

Министерство науки и высшего образования Российской Федерации  
Федеральное государственное автономное образовательное учреждение  
высшего образования  
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт цифрового развития  
Кафедра инфокоммуникаций

**ОТЧЕТ**  
**ПО РАБОТЕ №1.3.**  
**дисциплины «Основы кроссплатформенного программирования»**

Выполнил:  
Липчанский Дмитрий Сергеевич  
1 курс, группа ИТС-б-о-22-1,  
11.03.02 «Инфокоммуникационные  
технологии и системы связи»,  
направленность (профиль)  
«Инфокоммуникационные системы и  
сети», очная форма обучения

---

(подпись)

Руководитель практики:  
Воронкин Р.А., канд. тех. наук, доцент,  
доцент кафедры инфокоммуникаций

---

(подпись)

Отчет защищен с оценкой \_\_\_\_\_ Дата защиты \_\_\_\_\_

Ставрополь, 2023 г.

**Тема:** основы ветвления Git.

**Цель работы:** исследовать базовые возможности по работе с локальными и удаленными ветками Git.

### **Порядок выполнения работы:**

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

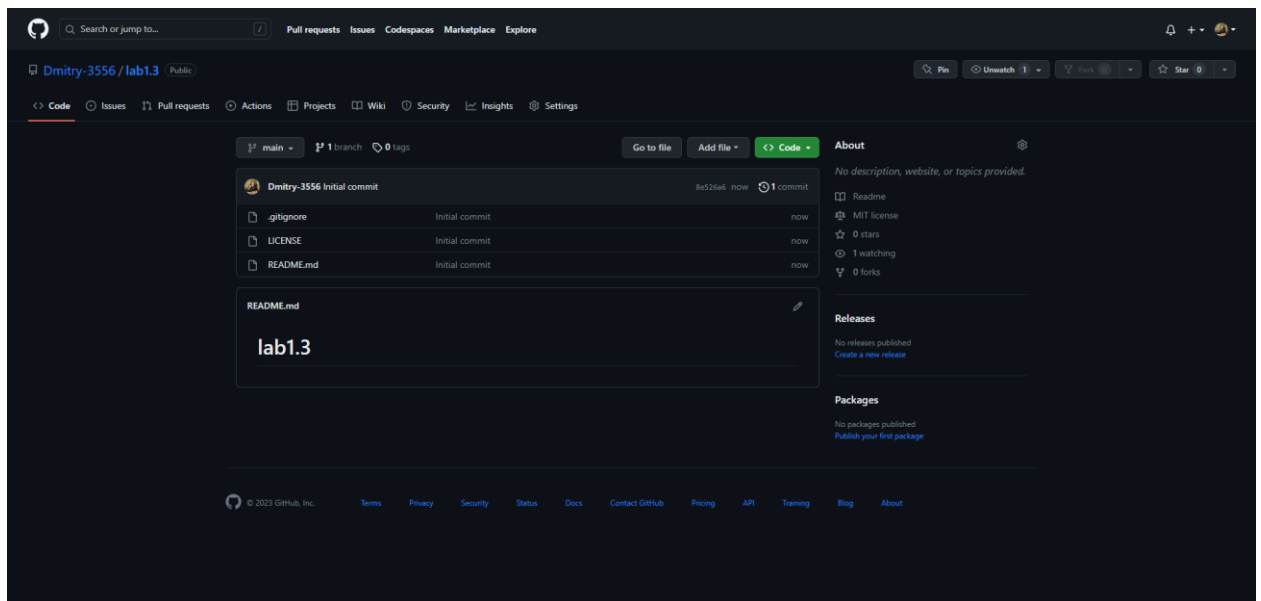


Рис. 1. Новый репозиторий.

2. Проклонировал репозиторий на свой компьютер.

```
C:\Work>git clone https://github.com/Dmitry-3556/lab1.3.git
Cloning into 'lab1.3'...
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 5 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (5/5), done.

C:\Work>cd C:\Work\lab1.3

C:\Work\lab1.3>git status
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean

C:\Work\lab1.3>
```

Рис. 2. Клонирование.

