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 практической работе №1.3

## дисциплины «Программирование на Python»

Выполнил:
Пустяков Андрей Сергеевич
2 курс, группа ИВТ-б-о-22-1,
09.03.01 «Информатика и вычислительная техника (профиль) «Программное обеспечение средств вычислительной техники и автоматизированных систем», очная форма обучения

Ставрополь, 2023 г.

Тема: Основы ветвления Git.

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

Ход работы:

Задание 2.

Создание общедоступного репозитория на GitHub (рис. 1).

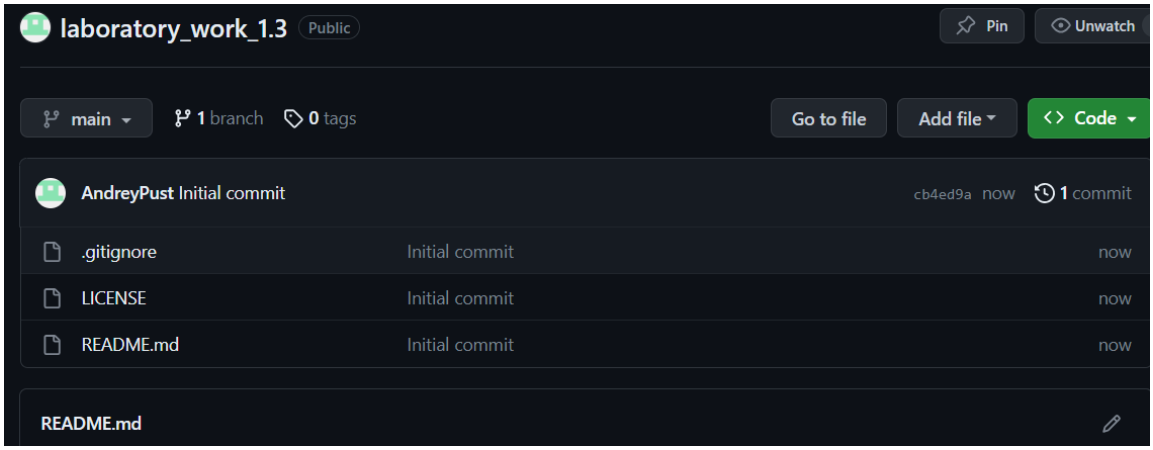


Рисунок 1 – Общедоступный репозиторий.

Задание 3.

Создание трех текстовых файлов 1.txt, 2.txt и 3.txt (рис. 2).

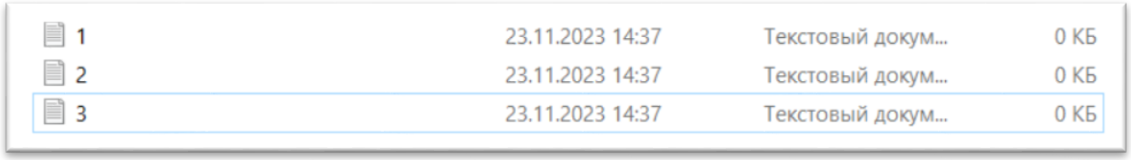


Рисунок 2 – Создание трех текстовых документов.

Задание 4.

Индексация первого файла и создание коммита с комментарием «add 1.txt file» (рис. 3).

```
C:\Users\Andrey\Desktop\пайтон\3 лаба\3 Лабораторная работа\laboratory_work_1.3>git add 1.txt

C:\Users\Andrey\Desktop\пайтон\3 лаба\3 Лабораторная работа\laboratory_work_1.3>git commit -m "add 1.txt file"
[main 6d5c9b3] add 1.txt file
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 1.txt
```

Рисунок 3 – Индексация первого файла.

Задание 5.

Индексация второго и третьего файлов и соответствующий им коммит (рис. 4).

```
C:\Users\Andrey\Desktop\пайтон\3 лаба\3 Лабораторная работа\laboratory_work_1.3>git add 2.txt 3.txt
```

Рисунок 4 – Индексация второго и третьего файлов.

