

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

**Кафедра инфокоммуникаций**

**Отчет по лабораторной работе №2**

**Исследование основных возможностей Git и GitHub**

**по дисциплине «Основы кроссплатформенного программирования»**

Выполнил студент группы ИВТ-б-о-20-1

Токарев В. А. « » \_\_\_\_\_ 20\_\_ г.

Подпись студента \_\_\_\_\_

Работа защищена « » \_\_\_\_\_ 20\_\_ г.

Проверил Воронкин Р. А. \_\_\_\_\_

(подпись)


Ставрополь 2021

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

## Ход работы:

Ссылка на репозиторий: <https://github.com/UnDeR-The-mAsK/lab2.git>

1. Создал новый репозиторий, сделал его удаленное клонирование командой git clone.

 Git CMD

```
C:\Users\Владислав>cd C:\Users\Владислав\Desktop\Git\rep2  
C:\Users\Владислав\Desktop\Git\rep2>git clone https://github.com/UnDeR-The-mAsK/lab2.git  
Cloning into 'lab2'...  
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.
```

Рис. 1 Клонирование репозитория на ПК

2. Проиндексировал первый файл и сделать коммит с комментарием "add 1.txt file". Аналогично проиндексировал и сделал коммит для других файлов.

```
C:\Users\Владислав\Desktop\Git\rep2\lab2>git add 1.txt  
C:\Users\Владислав\Desktop\Git\rep2\lab2>git init  
Reinitialized existing Git repository in C:/Users/Владислав/Desktop/Git/rep2/lab2/.git/  
C:\Users\Владислав\Desktop\Git\rep2\lab2>git commit -m"add 1.txt file"  
[main c17e151] add 1.txt file  
1 file changed, 0 insertions(+), 0 deletions(-)  
create mode 100644 1.txt
```

Рис. 2 Коммит для первого файла

```
C:\Users\Владислав\Desktop\Git\rep2\lab2>git add .  
C:\Users\Владислав\Desktop\Git\rep2\lab2>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:\Users\Владислав\Desktop\Git\rep2\lab2>git commit -m"add 2.txt and 3.txt"  
[main 8debaa8] add 2.txt and 3.txt  
2 files changed, 0 insertions(+), 0 deletions(-)  
create mode 100644 2.txt  
create mode 100644 3.txt
```

Рис. 3 Коммит для двух других файлов

3. Создал новую ветку и перешёл на неё.

```
C:\Users\Владислав\Desktop\Git\rep2\lab2>git branch my_first_branch  
C:\Users\Владислав\Desktop\Git\rep2\lab2>git checkout my_first_branch  
Switched to branch 'my_first_branch'
```

Рис. 4 Новая ветка

4. Добавил файл в новую ветку и сделал коммит.

```
C:\Users\Владислав\Desktop\Git\rep2\lab2>git add in_branch.txt  
C:\Users\Владислав\Desktop\Git\rep2\lab2>git commit -m "add in_branch.txt"  
[my_first_branch ad35b3e] add in_branch.txt  
1 file changed, 0 insertions(+), 0 deletions(-)  
create mode 100644 in_branch.txt
```

Рис. 5 Создание нового файла в ветке

5. Вернулся на ветку main и создал в ней новую, переключившись на неё.

```
C:\Users\Владислав\Desktop\Git\rep2\lab2>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\Владислав\Desktop\Git\rep2\lab2>git branch new_branch  
C:\Users\Владислав\Desktop\Git\rep2\lab2>git checkout new_branch  
Switched to branch 'new_branch'
```

Рис. 6 Создание новой ветки

6. Сделал изменения в файле 1.txt и закоммитил его.

```
C:\Users\Владислав\Desktop\Git\rep2\lab2>git add .  
C:\Users\Владислав\Desktop\Git\rep2\lab2>git commit -m "file changed"  
[new_branch ff5caa8] file changed  
1 file changed, 1 insertion(+)
```

Рис. 7 Изменения в файле

7. Перешёл на ветку main и слил ветки my\_first\_branch и new\_branch. Затем удалил их.

```
C:\Users\Владислав\Desktop\Git\rep2\lab2>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\Владислав\Desktop\Git\rep2\lab2>git merge my_first_branch
Updating 8debaa8..ad35b3e
Fast-forward
 in_branch.txt | 0
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 in_branch.txt

C:\Users\Владислав\Desktop\Git\rep2\lab2>git status
On branch main
Your branch is ahead of 'origin/main' by 3 commits.
  (use "git push" to publish your local commits)

nothing to commit, working tree clean

C:\Users\Владислав\Desktop\Git\rep2\lab2>git merge new_branch
Merge made by the 'recursive' strategy.
 1.txt | 1 +
 1 file changed, 1 insertion(+)

C:\Users\Владислав\Desktop\Git\rep2\lab2>git branch -d my_first_branch
Deleted branch my_first_branch (was ad35b3e).

C:\Users\Владислав\Desktop\Git\rep2\lab2>git branch -d new_branch
Deleted branch new_branch (was ff5caa8).
```

