

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

Институт цифрового развития  
Кафедра инфокоммуникаций

ОТЧЁТ

по лабораторной работе №3

**Дисциплина: «Программирование на Python»**  
**Тема: «Основы ветвления Git»**

Выполнил:

Дзуев Альберт Мухамедович  
2 курс, группа ИВТ-б-о-22-1,  
09.03.01 «Информатика и  
вычислительная техника»,  
направленность (профиль)  
«Программное обеспечение средств  
вычислительной  
техники и автоматизированных систем  
», очная форма обучения

---

(подпись)

Руководитель практики:

Воронкин Роман Александрович

---

(подпись)

Отчет защищен с оценкой \_\_\_\_\_ Дата защиты \_\_\_\_\_  
Ставрополь, 2023 г.

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

Порядок выполнения работы:

1. Создал новый репозиторий и клонировал его
2. Создал 3 файла: 1.txt, 2.txt и 3.txt. Проиндексировал первый файл и сделал коммит, потом проиндексировал оставшиеся и перезаписал уже сделанный коммит с новым именем:

3. Создал новую Ветку и перешел на нее, после чего создал новый файл, закоммитил изменения и вернулся на ветку main:

```
dzuev@LAPTOP-QEQHF3D5 MINGW32 ~/OneDrive/Рабочий стол/Python-3 (main)
$ git branch my_first_branch
fatal: a branch named 'my_first_branch' already exists

dzuev@LAPTOP-QEQHF3D5 MINGW32 ~/OneDrive/Рабочий стол/Python-3 (main)
$ git checkout my_first_branch
Switched to branch 'my_first_branch'

dzuev@LAPTOP-QEQHF3D5 MINGW32 ~/OneDrive/Рабочий стол/Python-3 (my_first_branch)
$ touch in_branch.txt

dzuev@LAPTOP-QEQHF3D5 MINGW32 ~/OneDrive/Рабочий стол/Python-3 (my_first_branch)
$ git add in_branch.txt.txt

dzuev@LAPTOP-QEQHF3D5 MINGW32 ~/OneDrive/Рабочий стол/Python-3 (my_first_branch)
$ git commit -m "in_branch.txt"
[my_first_branch c203ede] in_branch.txt
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 in_branch.txt.txt

dzuev@LAPTOP-QEQHF3D5 MINGW32 ~/OneDrive/Рабочий стол/Python-3 (my_first_branch)
$ 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 1. Создание и перемещение между ветками

4. Создал и сразу же перешел на новую ветку, добавил строку в файл 1.txt, закоммитил изменения:

```
dzuev@LAPTOP-QEQHF3D5 MINGW32 ~/OneDrive/Рабочий стол/Python-3 (main)
$ git branch new_branch

dzuev@LAPTOP-QEQHF3D5 MINGW32 ~/OneDrive/Рабочий стол/Python-3 (main)
$ git checkout new_branch
Switched to branch 'new_branch'

dzuev@LAPTOP-QEQHF3D5 MINGW32 ~/OneDrive/Рабочий стол/Python-3 (new_branch)
$ git add 1.txt.txt

dzuev@LAPTOP-QEQHF3D5 MINGW32 ~/OneDrive/Рабочий стол/Python-3 (new_branch)
$ git commit -m "update 1.txt"
[new_branch 4ab8941] update 1.txt
1 file changed, 1 insertion(+), 1 deletion(-)

dzuev@LAPTOP-QEQHF3D5 MINGW32 ~/OneDrive/Рабочий стол/Python-3 (new_branch)
$
```

Рисунок 2. Создание новой ветки и коммит обновленного файла

5. Переключился на ветку main, слил ее с my\_first\_branch, но не сразу получилось выполнить слияние с new\_branch:

```
dzuev@LAPTOP-QEQHF3D5 MINGW32 ~/OneDrive/Рабочий стол/Python-3 (main|MERGING)
$ git merge new_branch
fatal: You have not concluded your merge (MERGE_HEAD exists).
Please, commit your changes before you merge.