Задание 6.

Перезапись уже сделанного коммита с новым комментарием «add 2.txt and 3.txt» (рис. 5).

```
C:\Users\Andrey\Desktop\пайтон\3 лаба\3 Лабораторная работа\laboratory_work_1.3>git add 2.txt 3.txt

C:\Users\Andrey\Desktop\пайтон\3 лаба\3 Лабораторная работа\laboratory_work_1.3>git commit --amend -m "add 2.txt and 3.t
xt"
[main 33d35cd] add 2.txt and 3.txt
 Date: Wed Nov 29 18:11:33 2023 +0300
 3 files changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 1.txt
 create mode 100644 2.txt
 create mode 100644 3.txt
```

Рисунок 5 – Перезапись сделанного коммита.

Задание 7.

Создание новой ветки с названием «my_first_branch» (рис. 6)

```
C:\Users\Andrey\Desktop\пайтон\3 лаба\3 Лабораторная работа\laboratory_work_1.3>git branch my_first_branch
```

Рисунок 6 – Создание новой ветки.

Задание 8.

Переход на ветку «my_first_branch», создание файла и коммит изменений (рис. 7).

```
C:\Users\Andrey\Desktop\пайтон\3 лаба\3 Лабораторная работа\laboratory_work_1.3>git checkout my_first_branch
Switched to branch 'my_first_branch'

C:\Users\Andrey\Desktop\пайтон\3 лаба\3 Лабораторная работа\laboratory_work_1.3>git log --oneline --decorate
33d35cd (HEAD -> my_first_branch, main) add 2.txt and 3.txt
0d7ad86 delete 1 2 3
9c30128 add 2.txt and 3.txt
6d5c9b3 add 1.txt file
cb4ed9a (origin/main, origin/HEAD) Initial commit

C:\Users\Andrey\Desktop\пайтон\3 лаба\3 Лабораторная работа\laboratory_work_1.3>git add .

C:\Users\Andrey\Desktop\пайтон\3 лаба\3 Лабораторная работа\laboratory_work_1.3>git commit -m "add in_branch.txt in my_f
irst_branch"
[my_first_branch 75e807b] add in_branch.txt in my_first_branch
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 in_branch.txt
```

Рисунок 7 – Переход на новую ветку и добавление в коммит нового файла.

Задание 9.

Возврат на старую ветку «main» (рис. 8).

```
C:\Users\Andrey\Desktop\пайтон\3 лаба\3 Лабораторная работа\laboratory_work_1.3>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)
```

Рисунок 8 – Возврат на старую ветку.

Задание 10.

Создание новой ветки «new_branch» и переход на нее (рис. 9).

```
C:\Users\Andrey\Desktop\пайтон\3 лаба\3 Лабораторная работа\laboratory_work_1.3>git branch new_branch

C:\Users\Andrey\Desktop\пайтон\3 лаба\3 Лабораторная работа\laboratory_work_1.3>git checkout new_branch
Switched to branch 'new_branch'

C:\Users\Andrey\Desktop\пайтон\3 лаба\3 Лабораторная работа\laboratory_work_1.3>git log --oneline --decorate
33d35cd (HEAD -> new_branch, main) add 2.txt and 3.txt
0d7ad86 delete 1 2 3
```

Рисунок 9 – Создание ветки и переход на нее.

Задание 11.

Создание изменений в файле «1.txt» и коммит сделанных изменений (рис. 10).

```
C:\Users\Andrey\Desktop\пайтон\3 лаба\3 Лабораторная работа\laboratory_work_1.3>git add .

C:\Users\Andrey\Desktop\пайтон\3 лаба\3 Лабораторная работа\laboratory_work_1.3>git commit -m "add the change of 1.txt
n new_branch"
[new_branch fd60fc3] add the change of 1.txt in new_branch
 1 file changed, 1 insertion(+)
```

Рисунок 10 – Коммит в новой ветке.

