

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ  
ФЕДЕРАЦИИ ФГАОУ ВО «СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ  
УНИВЕРСИТЕТ»

Институт математики и информационных технологии имени  
профессора Н.И. Червякова

Кафедра инфокоммуникаций

**ОТЧЕТ**  
**по лабораторной работе №1**  
Дисциплина: «Языки программирования»

Выполнил: студент 2 курса  
группы ИТС-б-о-20-1  
Новомлинов Алексей Сергеевич  
Проверил доцент  
к.т.н., доцент  
Кафедры инфокоммуникаций  
Воронкин Роман Александрович

Работа защищена с оценкой: \_\_\_\_\_

Ставрополь, 2021

# ЛАБОРАТОРНАЯ РАБОТА №1

## Основы ветвления Git

**Цель:** исследование базовых возможностей по работе с локальными и удаленными ветками Git.

### Выполнение работы:

Создадим общедоступный репозиторий на GitHub, в котором будет использована лицензия MIT.

[https://github.com/Novomlinov/2C\\_Lab\\_1](https://github.com/Novomlinov/2C_Lab_1)

Создадим три файла: 1.txt, 2.txt, 3.txt. Проиндексируем первый файл и сделаем коммит с комментарием «add 1.txt file».

```
h:\Учеба\Программирование\lab_1\1s\2C_Lab_1>git add 1.txt
h:\Учеба\Программирование\lab_1\1s\2C_Lab_1>git commit -m "add 1.txt file"
[main a79a856] add 1.txt file
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 1.txt
h:\Учеба\Программирование\lab_1\1s\2C_Lab_1>git
```

Рисунок 1. Индексация первого файла и создания коммита

Проиндексируем второй и третий файл и перезапишем уже сделанный коммит с новым комментарием «add 2.txt and 3.txt»

```
h:\Учеба\Программирование\lab_1\1s\2C_Lab_1>git add 2.txt
h:\Учеба\Программирование\lab_1\1s\2C_Lab_1>git add 3.txt
h:\Учеба\Программирование\lab_1\1s\2C_Lab_1>git commit -m "add 2.txt and 3.txt"
[main 541bf90] add 2.txt and 3.txt
2 files changed, 0 insertions(+), 0 deletions(-)
create mode 100644 2.txt
create mode 100644 3.txt
h:\Учеба\Программирование\lab_1\1s\2C_Lab_1>
```

Рисунок 2. Индексация второго и третьего файла и создания коммита

Создадим новую ветку my\_first\_branch. Перейдем на ветку и создадим новый файл in\_branch.txt, закоммитим изменения.

```

h:\Учеба\Программирование\lab_1\1s\2C_Lab_1>git branch my_first_branch
h:\Учеба\Программирование\lab_1\1s\2C_Lab_1>git checkout my_first_branch
Switched to branch 'my_first_branch'
h:\Учеба\Программирование\lab_1\1s\2C_Lab_1>git add in_branch.txt
h:\Учеба\Программирование\lab_1\1s\2C_Lab_1>git commit -m "in_branch.txt"
[my_first_branch 5d980b1] in_branch.txt
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 in_branch.txt
h:\Учеба\Программирование\lab_1\1s\2C_Lab_1>

```

Рисунок 3. Создание новой ветки my\_first\_branch

Вернемся на ветку master. Создадим и сразу перейдем на ветку new\_branch. Сделаем изменения в файле 1.txt, добавим строчку «new row in the 1.txt file», закоммитим изменения.

```

h:\Учеба\Программирование\lab_1\1s\2C_Lab_1>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)
h:\Учеба\Программирование\lab_1\1s\2C_Lab_1>git checkout -b new_branch
Switched to a new branch 'new_branch'
h:\Учеба\Программирование\lab_1\1s\2C_Lab_1>git commit -m "red 1.txt"
On branch new_branch
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
no changes added to commit (use "git add" and/or "git commit -a")
h:\Учеба\Программирование\lab_1\1s\2C_Lab_1>git add 1.txt
h:\Учеба\Программирование\lab_1\1s\2C_Lab_1>git commit -m "red 1.txt"
[new_branch f08c018] red 1.txt
1 file changed, 1 insertion(+)

```

Рисунок 4. Создание новой ветки new\_branch

Перейдем на ветку master и сольем ветки master и my\_first\_branch, после чего сольем ветки master и new\_branch. Удалим ветки my\_first\_branch и new\_branch.

```

h:\Учеба\Программирование\lab_1\1s\2C_Lab_1>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)

h:\Учеба\Программирование\lab_1\1s\2C_Lab_1>git merge my_first_branch
Updating 541bf90..5d980b1
Fast-forward
 in_branch.txt | 0
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 in_branch.txt

h:\Учеба\Программирование\lab_1\1s\2C_Lab_1>git merge new_branch
Merge made by the 'recursive' strategy.
 1.txt | 1 +
 1 file changed, 1 insertion(+)

h:\Учеба\Программирование\lab_1\1s\2C_Lab_1>git branch -D my_first_branch
Deleted branch my_first_branch (was 5d980b1).

h:\Учеба\Программирование\lab_1\1s\2C_Lab_1>git branch -D new_branch
Deleted branch new_branch (was f08c018).