3. Добавил 3 новых текстовых файла.




 1	29.05.2023 13:59	Текстовый докум...	0 КБ
 2	29.05.2023 13:59	Текстовый докум...	0 КБ
 3	29.05.2023 13:59	Текстовый докум...	0 КБ

Рис. 3. Новые текстовые файлы.

4. Проиндексировал первый файл и сделать коммит.

```
C:\Work\lab1.3>git add 1.txt

C:\Work\lab1.3>git commit -m "1.txt file"
[main 1af9048] 1.txt file
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 1.txt

C:\Work\lab1.3>|
```

Рис. 4. Добавление изменений и их фиксация.

5. Проиндексировал второй и третий файлы.

```
C:\Work\lab1.3>git status
On branch main
Your branch is ahead of 'origin/main' by 1 commit.
  (use "git push" to publish your local commits)

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   2.txt
        new file:   3.txt

C:\Work\lab1.3>
```

Рис. 5. Добавление изменений.

6. Перезаписал уже сделанный коммит.

```
C:\Work\lab1.3>git commit --amend -m "add 2 and 3 txt file"
[main 30bab6] add 2 and 3 txt file
Date: Mon May 29 14:01:05 2023 +0300
 3 files changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 1.txt
 create mode 100644 2.txt
 create mode 100644 3.txt

C:\Work\lab1.3>|
```

Рис. 6. Редактирование существующего коммита.

7. Создание новой ветки и переход на нее и создать новый файл in\_branch.txt.

```
C:\Work\lab1.3>git branch my_first_branch

C:\Work\lab1.3>git checkout my_first_branch
Switched to branch 'my_first_branch'

C:\Work\lab1.3>git status
On branch my_first_branch
nothing to commit, working tree clean

C:\Work\lab1.3>git status
On branch my_first_branch
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        in_branch.txt

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

C:\Work\lab1.3>git add .

C:\Work\lab1.3>git commit -m "in_branch"
[my_first_branch 51b1698] in_branch
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 in_branch.txt

C:\Work\lab1.3>
```

Рис. 8. Создание файла и фиксация.

8. Перешел вновь на новую ветку main и создал и сразу перешел на ветку new\_branch.

```
C:\Work\lab1.3>git checkout main
Switched to branch 'main'
Your branch is ahead of 'origin/main' by 1 commit.
  (use "git push" to publish your local commits)

C:\Work\lab1.3>git checkout -b new_branch
Switched to a new branch 'new_branch'

C:\Work\lab1.3>
```

Рис. 10. Создание и сразу переход на ветку.

9. Сделал изменения в файле 1.txt, добавил строчку “new row in the 1.txt file”, закоммитил изменения.

```
C:\Work\lab1.3>git status
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")

C:\Work\lab1.3>git add .

C:\Work\lab1.3>git commit -m "new row in the 1.txt file"
[new_branch bb44d18] new row in the 1.txt file
1 file changed, 1 insertion(+)
```

Рис. 11. Изменение в 1.txt фиксация этих изменений.

10. Перешел на ветку main и слил ветки main и my\_first\_branch, после слил ветки main и new\_branch.

```
C:\Work\lab1.3>git merge my_first_branch
Updating 30babc6..51b1698
Fast-forward
 in_branch.txt | 0
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 in_branch.txt

C:\Work\lab1.3>git merge new_branch
Merge made by the 'ort' strategy.
 1.txt | 1 +
 1 file changed, 1 insertion(+)
```

Рис. 12. Слияние веток.

11. Удалил ветки my\_first\_branch и new\_branch.

```
C:\Work\lab1.3>git branch -d my_first_branch
Deleted branch my_first_branch (was 51b1698).

C:\Work\lab1.3>git branch -d new_branch
Deleted branch new_branch (was bb44d18).
```

Рис. 13. Удаление веток.

12. Создал ветки branch\_1 и branch\_2.

```
C:\Work\lab1.3>git branch branch_1

C:\Work\lab1.3>git branch branch_2

C:\Work\lab1.3>git branch
  branch_1
  branch_2
* main
```

Рис. 14. Создание новых веток.