Задание 12.

Переход на начальную ветку «main», слияние веток «main» и «my_first_branch», слияние веток «main» и «new_branch» (рис. 11).

```
C:\Users\Andrey\Desktop\пайтон\3 лаба\3 Лабораторная работа\laboratory_work_1.3>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:\Users\Andrey\Desktop\пайтон\3 лаба\3 Лабораторная работа\laboratory_work_1.3>git merge my_first_branch
Updating 33d35cd..75e807b
Fast-forward
 in_branch.txt | 0
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 in_branch.txt

C:\Users\Andrey\Desktop\пайтон\3 лаба\3 Лабораторная работа\laboratory_work_1.3>git merge new_branch
Merge made by the 'ort' strategy.
 1.txt | 1 +
 1 file changed, 1 insertion(+)
```

Рисунок 11 – Слияние веток.

Задание 13.

Удаление веток «my_first_branch» и «new_branch» (рис. 12).

```
C:\Users\Andrey\Desktop\пайтон\3 лаба\3 Лабораторная работа\laboratory_work_1.3>git branch -d my_first_branch
Deleted branch my_first_branch (was 75e807b).

C:\Users\Andrey\Desktop\пайтон\3 лаба\3 Лабораторная работа\laboratory_work_1.3>git branch -d new_branch
Deleted branch new_branch (was fd60fc3).
```

Рисунок 12 – Удаление ранее созданных веток.

Задание 14.

Создание новых веток «branch_1» и «branch_2» (рис. 13).

```
C:\Users\Andrey\Desktop\пайтон\3 лаба\3 Лабораторная работа\laboratory_work_1.3>git branch branch_1

C:\Users\Andrey\Desktop\пайтон\3 лаба\3 Лабораторная работа\laboratory_work_1.3>git branch branch_2
```

Рисунок 13 – Создание новых веток.

Задание 15.

Переход в ветку «branch_1», создание изменений в файле «1.txt», создание изменений в файле «3.txt», коммит данных изменений (рис. 14).

```
C:\Users\Andrey\Desktop\пайтон\3 лаба\3 Лабораторная работа\laboratory_work_1.3>git checkout branch_1
Switched to branch 'branch_1'

C:\Users\Andrey\Desktop\пайтон\3 лаба\3 Лабораторная работа\laboratory_work_1.3>git add .

C:\Users\Andrey\Desktop\пайтон\3 лаба\3 Лабораторная работа\laboratory_work_1.3>git commit -m "fix in the branch_1"
[branch_1 f762bf7] fix in the branch_1
 2 files changed, 2 insertions(+), 1 deletion(-)
```

Рисунок 14 – Коммит в новой ветке.

Задание 16.

Переход на ветку «branch_2», создание изменений других в файле «1.txt» и «3.txt», коммит данных изменений (рис. 15).

```
C:\Users\Andrey\Desktop\пайтон\3 лаба\3 Лабораторная работа\laboratory_work_1.3>git checkout branch_2
Switched to branch 'branch_2'

C:\Users\Andrey\Desktop\пайтон\3 лаба\3 Лабораторная работа\laboratory_work_1.3>git add .

C:\Users\Andrey\Desktop\пайтон\3 лаба\3 Лабораторная работа\laboratory_work_1.3>git commit -m "add fix in 1.txt and 2.tx
t for branch_2"
[branch_2 a4cb76f] add fix in 1.txt and 2.txt for branch_2
 2 files changed, 2 insertions(+), 1 deletion(-)
```

Рисунок 15 – Коммит во второй ветке.

Задание 17.

Слияние изменений в ветке «branch_2» в ветку «branch_1» (а точнее попытка слияния веток и приостановление процесса до устранения конфликта слияния) (рис. 16).

```
C:\Users\Andrey\Desktop\пайтон\3 лаба\3 Лабораторная работа\laboratory_work_1.3>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.
```

Рисунок 16 – Попытка слияния двух веток.