Рис. 8 Слияние веток

8. Создал ветки branch\_1 и branch\_2. Изменение в branch\_1 файлов, также изменение в ветки branch\_2.

```

C:\Users\Владислав\Desktop\Git\rep2\lab2>git branch branch_1
C:\Users\Владислав\Desktop\Git\rep2\lab2>git branch branch_2
C:\Users\Владислав\Desktop\Git\rep2\lab2>git checkout branch_1
Switched to branch 'branch_1'
C:\Users\Владислав\Desktop\Git\rep2\lab2>git add .
C:\Users\Владислав\Desktop\Git\rep2\lab2>git status
On branch branch_1
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   1.txt
        modified:   3.txt

C:\Users\Владислав\Desktop\Git\rep2\lab2>git commit -m "fixed 1.txt and 3.txt"
[branch_1 14010f8] fixed 1.txt and 3.txt
 2 files changed, 2 insertions(+), 1 deletion(-)

C:\Users\Владислав\Desktop\Git\rep2\lab2>git checkout branch_2
Switched to branch 'branch_2'
C:\Users\Владислав\Desktop\Git\rep2\lab2>git add .
C:\Users\Владислав\Desktop\Git\rep2\lab2>git status
On branch branch_2
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   1.txt
        modified:   3.txt

C:\Users\Владислав\Desktop\Git\rep2\lab2>git commit -m "new fix 1.txt and 3.txt"
[branch_2 f400c95] new fix 1.txt and 3.txt
 2 files changed, 2 insertions(+), 1 deletion(-)

```

Рис. 9 Добавление новых веток и изменение в них файлов

9. Сливаем изменение ветки branch\_2 в branch\_1 и получаем конфликт.

```

C:\Users\Владислав\Desktop\Git\rep2\lab2>git checkout branch_1
Switched to branch 'branch_1'

C:\Users\Владислав\Desktop\Git\rep2\lab2>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.

```

Рис. 10 Появление конфликта

10. Исправил конфликт вручную.

```

C:\Users\Владислав\Desktop\Git\rep2\lab2>git add .
C:\Users\Владислав\Desktop\Git\rep2\lab2>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

C:\Users\Владислав\Desktop\Git\rep2\lab2>git commit -m "fixed 1.txt and 3.txt"
[branch_1 c475409] fixed 1.txt and 3.txt
C:\Users\Владислав\Desktop\Git\rep2\lab2>git merge branch_2
Already up to date.

```

Рис. 11 Решение конфликтов

11. Отправил ветку branch\_1 на GitHub и удалил branch\_2.

```

C:\Users\Владислав\Desktop\Git\rep2\lab2>git push origin branch_1
Enumerating objects: 26, done.
Counting objects: 100% (26/26), done.
Delta compression using up to 24 threads
Compressing objects: 100% (17/17), done.
Writing objects: 100% (25/25), 2.13 KiB | 1.06 MiB/s, done.
Total 25 (delta 10), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (10/10), done.
remote:
remote: Create a pull request for 'branch_1' on GitHub by visiting:
remote:   https://github.com/UnDeR-The-mAsK/lab2/pull/new/branch_1
remote:
To https://github.com/UnDeR-The-mAsK/lab2.git
 * [new branch]      branch_1 -> branch_1

C:\Users\Владислав\Desktop\Git\rep2\lab2>git branch -d branch_2
Deleted branch branch_2 (was f400c95).

```

Рис. 12 Пуш ветки на GitHub

12. Средствами GitHub создал удаленную ветку branch\_3.

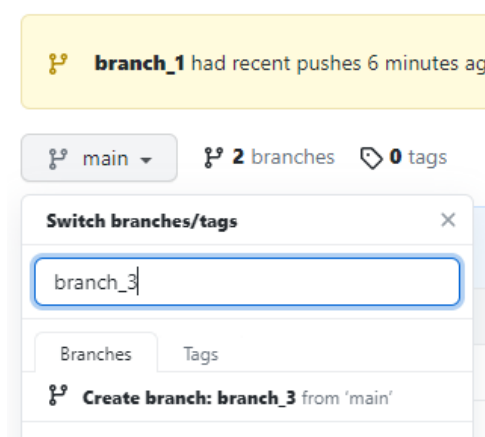


Рис. 13

13. Получил ветку branch\_3 на локальный репозиторий с помощью git fetch и создал ветку отслеживания.

```
C:\Users\Владислав\Desktop\Git\rep2\lab2>git fetch origin
From https://github.com/UnDeR-The-mAsK/lab2
* [new branch]      branch_3    -> origin/branch_3

C:\Users\Владислав\Desktop\Git\rep2\lab2>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'.

C:\Users\Владислав\Desktop\Git\rep2\lab2>git branch
  branch_1
* branch_3
  main
```