```

Рисунок 5. Слияние веток

Создадим ветки branch\_1 и branch\_2. Перейдем на ветку branch\_1 и изменим файл 1.txt, удалим все содержимое и добавим текст «fix in the 1.txt», изменим файл 3.txt, удалим все содержимое и добавим текст «fix in the 3.txt», закоммитим изменения.

```

h:\Учеба\Программирование\lab_1\1s\2C_Lab_1>git branch branch_1
h:\Учеба\Программирование\lab_1\1s\2C_Lab_1>git branch branch_2
h:\Учеба\Программирование\lab_1\1s\2C_Lab_1>git checkout branch_1
Switched to branch 'branch_1'

h:\Учеба\Программирование\lab_1\1s\2C_Lab_1>git add 1.txt
h:\Учеба\Программирование\lab_1\1s\2C_Lab_1>git add 2.txt
h:\Учеба\Программирование\lab_1\1s\2C_Lab_1>git add 3.txt
h:\Учеба\Программирование\lab_1\1s\2C_Lab_1>git commit -m "add 1 and 3 files"
[branch_1 e7f9f3f] add 1 and 3 files
 2 files changed, 2 insertions(+), 1 deletion(-)

```

Рисунок 6. Работа с веткой branch\_1

Перейдем на ветку branch\_2 и также изменить файл 1.txt, удалим все содержимое и добавим текст «My fix in the 1.txt», изменим файл 3.txt, удалим все содержимое и добавим текст «My fix in the 3.txt», закоммитим изменения.

```
h:\Учеба\Программирование\lab_1\1s\2C_Lab_1>git checkout branch_2
Switched to branch 'branch_2'

h:\Учеба\Программирование\lab_1\1s\2C_Lab_1>git add 1.txt

h:\Учеба\Программирование\lab_1\1s\2C_Lab_1>git add 3.txt

h:\Учеба\Программирование\lab_1\1s\2C_Lab_1>git commit -m "re 1 and 3 in br2"
[branch_2 5012151] re 1 and 3 in br2
2 files changed, 2 insertions(+), 1 deletion(-)
```

Рисунок 7. Работа с веткой branch\_2

Сольем изменения ветки branch\_2 в ветку branch\_1. Для жтоо перейдем на ветку branch\_1 и воспользуемся командой git merge.

```
h:\Учеба\Программирование\lab_1\1s\2C_Lab_1>git checkout branch_1
Switched to branch 'branch_1'

h:\Учеба\Программирование\lab_1\1s\2C_Lab_1>git merge branch_2
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 8. Слияние изменений branch\_2 в ветку branch\_1

Решим конфликт файла 1.txt в ручном режиме, а конфликт 3.txt используя команду git mergetool с помощью одной из доступных утилит.

```
h:\Учеба\Программирование\lab_1\1s\2C_Lab_1>git add 1.txt

h:\Учеба\Программирование\lab_1\1s\2C_Lab_1>git add 3.txt

h:\Учеба\Программирование\lab_1\1s\2C_Lab_1>git commit
[branch_1 bbff084] Merge branch 'branch_2' into branch_1
```

Рисунок 9. Решение конфликтов при слиянии

Отправим ветку branch\_1 на GitHub.