Задание 18.

Разрешения конфликта файла «1.txt» в ручном режиме (рис. 17).

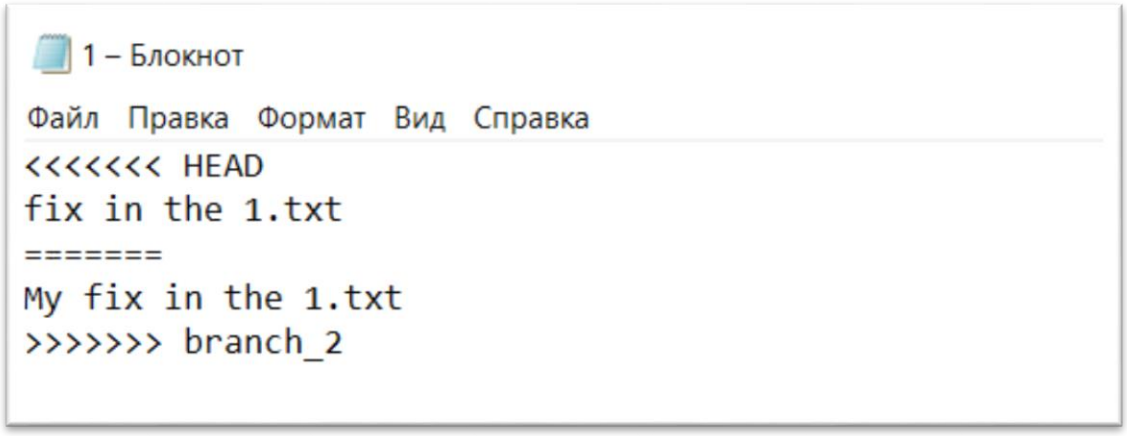


Рисунок 17 – Разрешение конфликта слияния вручную.

Разрешение конфликта файла «3.txt» при помощи команды «git metgetool» и утилиты «vimdiff» (рис. 18).

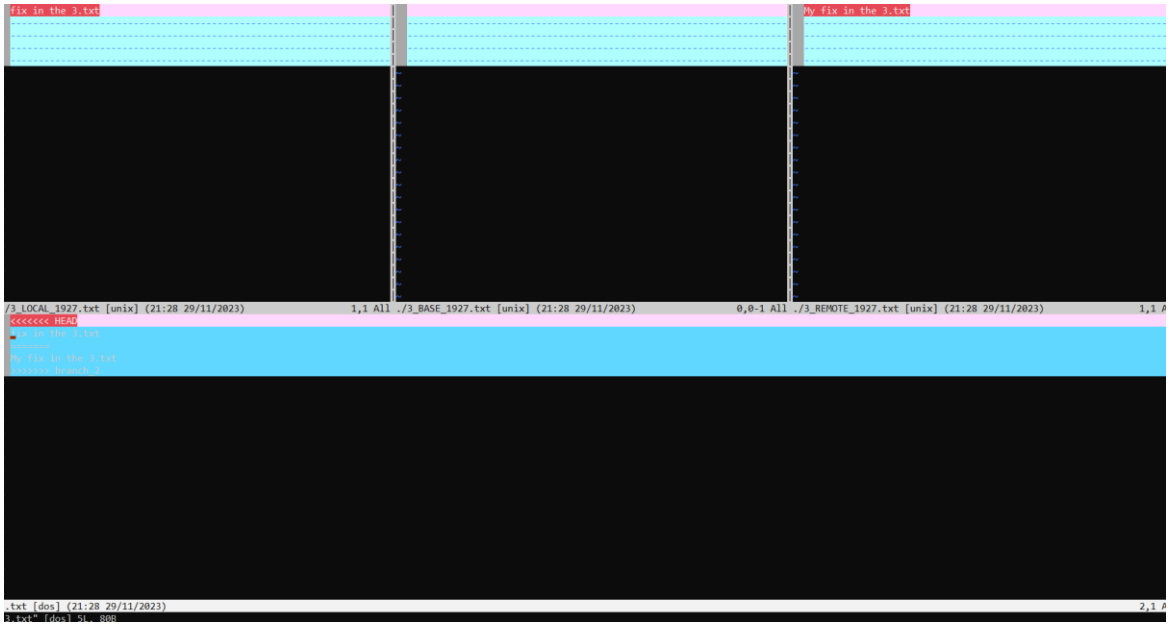

Рисунок 18 – Разрешение конфликта с использованием утилиты.