Рис. 14 Получение данных с GitHub

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

```
C:\Users\Владислав\Desktop\Git\rep2\lab2>git branch branch_2

C:\Users\Владислав\Desktop\Git\rep2\lab2>git checkout branch_2
Switched to branch 'branch_2'

C:\Users\Владислав\Desktop\Git\rep2\lab2>git rebase main
Successfully rebased and updated refs/heads/branch_2.
```

Рис. 15 Перемещение веток

15. Отправил измененные ветки 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.

Чтобы создать ветку необходимо ввести команду git branch и название ветки в командную строку GitCmd

4. Как узнать текущую ветку?

Текущая ветка будет выделяться звездой в командной строке GitCmd

5. Как переключаться между ветками?

Для переключения между ветками можно ввести команду `git checkout` и ввести название самой ветки.

#### 6. Что такое удаленная ветка?

Удалённые ветки – это ссылки удалённых репозиториях пользователя, включая ветки, теги и так далее. Полный список удалённых ссылок можно получить с помощью команды `git ls-remote` или команды `git remote show` для получения удалённых веток и дополнительной информации

#### 7. Что такое ветка отслеживания?

Ветки отслеживания — это локальные ветки, которые напрямую связаны с удалённой веткой. Если, находясь на ветке слежения, выполнить `git pull`, то Git уже будет знать с какого сервера получать данные и какую ветку использовать для слияния.

#### 8. Как создать ветку отслеживания?

Чтобы создать ветку отслеживания нужно воспользоваться параметрами `-u` или `-set-upstream-to` для команды `git branch`;

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

Чтобы отправить изменения из локальной ветки в удалённую необходимо ввести команду `git push`;

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

Команда `git fetch` забирает изменения из указанного удалённого репозитория

А команда `git pull` работает как две команды (`git fetch` и `git merge`), то есть забирает изменения из указанного удалённого репозитория, а затем пытается слить их с предыдущей веткой `git push` с параметром `--d` (`--delete`).

#### 11. Как удалить локальную и удаленную ветки?

Чтобы удалить локальную ветку достаточно просто удалить файлы, клонированные из удалённой. Однако, чтобы удалить удалённую ветку репозитория, нужно прописать команду `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`?

Типы ветвей:



- master– ветка, которая стоит по умолчанию;
- develop– ветка, которая создана для ответвления от ветки master и слияния в неё изменений, которые пользователь собирается сливать в master для;

Основные типы веток, которые используются в gitflow:

- Feature branches;
- Release branches;
- Hotfix branches;

Основные недостатки git flow: может замедлять работу; замедляет просмотр реальной работы из-за кучи merge commit'а; сложно делать релизы раньше одной недели; проблематичный в CI/CDсценариях; большие могут затратить несколько дней на решение конфликтов.

13. На прошлой лабораторной работе было задание выбрать одно из программных средств с GUI для работы с Git. Необходимо в рамках этого вопроса привести описание инструментов для работы с ветками Git, предоставляемых этим средством

Ветка в Git — это простой перемещаемый указатель на один из таких коммитов. По умолчанию, имя основной ветки в Git — master  
Sourcetree позволяет работать с теми же инструментами, которые представлены в строке CMD, но при этом управление здесь намного интуитивно понятнее и проще. SourceTree и ветвление git-flow позволяют убрать беспорядок в репозиториях.

– Чтобы создать новую ветку, нужно нажать “branch” в верхнем меню. В поле “new branch” ввести имя новой ветки. Выбрать пункт “create branch”.

– Если нажать на “history” в SourceTree можно увидеть, что в репозитории теперь есть незафиксированные изменения.

– Чтобы зафиксировать изменения в файле нужно нажать кнопку “commit” сверху, чтобы зафиксировать файл. Когда откроется окно сообщения нужно написать про свои изменения и подтвердить свои действия, нажав на “ОК”.

– Чтобы переместить данную ветку в репозитория необходимо нажать на кнопку “push”. После этого появится диалоговое окно, в котором нужно будет выбрать новую ветвь, чтобы указать перемещение этой ветви в исходную точку и после этого подтвердить свои действия, нажав на “ОК”

**Вывод:** в ходе выполнения лабораторной работы исследовал базовые возможности по работе с локальными и удаленными ветками Git.