13. Перешел на ветку branch\_1 и изменил файл 1.txt, удалил все содержимое и добавил текст “fix in the 1.txt”, изменил файл 3.txt, удалил все содержимое и добавил текст “fix in the 3.txt”, закоммитил изменения.

```
C:\Work\lab1.3>git checkout branch_1
Switched to branch 'branch_1'

C:\Work\lab1.3>git add .

C:\Work\lab1.3>git commit -m "izmeneniya v 1 and 3 file"
[branch_1 86a254a] izmeneniya v 1 and 3 file
2 files changed, 2 insertions(+), 1 deletion(-)
```

Рис. 15. Изменения в первой ветке и во второй ветка и фиксация изменений.

14. Перешел на ветку branch\_2 и также изменил файл 1.txt, удалил все содержимое и добавил текст “My fix in the 1.txt”, изменил файл 3.txt, удалил все содержимое и добавил текст “My fix in the 3.txt”, закоммитил изменения.

```
C:\Work\lab1.3>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:\Work\lab1.3>git add .

C:\Work\lab1.3>git commit -m "izmeneniya v 1 and 3 file in branch_2"
[branch_2 27095e8] izmeneniya v 1 and 3 file in branch_2
2 files changed, 2 insertions(+), 1 deletion(-)
```

Рис. 16. Переход на вторую ветку.



15. Слил изменения ветки branch\_2 в ветку branch\_1.

```
C:\Work\lab1.3>git chekout branch_1
git: 'chekout' is not a git command. See 'git --help'.

The most similar command is
    checkout

C:\Work\lab1.3>git checkout branch_1
Switched to branch 'branch_1'

C:\Work\lab1.3>git merge branch_2
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.

C:\Work\lab1.3>
```

Рис. 17. Слияние веток.

16. Решил конфликт файла 1.txt и 2.txt в ручном режиме.

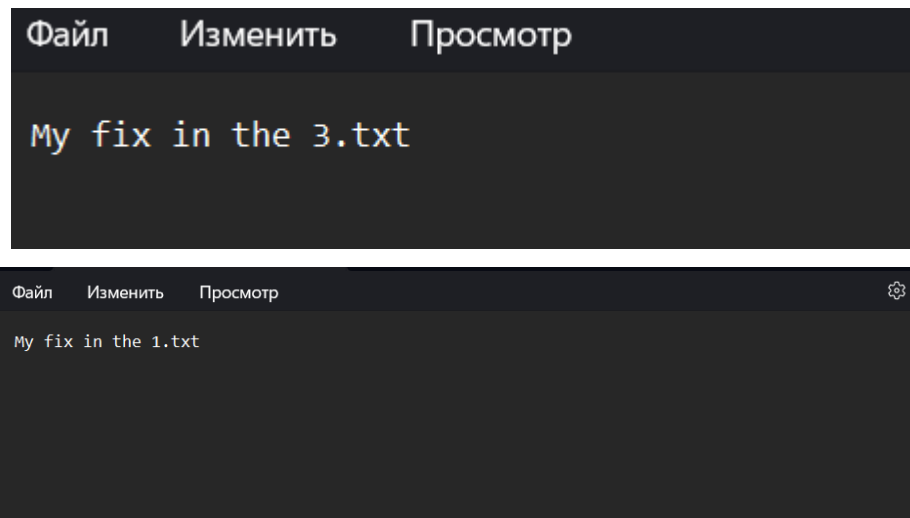


Рис. 18. Решение конфликта в ручную.

```
Changes to be committed:
  modified:   1.txt
  modified:   3.txt
```

Рис. 19. Решение конфликта в ручную.