Задание 19.

Отправка ветки «branch_1» на GitHub (рис. 19).

```
C:\Users\Andrey\Desktop\пайтон\3 лаба\3 Лабораторная работа\laboratory_work_1.3>git push origin branch_1
Enumerating objects: 27, done.
Counting objects: 100% (27/27), done.
Delta compression using up to 8 threads
Compressing objects: 100% (18/18), done.
Writing objects: 100% (26/26), 2.27 KiB | 1.14 MiB/s, done.
Total 26 (delta 9), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (9/9), done.
remote:
remote: Create a pull request for 'branch_1' on GitHub by visiting:
remote:      https://github.com/AndreyPust/laboratory_work_1.3/pull/new/branch_1
remote:
To https://github.com/AndreyPust/laboratory_work_1.3.git
 * [new branch]      branch_1 -> branch_1
```

Рисунок 19 – Отправка ветки в удаленный репозиторий.

Задание 20.

Создание удаленной новой ветки «branch_3» на сервисе «GitHub» (рис. 20).

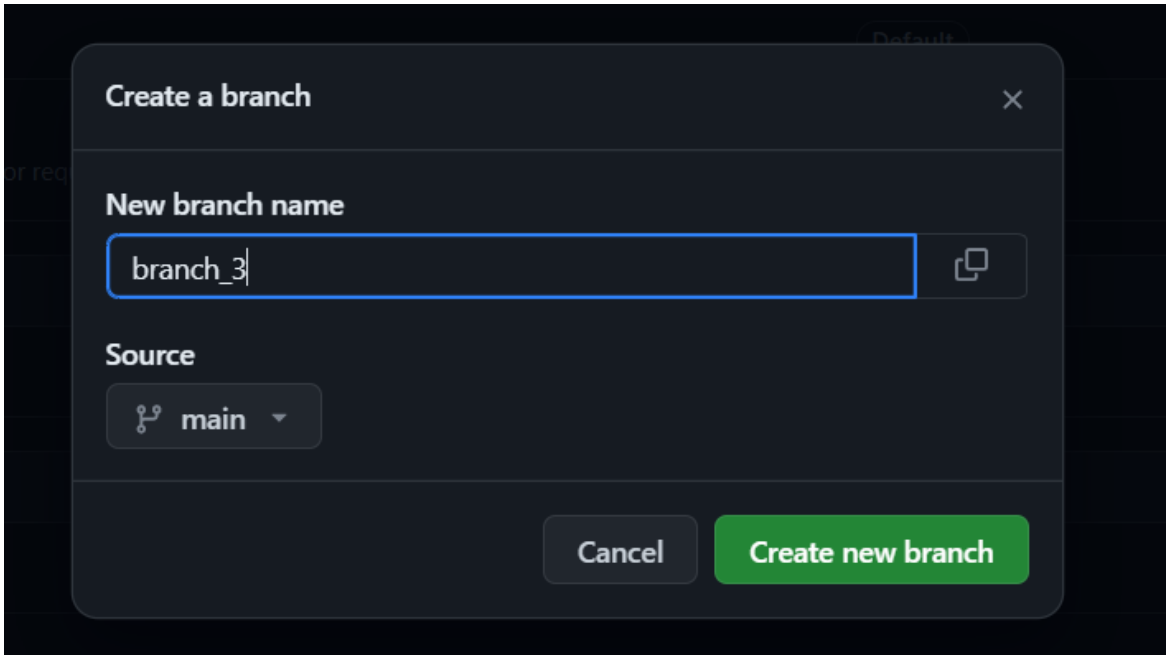


Рисунок 20 – Создание новой ветки на GitHub.

