

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ**

**Федеральное государственное автономное
образовательное учреждение высшего образования
«Северо-Кавказский федеральный университет»**

Кафедра инфокоммуникаций

**Отчет по лабораторной работе №3
«Основы ветвления Git»**

по дисциплине «Основы программной инженерии»

Выполнила:

Первых Дарья Александровна,
2 курс, группа ПИЖ-б-о-20-1,

Проверил:

Доцент кафедры инфокоммуникаций,
Воронкин Роман Александрович

Ставрополь 2021

ВЫПОЛНЕНИЕ

1. 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 рисунке 1.

```
C:\Users\BK201>git add 1.txt
fatal: not a git repository (or any of the parent directories): .git

C:\Users\BK201>cd laba3

C:\Users\BK201\laba3>git add 1.txt

C:\Users\BK201\laba3>git commit -m "add 1.txt file"
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)
      2.txt
      3.txt

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

C:\Users\BK201\laba3>git add 2.txt

C:\Users\BK201\laba3>git add 3.txt

C:\Users\BK201\laba3>git commit -m "add 2.txt and 3.txt"
[main 4d318a9] add 2.txt and 3.txt
 2 files changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 2.txt
 create mode 100644 3.txt
```

Рисунок 1 – Результат индексации и коммитов

2) Создание новой ветки my_first_branch показано на рисунке 2.

```
C:\Users\BK201\laba3>git branch my_first_branch

C:\Users\BK201\laba3>git checkout my_first_branch
Switched to branch 'my_first_branch'
```

Рисунок 2 – Результат создания новой ветки my_first_branch

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

```
C:\Users\BK201\laba3>git add in_branch.txt

C:\Users\BK201\laba3>git commit -m "add in_branch.txt"
[my_first_branch c92443f] add in_branch.txt
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 in_branch.txt
```

Рисунок 3 – Результат перехода на ветку и создание нового файла in_branch.txt

4) Возвращение на ветку main показано на рисунке 4.

```
C:\Users\BK201\laba3>git checkout -  
Switched to branch 'main'  
Your branch is ahead of 'origin/main' by 2 commits.  
(use "git push" to publish your local commits)
```

Рисунок 4 – Результат возвращения на ветку main

5) Создание и переход сразу на ветку new_branch показан на рисунке 5.

```
C:\Users\BK201\laba3>git checkout -b new_branch  
Switched to a new branch 'new_branch'
```

Рисунок 5 – Результат создания и перехода на ветку new_branch

6) Сделанные изменения в файле 1.txt, добавление строки “new row in the 1.txt file” показано на рисунке 6.

```
C:\Users\BK201\laba3>git commit -m "добавила тест в 1.txt"  
[new_branch 7b6eff9] добавила тест в 1.txt  
1 file changed, 1 insertion(+)
```

Рисунок 6 – Результат коммита после изменений

7) Переход на ветку master и слияние ветки main и my_first_branch, после чего слияние ветки main и new_branch показано на рисунках 7,8.

```
C:\Users\BK201\laba3>git checkout main  
Already on 'main'  
Your branch is up to date with 'origin/main'.  
  
C:\Users\BK201\laba3>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 7 – Результат слияния веток main и my_first_branch

```
C:\Users\BK201\laba3>git checkout main
Already on 'main'
Your branch is ahead of 'origin/main' by 1 commit.
  (use "git push" to publish your local commits)

C:\Users\BK201\laba3>git merge new_branch
Merge made by the 'recursive' strategy.
 1.txt | 1 +
 1 file changed, 1 insertion(+)
```

Рисунок 8 – Результат слияния веток main и new_branch

8) Удаление веток my_first_branch и new_branch показано на рисунке 9.

```
C:\Users\BK201\laba3>git branch -d my_first_branch
Deleted branch my_first_branch (was c92443f).

C:\Users\BK201\laba3>git branch -d new_branch
Deleted branch new_branch (was 7b6eff9).
```

Рисунок 9 – Результат удаления веток

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

```
C:\Users\BK201\laba3>git branch branch_1

C:\Users\BK201\laba3>git branch branch_2
```

Рисунок 10 – Результат создания веток

10) Переход на ветки и изменения файлов с последующим коммитом показаны на рисунке 11.


```

C:\Users\BK201\laba3>git checkout branch_1
Switched to branch 'branch_1'

C:\Users\BK201\laba3>git add .

C:\Users\BK201\laba3>git commit -m "изменила 1.txt и 3.txt"
[branch_1 858aff3] изменила 1.txt и 3.txt
 2 files changed, 2 insertions(+), 1 deletion(-)

C:\Users\BK201\laba3>git checkout branch_2
Switched to branch 'branch_2'

C:\Users\BK201\laba3>git add .