dzuev@LAPTOP-QEQHF3D5 MINGW32 ~/OneDrive/Рабочий стол/Python-3 (main|MERGING)
$ git add .

dzuev@LAPTOP-QEQHF3D5 MINGW32 ~/OneDrive/Рабочий стол/Python-3 (main|MERGING)
$ git commit -m "my_first_branch merge"
[main 6271dcd] my_first_branch merge

dzuev@LAPTOP-QEQHF3D5 MINGW32 ~/OneDrive/Рабочий стол/Python-3 (main)
$ git merge new_branch
Already up to date.
```

Рисунок 3. Получилось выполнить слияние с new\_branch

6. Создал новые ветки, перешел в ветку branch\_1 и изменил файлы 1.txt, 3.txt, закоммитил изменения:

```

dzuev@LAPTOP-QEQHF3D5 MINGW32 ~/OneDrive/Рабочий стол/Python-3 (main)
$ git branch branch_1

dzuev@LAPTOP-QEQHF3D5 MINGW32 ~/OneDrive/Рабочий стол/Python-3 (main)
$ git branch branch_2

dzuev@LAPTOP-QEQHF3D5 MINGW32 ~/OneDrive/Рабочий стол/Python-3 (main)
$ git checkout branch_1
Switched to branch 'branch_1'

dzuev@LAPTOP-QEQHF3D5 MINGW32 ~/OneDrive/Рабочий стол/Python-3 (branch_1)
$ git add 1.txt.txt 3.txt.txt

dzuev@LAPTOP-QEQHF3D5 MINGW32 ~/OneDrive/Рабочий стол/Python-3 (branch_1)
$ git commit -m "fix in the txt"
[branch_1 d1afce7] fix in the txt
2 files changed, 2 insertions(+), 1 deletion(-)

```

Рисунок 4. Создал новые ветки, закоммитил изменения файлов в первой

7. Перешел во вторую созданную ветку и изменил там те же самые файлы, закоммитил изменения:

```

dzuev@LAPTOP-QEQHF3D5 MINGW32 ~/OneDrive/Рабочий стол/Python-3 (branch_1)
$ git checkout branch_2
Switched to branch 'branch_2'

dzuev@LAPTOP-QEQHF3D5 MINGW32 ~/OneDrive/Рабочий стол/Python-3 (branch_2)
$ git add 1.txt.txt 3.txt.txt

dzuev@LAPTOP-QEQHF3D5 MINGW32 ~/OneDrive/Рабочий стол/Python-3 (branch_2)
$ git commit -m "My fix in the txt"
[branch_2 68300c3] "My fix in the txt"
2 files changed, 2 insertions(+), 1 deletion(-)

```

Рисунок 5. Закоммитил новые изменения в файлах в ветке branch\_2

8. Перешел в ветку branch\_1 и попытался слить в нее branch\_2:

```
dzuev@LAPTOP-QEQHF3D5 MINGW32 ~/OneDrive/Рабочий стол/Python-3 (branch_2)
$ git checkout branch_1
Switched to branch 'branch_1'