17. Отправил branch\_1 на удаленный *git push origin branch\_1*.

```
C:\Work\lab1.3>git commit -m "Reshenie konfliktov"
[branch_1 2a65756] Reshenie konfliktov

C:\Work\lab1.3>git status
On branch branch_1
nothing to commit, working tree clean

C:\Work\lab1.3>git push origin branch_1
Enumerating objects: 23, done.
Counting objects: 100% (23/23), done.
Delta compression using up to 8 threads
Compressing objects: 100% (14/14), done.
Writing objects: 100% (22/22), 1.78 KiB | 1.78 MiB/s, done.
Total 22 (delta 8), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (8/8), done.
remote:
remote: Create a pull request for 'branch_1' on GitHub by visiting:
remote:   https://github.com/Dmitry-3556/lab1.3/pull/new/branch_1
remote:
To https://github.com/Dmitry-3556/lab1.3.git
 * [new branch]      branch_1 -> branch_1

C:\Work\lab1.3>
```

Рис. 20. Отправление ветки на удаленный сервер.

18. Создал средствами GitHub удаленную ветку branch\_3.

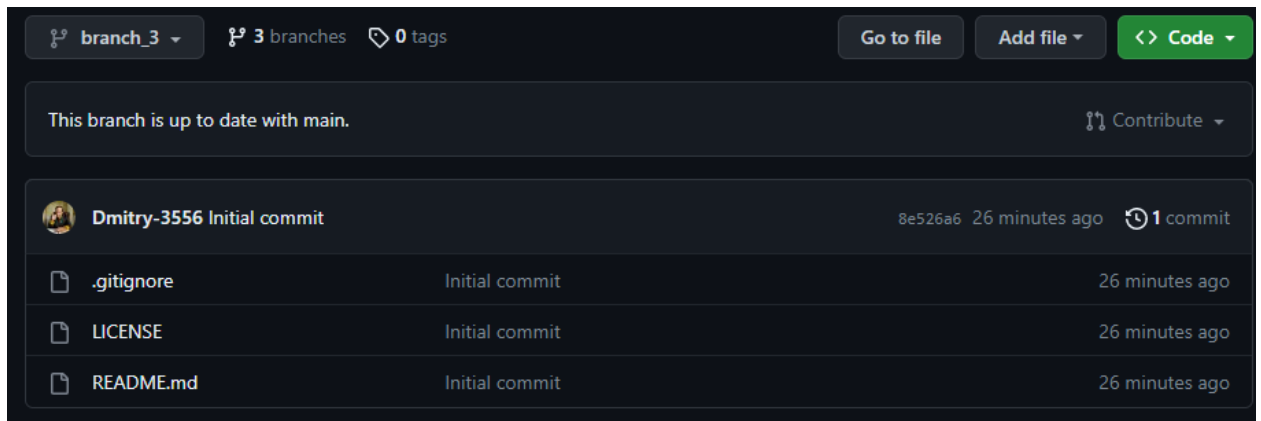


Рис. 21. Создание ветки на GitHub.

19. Создал в локальном репозитории ветку отслеживания удаленной ветки branch\_3.

```
C:\Work\lab1.3>git fetch origin
From https://github.com/Dmitry-3556/lab1.3
* [new branch]      branch_3    -> origin/branch_3

C:\Work\lab1.3>git checkout -b branch_3 origin/branch_3
Switched to a new branch 'branch_3'
branch 'branch_3' set up to track 'origin/branch_3'.
```

Рис. 22. Создание ветки отслеживания.

20. Перешел на ветку branch\_3 и добавил файл файл 2.txt строку "the final fantasy in the 4.txt file".

21. Выполнил перемещение ветки main на ветку branch\_2.

```
C:\Work\lab1.3>git commit -m "the final fantasy in the 4.txt file in branch
_3"
[branch_3 564cc5d] the final fantasy in the 4.txt file in branch _3
1 file changed, 1 insertion(+)
create mode 100644 2.txt

C:\Work\lab1.3>git checkout main
Switched to branch 'main'
Your branch is ahead of 'origin/main' by 4 commits.
(use "git push" to publish your local commits)

C:\Work\lab1.3>git rebase branch_2
Successfully rebased and updated refs/heads/main.

C:\Work\lab1.3>git checkout branch_2
Switched to branch 'branch_2'

C:\Work\lab1.3>git merge main
Already up to date.

C:\Work\lab1.3>
```

Рис. 25. Перемещение веток.

