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

Кафедра инфокоммуникаций

Языки програмирования
Отчет по лабораторной работе №1

Выполнил студент группы
ИТС-б-о-20-1 (2)
Усиков Д. А. « »_____20__г.
Подпись студента_____
Работа защищена « »_____20__г.
Проверил к.т.н., доцент
Кафедры инфокоммуникаций
Воронкин Р.А.
_____
(подпись)

Ставрополь 2021

Ссылка на репозиторий – https://github.com/danilusikov0913/lr1

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

Ход работы:

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

```
C:\Users\Acer\Desktop\Учеба\2курс 1семетр\Языки Програирования>git clone https://github.com/danilusikov0913/lr1.git
Cloning into 'lr1'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (3/3), done.

C:\Users\Acer\Desktop\Учеба\2курс 1семетр\Языки Програирования>
```

Рисунок 1. Клонирование репозитория

2. Создал три файла: 1.txt, 2.txt, 3.txt:

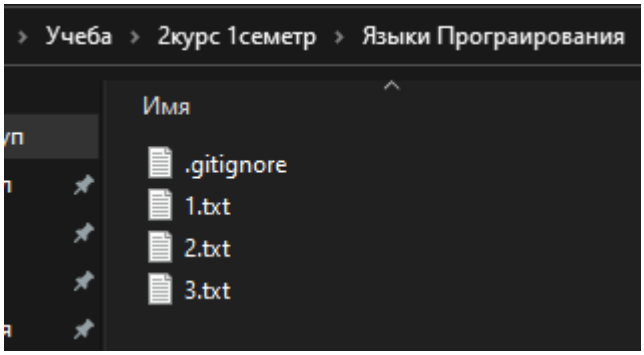

Рисунок 2. Созданные текстовые документы

3. Проиндексировал первый файл и сделала коммит с комментарием "add 1.txt file":

```
C:\Users\Acer\Desktop\Учеба\2курс 1семетр\Языки Програирования\lr1>git add 1.txt

C:\Users\Acer\Desktop\Учеба\2курс 1семетр\Языки Програирования\lr1>git commit -m "add 1.txt file"
[main 9c3b098] add 1.txt file
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 1.txt
```

Рисунок 3. Индексация первого файла

4. Проиндексировал второй и третий файлы, сделал коммит с комментарием "2.txt and 3.txt", перезаписал уже сделанный коммит с новым комментарием "add 2.txt and 3.txt

```
C:\Users\Acer\Desktop\Учеба\2курс 1семетр\Языки Програирования\lr1>git add 2.txt 3.txt

C:\Users\Acer\Desktop\Учеба\2курс 1семетр\Языки Програирования\lr1>git commit -m "2.txt and 3.txt"
[main 7d1125f] 2.txt and 3.txt
 2 files changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 2.txt
 create mode 100644 3.txt
```

```
C:\Users\Acer\Desktop\Учеба\2курс 1семетр\Языки Програирования\lr1>git commit --amend -m "add 2.txt and 3.txt"
[main 593e026] add 2.txt and 3.txt
 Date: Sat Sep 25 01:14:39 2021 +0300
 2 files changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 2.txt
 create mode 100644 3.txt

C:\Users\Acer\Desktop\Учеба\2курс 1семетр\Языки Програирования\lr1>
```

Рисунок 4. Индексация второго и третьего файлов

5. Создал новую ветку my_first_branch, перешел на нее и создал новый файл in_branch.txt, закоммитив изменения:

```
C:\Users\Acer\Desktop\Учеба\2курс 1семетр\Языки Програирования\lr1>git branch my_first_branch

C:\Users\Acer\Desktop\Учеба\2курс 1семетр\Языки Програирования\lr1>git checkout my_first_branch
Switched to branch 'my_first_branch'

C:\Users\Acer\Desktop\Учеба\2курс 1семетр\Языки Програирования\lr1>git add in_branch.txt

C:\Users\Acer\Desktop\Учеба\2курс 1семетр\Языки Програирования\lr1>git commit -m "add in_branch.txt"
[my_first_branch bab15b5] add in_branch.txt
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 in_branch.txt
```

Рисунок 5. Создание новой ветки и коммит изменений

6. Вернуться на ветку main
7. Создал и сразу перешел на ветку new_branch:

```
C:\Users\Acer\Desktop\Учеба\2курс 1семетр\Языки Програирования\lr1>git checkout -b new_branch
Switched to a new branch 'new_branch'

C:\Users\Acer\Desktop\Учеба\2курс 1семетр\Языки Програирования\lr1>
```

Рисунок 6. Создание новой ветки

8. Сделал изменения в файле 1.txt, добавил строчку “new row in the 1.txt file”, закоммитив изменения:

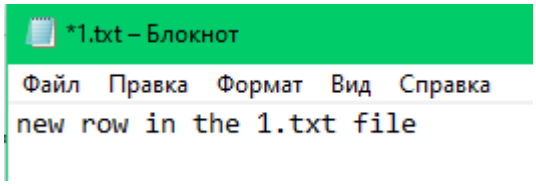

Рисунок 7. Изменение текстового файла

```
C:\Users\Acer\Desktop\Учеба\2курс 1семетр\Языки Програирования\lr1>git add 1.txt

C:\Users\Acer\Desktop\Учеба\2курс 1семетр\Языки Програирования\lr1>git commit -m "change 1.txt"
[new_branch 71561f7] change 1.txt
 1 file changed, 1 insertion(+)
```

Рисунок 8. Коммит изменений

9. Перешл на ветку main и слил ветки master и my_first_branch, после чего слил ветки main и new_branch:

```
C:\Users\Acer\Desktop\Учеба\2курс 1семетр\Языки Програирования\lr1>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\Acer\Desktop\Учеба\2курс 1семетр\Языки Програирования\lr1>git merge my_first_branch
Updating 593e026..bab15b5
Fast-forward
 in_branch.txt | 0
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 in_branch.txt

C:\Users\Acer\Desktop\Учеба\2курс 1семетр\Языки Програирования\lr1>git merge new_branch
Merge made by the 'recursive' strategy.
 1.txt | 1 +
 1 file changed, 1 insertion(+)

C:\Users\Acer\Desktop\Учеба\2курс 1семетр\Языки Програирования\lr1>git branch
* main
  my_first_branch
  new_branch
```

Рисунок 9. Слияние веток

10. Удалил ветки my_first_branch и new_branch и создал ветки branch_1 и branch_2:

```
C:\Users\Acer\Desktop\Учеба\2курс 1семетр\Языки Програирования\lr1>git branch -d my_first_branch
Deleted branch my_first_branch (was bab15b5).

C:\Users\Acer\Desktop\Учеба\2курс 1семетр\Языки Програирования\lr1>git branch -d new_branch
Deleted branch new_branch (was 71561f7).

C:\Users\Acer\Desktop\Учеба\2курс 1семетр\Языки Програирования\lr1>git branch branch_1

C:\Users\Acer\Desktop\Учеба\2курс 1семетр\Языки Програирования\lr1>git branch branch_2

C:\Users\Acer\Desktop\Учеба\2курс 1семетр\Языки Програирования\lr1>git branch
  branch_1
  branch_2
* main
```

Рисунок 10. Удаление и создание новых веток

11. Перешел на ветку branch_1 и изменил файл 1.txt на текст “fix in the 1.txt”, изменил файл 3.txt на текст “fix in the 3.txt”, закоммитив изменения

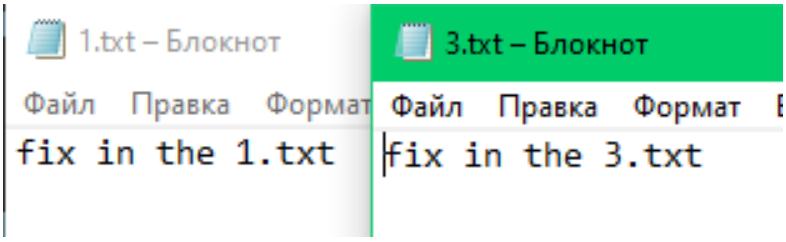

Рисунок 11. Изменение текстовых файлов

```
C:\Users\Acer\Desktop\Учеба\2курс 1семетр\Языки Програирования\lr1>git add 1.txt 3.txt

C:\Users\Acer\Desktop\Учеба\2курс 1семетр\Языки Програирования\lr1>git commit -m "change files 1.txt and 3.txt"
[branch_1 b42dd24] change files 1.txt and 3.txt
 2 files changed, 2 insertions(+), 1 deletion(-)

C:\Users\Acer\Desktop\Учеба\2курс 1семетр\Языки Програирования\lr1>git branch
* branch_1
  branch_2
  main
```

Рисунок 12. Коммит изменений

12. Перешел на ветку branch_2 и также изменил файл 1.txt на текст “My fix in the 1.txt”, изменил файл 3.txt на текст “My fix in the 3.txt”, закоммитив изменения:

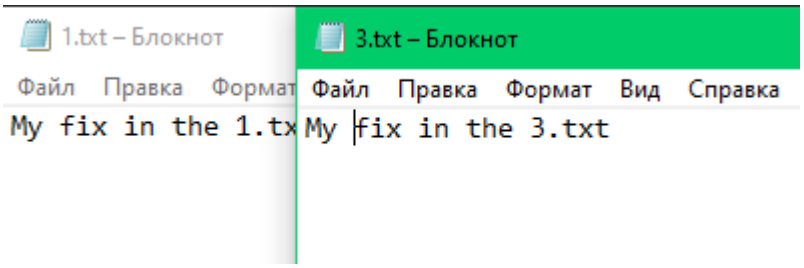

Рисунок 13. Изменение текстовых файлов

```
C:\Users\Acer\Desktop\Учеба\2курс 1семетр\Языки Програирования\lr1>git add 1.txt 3.txt

C:\Users\Acer\Desktop\Учеба\2курс 1семетр\Языки Програирования\lr1>git commit -m "My fix file 1.txt and 3.txt"
[branch_2 90eefa3] My fix file 1.txt and 3.txt
 2 files changed, 2 insertions(+), 1 deletion(-)
```

Рисунок 14. Коммит изменений

13. Слила изменения ветки branch_2 в ветку branch_1:

```
C:\Users\Acer\Desktop\Учеба\2курс 1семетр\Языки Програирования\lr1>git checkout branch_1
Switched to branch 'branch_1'

C:\Users\Acer\Desktop\Учеба\2курс 1семетр\Языки Програирования\lr1>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 15. Слияние веток

14. Решил конфликт файла 1.txt в ручном режиме:

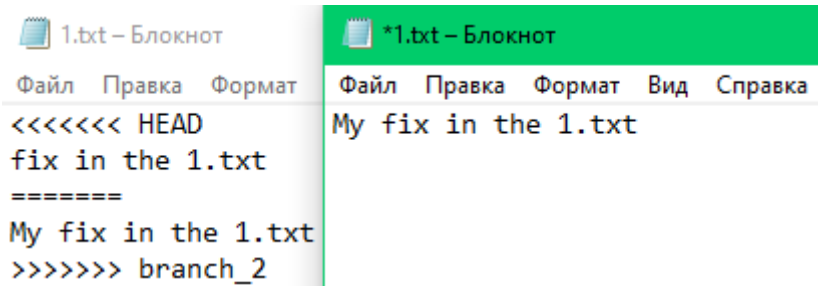

Рисунок 16. Решенние конфликта файлов

15. Отправил ветку branch_1 на GitHub:

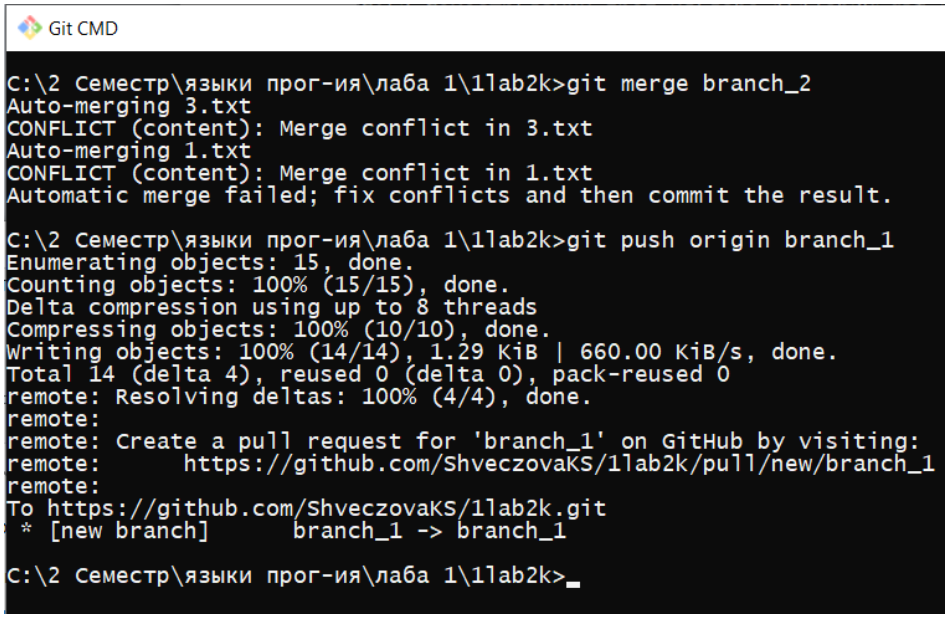

Рисунок 17. Отправление ветки

16. Создала средствами GitHub удаленную ветку branch_3:

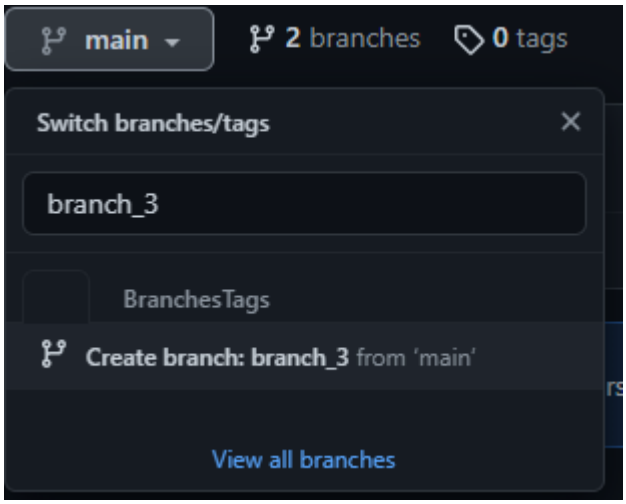

Рисунок 22. Создание удаленной ветки

17. Перешел на ветку branch_3 и добавила файл 2.txt строку "the final fantasy in the 4.txt file":

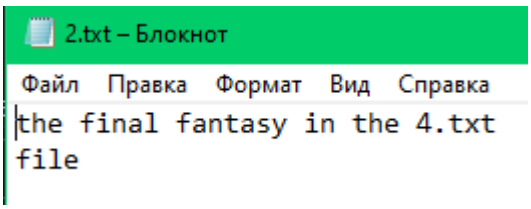

Рисунок 23. Добавление текстового файла

18. Выполнила перемещение ветки master на ветку branch_2 и отправила изменения веток master и branch_2 на GitHub:

```
C:\Users\Acer\Desktop\Учеба\2курс 1семетр\Языки Програирования\lr1>git push origin branch_3
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 4 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 296 bytes | 296.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/danilusikov0913/lr1.git
   6ada000..7d8f21f  branch_3 -> branch_3

C:\Users\Acer\Desktop\Учеба\2курс 1семетр\Языки Програирования\lr1>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:\Users\Acer\Desktop\Учеба\2курс 1семетр\Языки Програирования\lr1>git merge branch_2
Updating cf05b0c..a4c8618
Fast-forward
 1.txt | 2 +-
 3.txt | 1 +
 2 files changed, 2 insertions(+), 1 deletion(-)

C:\Users\Acer\Desktop\Учеба\2курс 1семетр\Языки Програирования\lr1>git status
On branch main
Your branch is ahead of 'origin/main' by 7 commits.
  (use "git push" to publish your local commits)

nothing to commit, working tree clean

C:\Users\Acer\Desktop\Учеба\2курс 1семетр\Языки Програирования\lr1>git push origin main
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/danilusikov0913/lr1.git
```

Рисунок 24. Отправка измененных веток

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. Суть данного указателя можно попытаться объяснить с разных сторон.

Во-первых, HEAD – это указатель на коммит в вашем репозитории, который станет родителем следующего коммита.

Во-вторых, HEAD указывает на коммит, относительно которого будет создана рабочая копия во-время операции checkout . Другими словами, когда вы переключаетесь с ветки на ветку, используя операцию checkout , то в вашем репозитории указатель HEAD будет переключаться между последними коммитами выбираемых вами ветвей.

3. Способы создания веток.

Ветки можно создать с помощью команды “git branch имя_ветки”, и чтобы перейти на неё необходимо использовать команду “git checkout имя_ветки”. Можно же создать ветку и сразу перейти на неё с помощью “git checkout –b имя_ветки”.

4. Как узнать текущую ветку?

С помощью команды git branch высветятся все ветки, текущая будет подкрашена и/или будет со знаком *.

5. Как переключаться между ветками?

Чтобы переходить по веткам, необходимо использовать команду “git checkout имя_ветки”

6. Что такое удаленная ветка?

Это ветка, находящаяся на удалённом сервере GitHub.

7. Что такое ветка отслеживания?

Ветки слежения — это ссылки на определённое состояние удалённых веток. Это локальные ветки, которые напрямую связаны с удалённой веткой. При клонировании репозитория, как правило, автоматически создаётся ветка master, которая следит за origin/master.

8. Как создать ветку отслеживания?

С помощью команды git checkout -b имя_ветки origin/имя_ветки.

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

С помощью команды git push имя_ветки.

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

Git fetch лишь показывает изменения веток на сервере, но не копирует их на локальный репозиторий, в отличие от команды Git pull.

11.Как удалить локальную и удаленную ветки?

Можно удалить ветку на удалённом сервере используя параметр --delete для команды git push .

Локальную ветку можно удалить с помощью команды git branch –d имя_ветки.

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

Git Flow описывает несколько веток для разработки, релизов и взаимодействия между ними. Репозиторий содержит 2 главные ветки:

– master (стандартная ветка)

– develop (ответвляется от основной, в нее добавляется что-то новое и затем сливается с основной)

Как следствие master выступает релизной веткой в этой концепции. В Git Flow мы можем использовать такие типы веток как:

– Feature branches;

– Release branches;

– Hotfix branches;

Минусы git-flow:

– git flow может замедлять работу;

– релизы сложно делать чаще, чем раз в неделю;

– большие функции могут потратить дни на конфликты.

Вывод: были исследованы базовые возможности работы с локальными и удаленными ветками Git.