МИНИСТЕРСТВО НАУКИ ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

ФГАОУ ВО «СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт цифрового развития

Кафедра инфокоммуникаций.

ОТЧЁТ

по лабораторной работе №3

Дисциплина: «Программирование на Python»

Тема: «Основы ветвления Git.»

Выполнил:

студент 2 курса группы ИВТ-б-о-22-1

Бабенко Артём Тимофеевич

Проверил:

Доцент кафедры инфокоммуникаций

Воронкин Р.А

Работа защищена с оценкой: _____

Ставрополь, 2023

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

Ход работы:

```
D:\git\lab3>git commit --amend -m "add 2.txt and 3.txt"
[main 7771167] add 2.txt and 3.txt
 Date: Thu Nov 16 16:09:09 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 1. Команда git commit с параметром --amend

```
D:\git\lab3>git branch my_first_branch

D:\git\lab3>git log
commit 7771167d582a67fbae27702b27df04e42cf12306 (HEAD -> main, my_first_branch)
Author: Babenko Artem <kostydu2342@gmail.com>
Date:   Thu Nov 16 16:09:09 2023 +0300

    add 2.txt and 3.txt
```

Рисунок 2. Команда git branch, создание ветки

```
D:\git\lab3>git checkout my_first_branch
Switched to branch 'my_first_branch'
```

Рисунок 3. Команда checkout, перемещение на новую ветку

```
D:\git\lab3>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)
```

Рисунок 4. Команда checkout, перемещение обратно в главную ветку

```
D:\git\lab3>git merge my_first_branch
Updating 7771167..9da6543
Fast-forward
 in_branch.txt | 0
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 in_branch.txt

D:\git\lab3>git merge new_branch
Merge made by the 'recursive' strategy.
 1.txt | 1 +
 1 file changed, 1 insertion(+)

D:\git\lab3>
```

Рисунок 5. Команда git merge, слияние веток.

```
D:\Gitlab\lab3>git branch -d new_branch
Deleted branch new_branch (was f5a1fea).

D:\Gitlab\lab3>git branch -d my_first_branch
Deleted branch my_first_branch (was 9da6543).
```

Рисунок 6. Команда git branch -d, удаление веток.

```
D:\Gitlab\lab3>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 7. Команда merge, конфликты слияния.

```
D:\Gitlab\lab3>git status
On branch branch_1
You have unmerged paths.
  (fix conflicts and run "git commit")
  (use "git merge --abort" to abort the merge)

Changes to be committed:
        modified:   1.txt

Unmerged paths:
  (use "git add <file>..." to mark resolution)
        both modified:   3.txt
```

Рисунок 8. Команда git status первый конфликт решён вручную

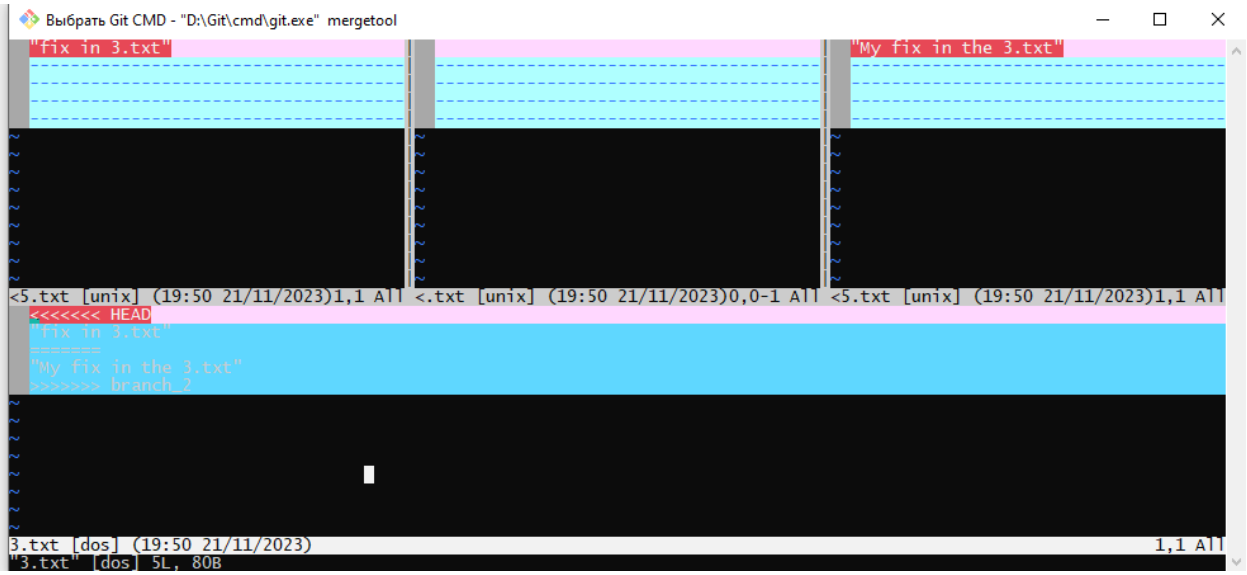

Рисунок 9. Команда mergetool, решение конфликта.

```
D:\Gitlab\lab3>git push origin branch_1
Enumerating objects: 23, done.
Counting objects: 100% (23/23), done.
Delta compression using up to 6 threads
Compressing objects: 100% (14/14), done.
Writing objects: 100% (22/22), 1.78 KiB | 259.00 KiB/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/lorianss/lab3/pull/new/branch_1
remote:
To https://github.com/lorianss/lab3.git
 * [new branch]      branch_1 -> branch_1
```

Рисунок 10. Git push, отправление ветки на удалённый репозеторий.