22. Отправил изменения веток master и branch\_2 на удаленный сервер.

```
C:\Work\lab1.3>git checkout main
Switched to branch 'main'
Your branch is ahead of 'origin/main' by 5 commits.
  (use "git push" to publish your local commits)

C:\Work\lab1.3>git push origin main
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/Dmitry-3556/lab1.3.git
   8e526a6..27095e8  main -> main

C:\Work\lab1.3>git chekcout branch_2
git: 'chekcout' is not a git command. See 'git --help'.

The most similar command is
      checkout

C:\Work\lab1.3>git checkout branch_2
Switched to branch 'branch_2'

C:\Work\lab1.3>git push 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/Dmitry-3556/lab1.3/pull/new/branch_2
remote:
To https://github.com/Dmitry-3556/lab1.3.git
 * [new branch]      branch_2 -> branch_2

C:\Work\lab1.3>
```

Рис. 25. Отправка изменения веток на GitHub.

**Ссылка:** <https://github.com/Dmitry-3556/lab1.3>

### Ответы на контрольные вопросы:

1. Что такое ветка?

Почти каждая система контроля версий (СКВ) в какой-то форме поддерживает ветвление. Используя ветвление, Вы отклоняетесь от основной линии разработки и продолжаете работу независимо от неё, не вмешиваясь в основную линию.

2. Что такое HEAD?

HEAD в Git – это указатель на текущую ссылку ветви, которая, в свою очередь, является указателем на последний сделанный вами коммит или последний коммит, который был извлечен из вашего рабочего каталога.

3. Способы создания веток.

Создать ветку можно с помощью двух команд. Команда, которая просто создает ветку: `git branch "name_branch"`.

Команда, которая создает ветку и сразу же к ней переходит: `git checkout -b "name_branch"`.

4. Как узнать текущую ветку?

Текущую ветку можно узнать с помощью команды: `git branch`.

5. Как переключаться между ветками?

Между ветками можно переключаться с помощью команды: `git checkout "name_branch"`.

6. Что такое удаленная ветка?

Удаленные ветки - это ссылки на определенное состояние удалённых веток. Это локальные ветки, которые нельзя перемещать.

7. Что такое ветка отслеживания?

Отслеживаемые ветки — это локальные ветки, которые напрямую связаны с удалённой веткой.

Если, находясь на отслеживаемой ветке, вы наберёте `git push`, Git уже будет знать, на какой сервер и в какую ветку отправлять изменения.

8. Как создать ветку отслеживания?

Ветку отслеживания можно создать с помощью команды: `git checkout --track origin/<name_branch>`.

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

Отправить изменения из локальной ветки в удаленную можно с помощью команды: `git push origin <name_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?

Для удаление локальной ветки используется команда: `git branch -d <name_branch>`

Для удаления удаленной ветки используется команда: `git push --delete origin/<name_branch>`

## 12. Какие основные типы веток присутствуют в модели git-flow? Как организована работа с ветками в модели git-flow? В чем недостатки git-flow?

Существуют следующие типы ветвей:

- 1) ветви функциональностей;
- 2) ветви релизов;
- 3) ветви исправлений.

Ветви функциональностей (feature branches), также называемые иногда тематическими ветвями (topic branches), используются для разработки новых функций, которые должны появиться в текущем или будущем релизах.

Ветви релизов (release branches) используются для подготовки к выпуску новых версий продукта. Они позволяют расставить финальные точки над *i* перед выпуском новой версии.

Ветви для исправлений (hotfix branches) весьма похожи на ветви релизов (release branches), так как они тоже используются для подготовки новых выпусков продукта, разве лишь незапланированных.

Недостатки git flow: авторам приходится использовать ветку develop вместо master, поскольку master зарезервирован для кода.

Вторая проблема процесса git flow – сложности, возникающие из-за веток для патчей и для релиза. Подобная структура может подойти некоторым организациям, но для абсолютного большинства она просто убийственно излишня.