Задание 21.

Создание в локальном репозитории новой ветки отслеживания для удаленной ветки «branch_3» (рис. 21).

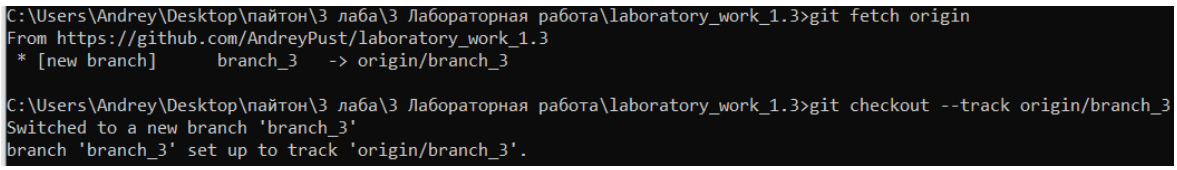


Рисунок 21 – Создание ветки слежения.

Задание 22.

Переход на новую ветку «branch_3» и создание нового файла с изменениями (рис. 22).

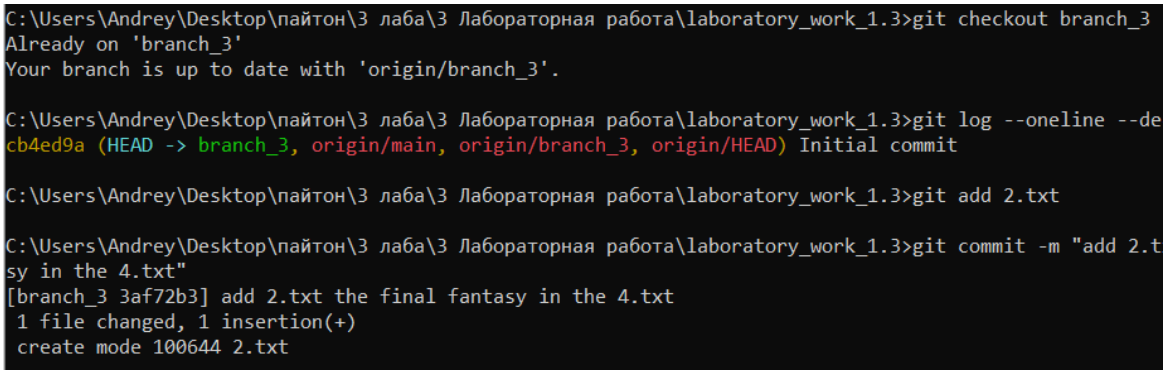


Рисунок 22 – Изменения в новой ветке.

Задание 23.

Перемещение ветки «main» на ветку «branch_2» (рис. 23).

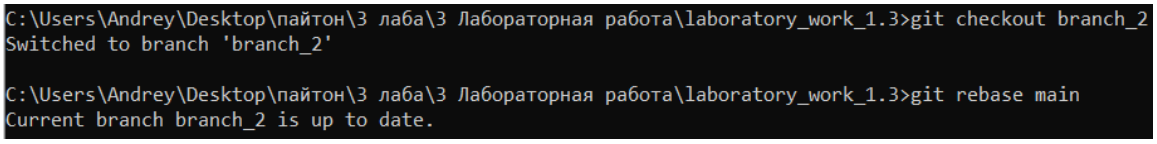


Рисунок 23 – Использование команды «git rebase».

Задание 24.

Отправка изменений веток «main» и «branch_2» на GitHub (рис. 24).

