1@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.

```
dogim@DESKTOP-E8SKKTS MINGW64 ~/OneDrive/Рабочий
$ git merge my_first_branch
Updating 0d85f6c..02b57e4
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

```
dogim@DESKTOP-E8SKKTS MINGW64 ~/OneDrive/Рабочий стол
$ git merge new_branch
Updating 0d85f6c..02b57e4
Fast-forward
 in_branch.txt | 0
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 in_branch.txt
```

Рисунок 10 – Результат слияния веток main и new_branch

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

```
dogim@DESKTOP-E8SKKTS MINGW64 ~/OneDrive/Рабочий стол/оси/software_engineering_lab3
$ git branch -d my_first_branch
Deleted branch my_first_branch (was 02b57e4).

dogim@DESKTOP-E8SKKTS MINGW64 ~/OneDrive/Рабочий стол/оси/software_engineering_lab3
$ git branch -d new_branch
Deleted branch new_branch (was 02b57e4).
```

Рисунок 11 – Результат удаления веток

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

```
dogim@DESKTOP-E8SKKTS MINGW64 ~/OneDrive/Рабочий стол/оси/software_engineering_lab3
$ git branch branch_1

dogim@DESKTOP-E8SKKTS MINGW64 ~/OneDrive/Рабочий стол/оси/software_engineering_lab3
$ git branch branch_2
```

Рисунок 10 – Результат создания веток

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

```
dogim@DESKTOP-E8SKKTS MINGW64 ~/OneDrive/Рабочий стол/оси/software_engineering_lab3 (branch_1)
$ git add .

dogim@DESKTOP-E8SKKTS MINGW64 ~/OneDrive/Рабочий стол/оси/software_engineering_lab3 (branch_1)
$ git commit -m "fix in the 1.txt and 3.txt"
[branch_1 8214fe1] fix in the 1.txt and 3.txt
2 files changed, 2 insertions(+)
```

Рисунок 12– результатов коммитов на ветке branch_1

```
dogim@DESKTOP-E8SKKTS MINGW64 ~/OneDrive/Рабочий стол/оси/software_engineering_lab3 (branch_2)
$ git add .

dogim@DESKTOP-E8SKKTS MINGW64 ~/OneDrive/Рабочий стол/оси/software_engineering_lab3 (branch_2)
$ git commit -m "My fix in the 1.txt and 3.txt"
[branch_2 1e2c146] My fix in the 1.txt and 3.txt
2 files changed, 2 insertions(+)
```

Рисунок 13– результатов коммитов на ветке branch_2

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

```
dogim@DESKTOP-E8SKKTS MINGW64 ~/OneDrive/Рабочий стол/оси/software_engineering_lab3 (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.

dogim@DESKTOP-E8SKKTS MINGW64 ~/OneDrive/Рабочий стол/оси/software_engineering_lab3 (branch_2|MERGING)
$ |
```

Рисунок 14 – Результат слияния веток

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

```

dogim@DESKTOP-E8SKKTS MINGW64 ~/OneDrive/Рабочий стол/оси/software_engineering_lab3 (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")

dogim@DESKTOP-E8SKKTS MINGW64 ~/OneDrive/Рабочий стол/оси/software_engineering_lab3 (branch_2|MERGING)
$ git add 1.txt

dogim@DESKTOP-E8SKKTS MINGW64 ~/OneDrive/Рабочий стол/оси/software_engineering_lab3 (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

```

dogim@DESKTOP-E8SKKTS MINGW64 ~/OneDrive/Рабочий стол/оси/software_engineering_lab3 (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 ecmerge p4merge araxis bc codecompare smerge em
erge 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

```

dogim@DESKTOP-E8SKKTS MINGW64 ~/OneDrive/Рабочий стол/оси/software_engineering_lab3 (branch_2|MERGING)
$ git commit -m "add 1.txt and 3.txt"
[branch_2 c065ad0] add 1.txt and 3.txt

```

Рисунок 17 – Результат решения конфликтов

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

```

dogim@DESKTOP-E8SKKTS MINGW64 ~/OneDrive/Рабочий стол/оси/software_engineering_lab3 (branch_2)
$ git push origin branch_2
Enumerating objects: 17, done.
Counting objects: 100% (16/16), done.
Delta compression using up to 8 threads
Compressing objects: 100% (10/10), done.
Writing objects: 100% (14/14), 1.20 KiB | 615.00 KiB/s, done.
Total 14 (delta 4), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (4/4), completed with 1 local object.
remote:
remote: Create a pull request for 'branch_2' on GitHub by visiting:
remote:   https://github.com/Lunyawa/software_engineering_lab3/pull/new/branch_2
remote:
To https://github.com/Lunyawa/software_engineering_lab3.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

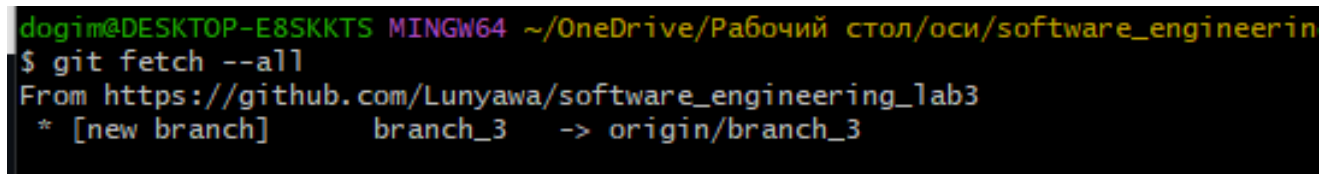


Рисунок 20–получение данных со всех удалённых серверов

14) Переход на ветку branch_3 и добавить файл файл 2.txt строку "the final fantasy in the 4.txt file" показано на рисунке 21.

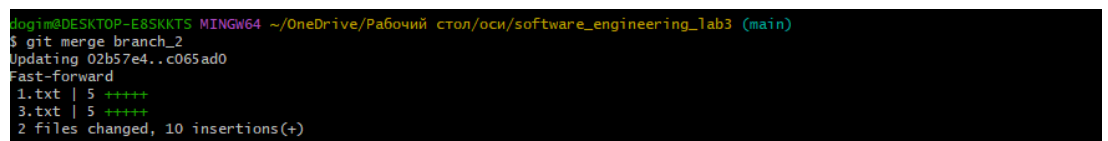


Рисунок 21 –Перемещение ветки мастер на branch_2

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.