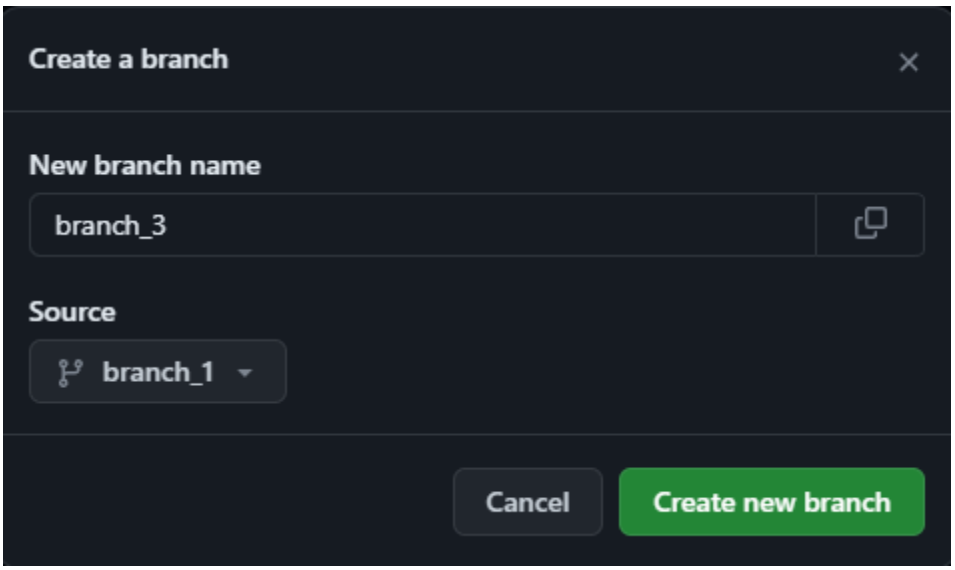

Рисунок 11. Создание ветки в удалённом репозетории.

```
D:\Gitlab\lab3>git rebase main
Current branch branch_2 is up to date.

D:\Gitlab\lab3>git status
On branch branch_2
nothing to commit, working tree clean
```

Рисунок 12. Команда git rebase, перемещение главной ветки на вторую.

Вывод: исследовал базовые возможности по работе с локальными и удаленными ветками Git.

Контрольные вопросы:

Что такое ветка?

Ветка – это отклонение от основной линии разработке, и работа в её “копии” без вмешательства в основную линию.

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, относительного 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.

Способы создания веток.

Можно это сделать командой git branch (название ветки)

Как узнать текущую ветку?

При помощи простой команды git log , которая покажет вам куда указывают указатели веток. Так же нужна опция --decorate .

Как переключаться между ветками?

Для переключения на существующую ветку можно использовать команду git checkout (название ветки)

Что такое удаленная ветка?

Удалённые ссылки — это ссылки (указатели) в ваших удалённых репозиториях, включая ветки, теги и так далее

Что такое ветка отслеживания?

Ветки слежения — 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.

Как создать ветку отслеживания?

Получение локальной ветки из удалённой ветки автоматически создаёт то, что называется “веткой слежения” (а ветка, за которой следит локальная называется “upstream branch

При клонировании репозитория, как правило, автоматически создаётся ветка master , которая следит за origin/master . Однако, при желании вы можете настроить отслеживание и других веток — следить за ветками на других серверах или отключить слежение за веткой master . это можно с помощью команды git checkout -b / . Это часто используемая команда, поэтому Git предоставляет сокращённую форму записи в виде флага --track :

> git checkout --track origin/serverfix

Как отправить изменения из локальной ветки в удаленную ветку?

git push <remote> <branch>

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.

Команда git pull , в большинстве случаев является комбинацией команд git fetch , за которой непосредственно следует команда git merge .

Как удалить локальную и удаленную ветки?

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

Для удаления локальной ветки можно выполнить команду git branch с параметром –d <название ветки

Изучить модель ветвления git- Какие основные типы веток присуствуют в модели git-flow? Как организована работа с ветками в модели git-flow? В чем недостатки git-flow?

Ветка разработки и главная ветка

Функциональные ветки (feature)

Ветки выпуска (release)

Ветки исправления (hotfix)

Схема работы

1 Из ветки main создается ветка develop.

2. Из ветки develop создается ветка release.

3. Из ветки develop создаются ветки feature.

4. Когда работа над веткой feature завершается, она сливается в ветку develop.

5. Когда работа над веткой release завершается, она сливается с ветками develop и main.

6. Если в ветке main обнаруживается проблема, из main создается ветка hotfix.

7. Когда работа над веткой hotfix завершается, она сливается с ветками develop и main.

Недостатки:
Git Flow может замедлять работу, когда приходится ревьювить большие пулл реквесты, когда вы пытаетесь выполнить итерацию быстро. Релизы сложно делать чаще, чем раз в неделю. Большие функции могут потратить дни на мерж и резолв конфликтов и форсировать несколько циклов тестирования.

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.