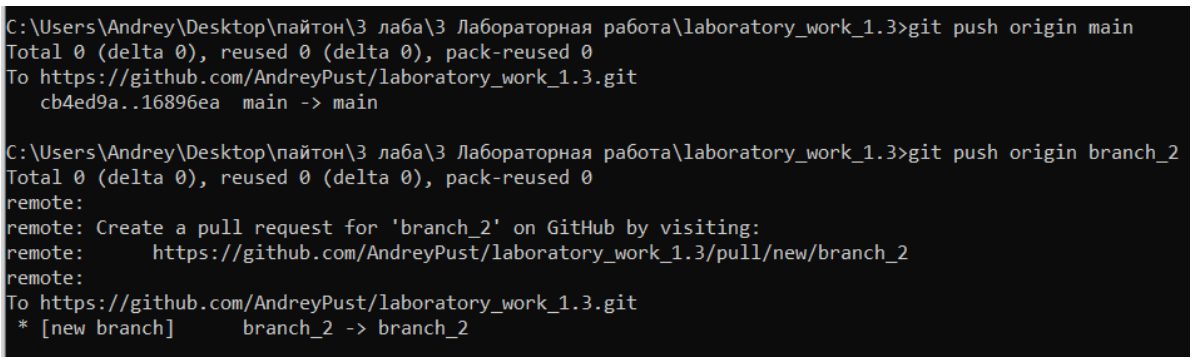


Рисунок 23 – Отправка веток в удаленный репозиторий.

Ответы на контрольные вопросы:

1. Ветка – это отклонение от основной линии разработки. Ветка в Git – это простой перемещаемый указатель на один из коммитов.

2. HEAD в Git – это указатель, задача которого ссылаться на определенный коммит в репозитории. Он может стать родителем следующего коммита, и он указывает на коммит, относительно которого будет создана рабочая копия (указатель HEAD перемещается между коммитами во время операции «checkout»).

3. Для того, чтобы создать ветку в Git, нужно создать новый указатель на текущий коммит с помощью команды «git branch», а переключиться на эту ветку можно с помощью команды «git checkout …». Также для создания ветки можно воспользоваться командой «git checkout» с определенным ключом «-b» и указав еще имя новой ветки (таким образом мы создадим новую ветку и сразу же переключимся на нее). Создать ветку можно также на удаленном репозитории, а затем создать на нее указатель (ветку слежения) в локальном репозитории.

4. Для того, чтобы узнать, текущую ветку необходимо использовать команду «git branch», в результате на экран будет выведена информация о всех локальных ветках и на против текущей будет стоять знак «*». Также можно воспользоваться командой «git symbolic-ref --short HEAD», в результате будет выведена информация только о текущей ветке. Можно использовать команду

«git log --oneline --decoration», таким образом выведется информация о коммитах и указателе «HEAD», который указывает на текущую ветку.

5. Чтобы переключаться между ветками нужно переместить указатель HEAD на нужную ветку можно с помощью команды «git checkout *», где вместо «*» указывается имя необходимой ветки соответственно.

6. Удаленная ветка – это ветка, которая существует в удалённом репозитории. При клонировании репозитория, все ветки, которые находились на удалённом репозитории, также копируются на ваш локальный репозиторий. Однако они могут отличаться от локальных веток, потому что удалённая ветка отслеживает состояние в удалённом репозитории.

7. Ветка отслеживания – это ссылки на определенное состояние удаленных веток, они же локальные ветки, которые нельзя перемещать, но они перемещаются автоматически при подключении Git к удаленному репозиторию. Ветки отслеживания показывают, где ветки в удаленных репозиториях находились во время последнего подключения.

8. Для того, чтобы создать ветку отслеживания необходимо создать локальную ветку, которая будет отслеживать удаленную ветку. Для этого необходимо получить информацию из удаленного репозитория используя команду «git fetch», а затем перейти и создать ветку слежения используя команду с ключом «git checkout –track *».

9. Для того, чтобы отправить изменения из локальной ветки в удаленную нужно обладать правами на запись на этом сервере, нужно явно указать те ветки, которые мы хотим отправить. Это можно сделать с помощью команды «git push origin …», где мы укажем имя ветки, которую хотим отправить (желательно обновить данные об удаленном репозитории с помощью «git fetch»).