C:\Users\BK201\laba3>git commit -m "изменены файлы 1 и 3"
[branch_2 d4c29ad] изменены файлы 1 и 3
 2 files changed, 2 insertions(+), 1 deletion(-)

```

Рисунок 11 – Результат коммитов

11) Слияние изменения ветки branch_2 в ветку branch_1 показано на рисунке 12.

```

C:\Users\BK201\laba3>git checkout branch_2
Already on 'branch_2'

C:\Users\BK201\laba3>git merge branch_1
Auto-merging 3.txt
CONFLICT (content): Merge conflict in 3.txt
Auto-merging 1.txt
CONFLICT (content): Merge conflict in 1.txt
Automatic merge failed; fix conflicts and then commit the result.

```

Рисунок 12 – Результат слияния веток

12) Решение конфликтов при слиянии веток показано на рисунках 13,14

```

C:\Users\BK201\laba3>git status
On branch branch_2
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   1.txt
        modified:   3.txt

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

C:\Users\BK201\laba3>git add 1.txt

C:\Users\BK201\laba3>git status
On branch branch_2
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   1.txt

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   3.txt

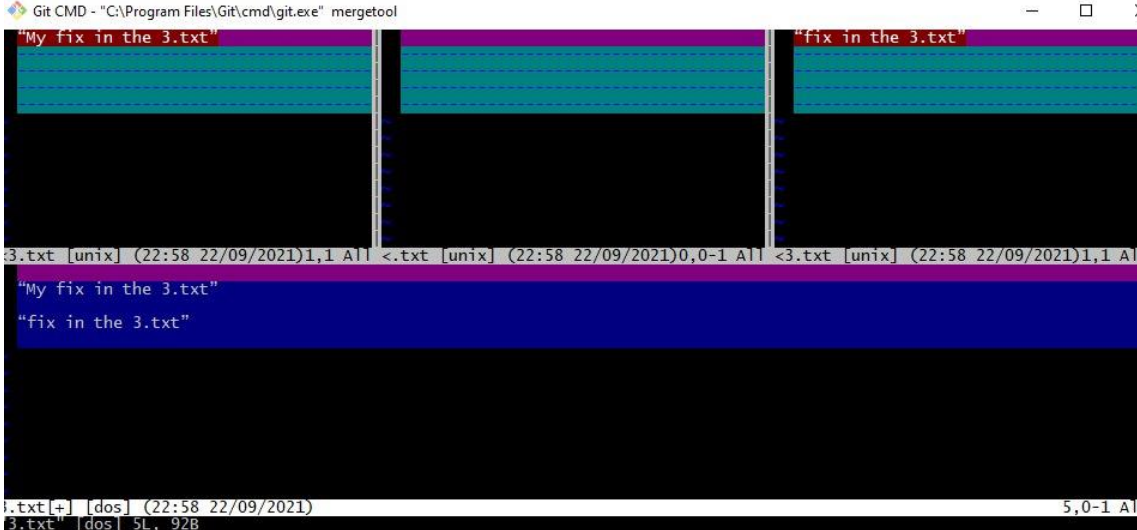
```

Рисунок 13 – Результат решения конфликтов

```
C:\Users\BK201\laba3>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:
tortoisemerge 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): _
```



```
C:\Users\BK201\laba3>git commit -m "add 1.txt file and 3.txt file"
[branch_2 9fb21ff] add 1.txt file and 3.txt file

C:\Users\BK201\laba3>git merge branch_1
Already up to date.
```

Рисунок 14 – Результат решения конфликтов

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

```
C:\Users\BK201\laba3>git push origin branch_1
Enumerating objects: 17, done.
Counting objects: 100% (17/17), done.
Delta compression using up to 4 threads
Compressing objects: 100% (12/12), done.
Writing objects: 100% (16/16), 1.40 KiB | 357.00 KiB/s, done.
Total 16 (delta 6), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (6/6), done.
remote:
remote: Create a pull request for 'branch_1' on GitHub by visiting:
remote:   https://github.com/PervykhDarya/laba3/pull/new/branch_1
remote:
To https://github.com/PervykhDarya/laba3.git
 * [new branch]      branch_1 -> branch_1
```

Рисунок 15 – Результат отправления ветки

14) Создание средствами GitHub удаленной ветки branch_3 показано на рисунке 16.

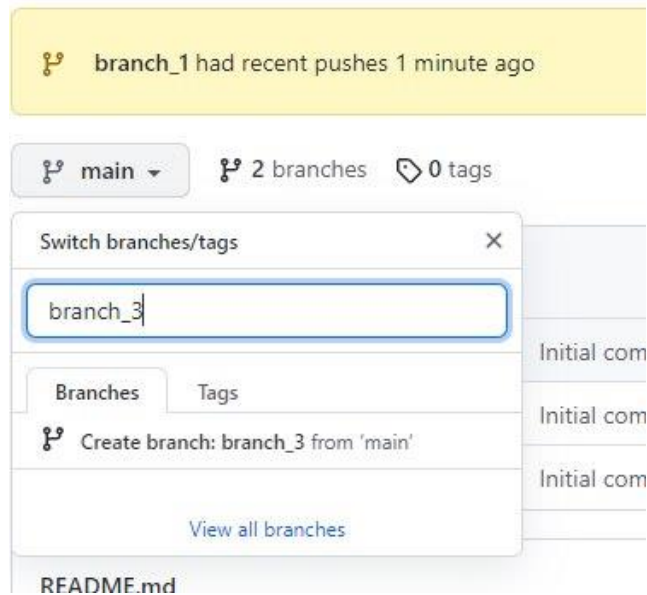


Рисунок 16 – Результат создания ветки

15) Создание в локальном репозитории ветку отслеживания удаленной ветки branch_3 показано на рисунке 17.