```
C:\Users\user\nen\2C_Lab_1>git push origin branch_1
Enumerating objects: 25, done.
Counting objects: 100% (25/25), done.
Delta compression using up to 12 threads
Compressing objects: 100% (16/16), done.
Writing objects: 100% (24/24), 2.09 KiB | 1.04 MiB/s, done.
Total 24 (delta 7), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (7/7), done.
remote:
remote: Create a pull request for 'branch_1' on GitHub by visiting:
remote:   https://github.com/Novomlinov/2C_Lab_1/pull/new/branch_1
remote:
To https://github.com/Novomlinov/2C_Lab_1.git
 * [new branch]      branch_1 -> branch_1
```

Рисунок 10. Отправка ветки branch\_1 на GitHub

Создадим средствами GitHub удаленную ветку branch\_3.

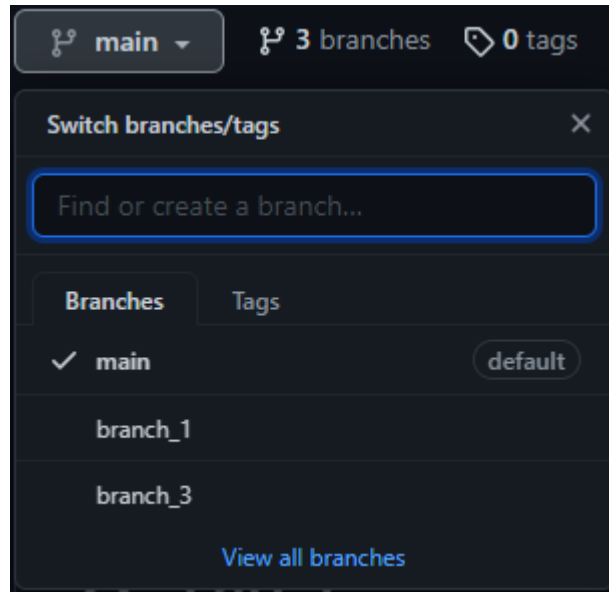


Рисунок 11. Создание удаленной ветки

Создадим в локальном репозитории ветку отслеживания удаленной ветки branch\_3.

```
C:\Users\user\nen\2C_Lab_1>git fetch origin
From https://github.com/Novomlinov/2C_Lab_1
+ bbff084...2408200 branch_3 -> origin/branch_3 (forced update)

C:\Users\user\nen\2C_Lab_1>git checkout -b branch_3 origin/branch_3
Switched to a new branch 'branch_3'
Branch 'branch_3' set up to track remote branch 'branch_3' from 'origin'.
```

Рисунок 12. Создание ветки отслеживания

Перейдем на ветку branch\_3 и добавим файл 2.txt, в котором добавим строку "the final fantasy in the 4.txt file". Выполним перемещение ветки master на ветку branch\_3. Отправим изменения веток master и branch\_2 на GitHub.

```
C:\Users\user\nen\2C_Lab_1>git push origin main
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/Novomlinov/2C_Lab_1.git
  2408200..0cf2ec8  main -> main

C:\Users\user\nen\2C_Lab_1>git push
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 12 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 336 bytes | 336.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/Novomlinov/2C_Lab_1.git
  2408200..9289a9a  branch_3 -> branch_3

C:\Users\user\nen\2C_Lab_1>git push origin branch_3
Everything up-to-date
```

Рисунок 13. Отправка веток master и branch на GitHub

### Контрольные вопросы:

1. Что такое ветка?

Ветка в Git — это простой перемещаемый указатель на один из таких коммитов.

2. Что такое HEAD?

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

3. Способы создания веток.

Ветки создаются с помощью команд «git branch \*имя ветки\*» или команды «git checkout -b \*имя ветки\*». Последняя команда позволяет сразу же перейти на созданную ветку.

4. Как узнать текущую ветку?

Введя команду «git branch» высветится список всех веток, а текущая ветка будет выделена цветом.

5. Как переключаться между ветками?

Команда «git checkout \*имя ветки\*»

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.

## 8. Как создать ветку отслеживания?

Команда «`git checkout -b *имя ветки* origin/*имя ветки*`».

## 9. Как отправить изменения из локальной ветки в удалённую ветку?

С помощью команды `git push *имя ветки*`.

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

`Git fetch` лишь копирует данные на локальный репозиторий, а команда `Git pull` получает данные и выполняет слияние с веткой на рабочем месте.

## 11. Как удалить локальную и удалённую ветки?

С помощью команды «`git branch -d *имя ветки*`»

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`?

Основные типы веток в модели `git-flow`:

- ветка разработки (`develop`);
- функциональная ветка (`feature`);
- ветка выпуска (`release`);
- ветка исправления (`hotfix`);

Работа с ветками организована следующим образом:

- из ветки `main` создается ветка `develop`;
- из ветки `develop` создается ветка `release`;
- из ветки `develop` создаются ветки `feature`;
- когда работа над веткой `feature` завершается, она сливается в ветку `develop`;
- когда работа над веткой `release` завершается, она сливается с ветками `develop` и `main`;



- если в ветке main обнаруживается проблема, из main создается ветка hotfix;

- когда работа над веткой hotfix завершается, она сливается с ветками develop и main.

Недостатки git-flow:

- git flow может замедлять работу;

- сложности с частотой релизов;

- трата времени в случае конфликтов;

**Вывод:** были исследованы базовые возможности по работе с локальными и удаленными ветками Git.