10. Отличие команд «git fetch» и «git pull» заключается в том, команда «git fetch» получает изменения из удаленного репозитория, но не меняет локальную директорию, а команда «git pull» автоматически интегрирует полученные изменения в локальный репозиторий, старается все слить.

11. Для удаления локальной ветки необходимо перейти на другую ветку и выполнить команду «git branch -d *» с именем соответствующей ветки, таким образом это будет означать, что работа с веткой закончена. Для удаления удаленной ветки можно использовать команду «git push» с опцией «--delete».

12. Git-flow — альтернативная модель ветвления Git, в которой используются функциональные и несколько основных веток. В Git-flow используются следующие типы веток:

Функциональные ветки - создаются на основе последней ветки разработки. Когда работа с функцией завершена, ветка сливается с develop веткой. Создаётся при помощи «git flow feature start» (название ветки) (если есть библиотека) или «git checkout -b» (название ветки) от ветки разработки. Если необходимо закончить работу с веткой, то при наличии библиотеки можно воспользоваться командой «git flow feature finish» (название ветки) или командами «git checkout develop» и «git merge» (название ветки).

Ветки выпуска - когда в ветке develop оказывается достаточно функций для выпуска, от ветки разработки создается ветка выпуска. Создание этой ветки запускает следующий цикл релиза, и с этого момента новые функции добавить больше нельзя — допускается лишь исправление ошибок, 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. Для создания такой ветки можно воспользоваться командами «git checkout develop» и «git checkout -b release/(версия)» или, если есть библиотека, можно воспользоваться командой «git flow release start версия». Для завершения работы можно использовать команды «git checkout main» и «git merge release/(версия)», или, если есть библиотека, можно воспользоваться «git flow release finish (версия)».

Ветки исправления - используются для быстрого внесения исправлений в рабочие релизы. Они создаются на основе «main», а не «develop». Это единственная ветка, которую нужно обязательно создавать напрямую от «main». Как только исправление завершено, эту ветку следует слить с «main» и «develop» (или текущей веткой «release»), а ветке «main» присвоить обновленный номер версии. Для создания такой ветки можно воспользоваться командами «git checkout main» и «git checkout -b (название ветки)».

Также присутствуют ветка разработки «develop», в ней хранится полная история проекта. Её требуется создать самостоятельно и отправить на сервер вручную, или же можно использовать библиотеку «git-flow» и воспользоваться командой «git flow init» и главная ветка «main», в ней хранится сокращённая история проекта).

Минусы Git-flow – такие долгосрочные функциональные ветки требуют тесного взаимодействия разработчиков при слиянии и создают повышенный риск отклонения от магистральной ветки. В них также могут присутствовать конфликтующие обновления.

13. Работа в сервисе «GitLab» с ветками осуществляется примерно также как и в сервисе «GitHub». Удаленные ветки можно создавать, сливать, создавать ветки слежения, отправлять явно ветки на репозиторий, просмотреть ветки в сервисе, получить изменения из репозитория.

Вывод: в ходе выполнения лабораторной работы были рассмотрены основные моменты работы с ветками в Git и рассмотрены основы ветвления. Была изучена работа с ветками как локальными в локальных репозиториях, так и с удаленными ветками на удаленных репозиториях (были изучены способы создания ветки слежения). Были изучены способы создания веток, переход между ветками, способы удаления веток. Были изучены способы слияния веток, а также ручное устранение конфликтов слияния и устранение конфликтов слияния с использованием утилит. Были изучены способы перемещения одной ветки в другую. Были изучены методы отправки

локальной ветки на удаленный репозиторий и получения изменений с удаленного репозитория на локальный с автоматическим интегрированием и с возможностью слить ветки позже. Были изучены способы получения информации о текущих и существующих веток в репозиториях.