```
C:\Users\BK201\laba3>git checkout --track origin/branch_3
Switched to a new branch 'branch_3'
Branch 'branch_3' set up to track remote branch 'branch_3' from 'origin'.
```

Рисунок 17 – Результат создания ветки слежения

16) Переход на ветку branch_3 и добавить файл файл 2.txt строку "the final fantasy in the 4.txt file" показано на рисунке 18.

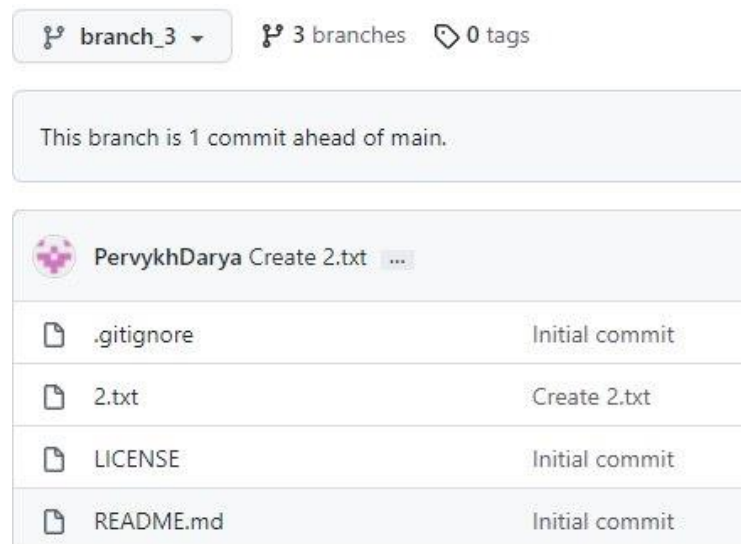


Рисунок 18 – Результат добавления файла

17) Отправление изменений ветки branch_2 на GitHub показано на рисунке 19.

```
C:\Users\BK201\laba3>git push origin branch_2
Enumerating objects: 14, done.
Counting objects: 100% (14/14), done.
Delta compression using up to 4 threads
Compressing objects: 100% (6/6), done.
Writing objects: 100% (8/8), 729 bytes | 243.00 KiB/s, done.
Total 8 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (2/2), completed with 1 local object.
remote:
remote: Create a pull request for 'branch_2' on GitHub by visiting:
remote:   https://github.com/PervykhDarya/laba3/pull/new/branch_2
remote:
To https://github.com/PervykhDarya/laba3.git
 * [new branch]      branch_2 -> branch_2
```

Рисунок 19 – Результат отправления изменений

Ответы на 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. Как только вы начнёте создавать коммиты, ветка master будет всегда указывать на последний коммит. Каждый раз при создании коммита указатель ветки master будет передвигаться на следующий коммит автоматически.

2. Что такое HEAD?

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

3. Способы создания веток.

Вы можете это сделать командой `git branch`. Команда `git branch` только создаёт новую ветку, но не переключает на неё.

4. Как узнать текущую ветку?

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

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 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. Представляйте их как закладки для напоминания о том, где ветки в удалённых репозиториях находились во время последнего подключения к ним.

8. Как создать ветку отслеживания?

Ветки слежения — это локальные ветки, которые напрямую связаны с удалённой веткой. `git checkout --track origin/serverfix`

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

Если у вас есть ветка `serverfix`, над которой вы хотите работать с кем-то ещё, вы можете отправить её точно так же, как вы отправляли вашу первую ветку. Выполните команду `git push <remote> <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`. Если у вас настроена ветка слежения как показано в предыдущем разделе, или она явно установлена, или она была создана автоматически командами `clone` или `checkout`, `git pull` определит сервер и ветку, за которыми следит ваша текущая ветка, получит данные с этого сервера и затем попытается слить удалённую ветку.

11. Как удалить локальную и удаленную ветки?

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

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.