dzuev@LAPTOP-QEQHF3D5 MINGW32 ~/OneDrive/Рабочий стол/Python-3 (branch_1)
$ git merge branch_2
Auto-merging 1.txt.txt
CONFLICT (content): Merge conflict in 1.txt.txt
Auto-merging 3.txt.txt
CONFLICT (content): Merge conflict in 3.txt.txt
Automatic merge failed; fix conflicts and then commit the result.
```

Рисунок 6. Попытка слияния завершилась неудачно

9. Исправил конфликты, один вручную: изменил файл, другой используя git mergetool:

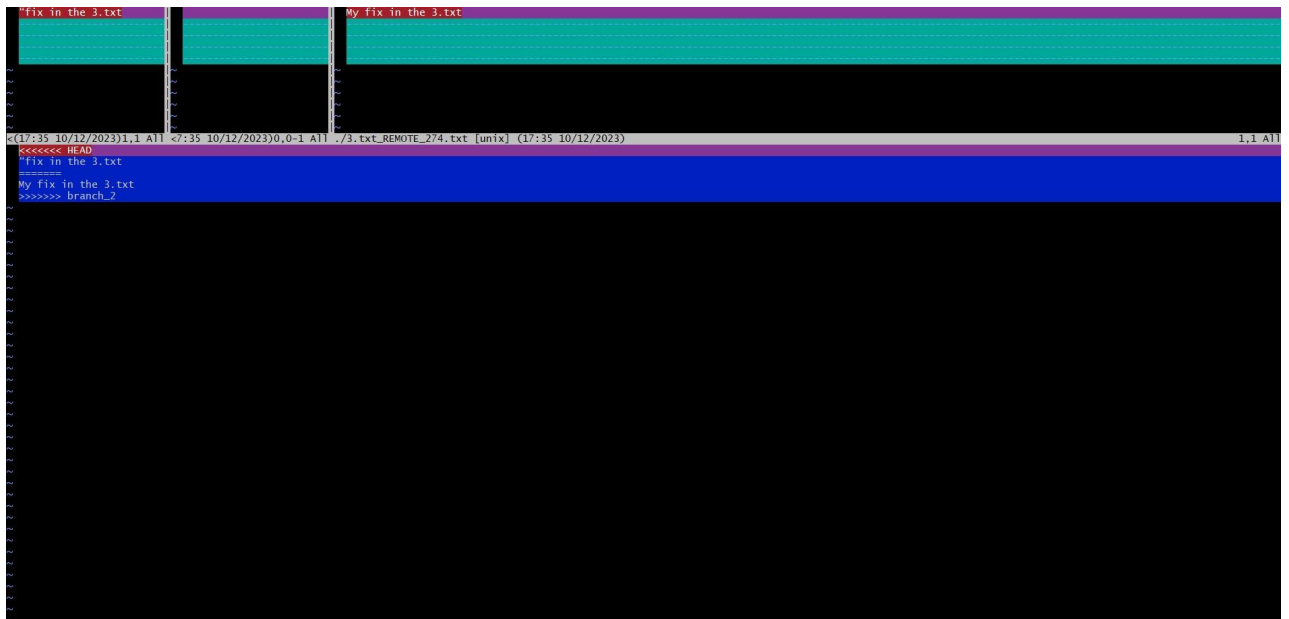


Рисунок 7. Инструмент tortoisemerge

10. Отправил ветку branch\_1 на удаленный репозиторий:

```
dzuev@LAPTOP-QEQHF3D5 MINGW32 ~/OneDrive/Рабочий стол/Python-3 (branch_1)
$ git push origin branch_1
Enumerating objects: 22, done.
Counting objects: 100% (22/22), done.
Delta compression using up to 8 threads
Compressing objects: 100% (14/14), done.
Writing objects: 100% (21/21), 1.65 KiB | 337.00 KiB/s, done.
Total 21 (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/dzeall00/Python-3/pull/new/branch_1
remote:
To https://github.com/dzeall00/Python-3.git
 * [new branch]      branch_1 -> branch_1
```

Рисунок 8. Отправка ветки

11. Создал ветку средствами GitHub и перенес ее в локальный репозиторий:

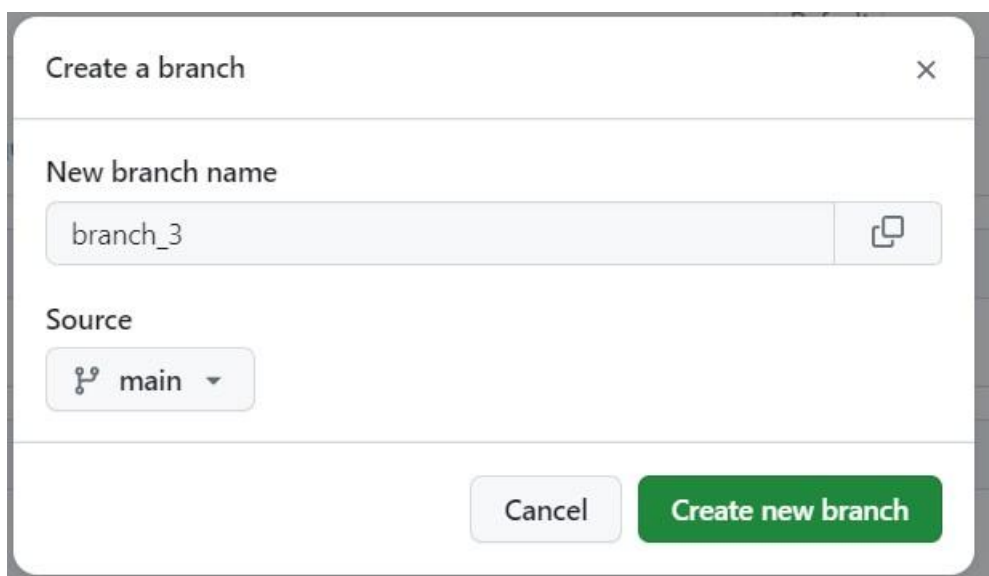




Рисунок 9. Создание ветки

```
dzuev@LAPTOP-QEQHF3D5 MINGW32 ~/OneDrive/Рабочий стол/Python-3 (branch_1)
$ git pull
From https://github.com/dzeall00/Python-3
* [new branch]      branch_3 -> origin/branch_3
There is no tracking information for the current branch.
Please specify which branch you want to merge with.
See git-pull(1) for details.

    git pull <remote> <branch>

If you wish to set tracking information for this branch you can do so with:

    git branch --set-upstream-to=origin/<branch> branch_1
```

Рисунок 10. Успешный перенос ветки

12. Добавил в файл 2.txt новую строку и закоммитил:

```
dzuev@LAPTOP-QEQHF3D5 MINGW32 ~/OneDrive/Рабочий стол/Python-3 (branch_3)
$ git checkout branch_2
Switched to branch 'branch_2'

dzuev@LAPTOP-QEQHF3D5 MINGW32 ~/OneDrive/Рабочий стол/Python-3 (branch_2)
$ git rebase main
Current branch branch_2 is up to date.

dzuev@LAPTOP-QEQHF3D5 MINGW32 ~/OneDrive/Рабочий стол/Python-3 (branch_2)
$ 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)

dzuev@LAPTOP-QEQHF3D5 MINGW32 ~/OneDrive/Рабочий стол/Python-3 (main)
$ git push origin main
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/dzeall00/Python-3.git
7604653..6271dcd main -> main
```

Рисунок 11. Коммит файла 2.txt

Ответы на контрольные вопросы:

1. Что такое ветка?

Ветка в Git — это простой перемещаемый указатель на один из КОММИТОВ.

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.

3. Способы создания веток.



Чтобы создать новую ветку, необходимо использовать команду `git branch`.

Чтобы создать ветку и сразу переключиться на нее, можно использовать команду `git checkout -b`.

#### 4. Как узнать текущую ветку?

Увидеть текущую ветку можно при помощи простой команды `git log`, которая покажет куда указывают указатели веток. Эта опция называется `--decorate`. HEAD будет стоять рядом с текущей веткой. Также можно использовать `git branch -v`, рядом с текущей веткой будет значек `*`.

#### 5. Как переключаться между ветками?

Для переключения на существующую ветку выполните команду `git checkout`.

#### 6. Что такое удаленная ветка?

Удалённые ссылки — это ссылки (указатели) в ваших удалённых репозиториях, включая ветки, теги и так далее. Полный список удалённых ссылок можно получить с помощью команды `git ls-remote <remote>` или

команды `git remote show <remote>` для получения удалённых веток и дополнительной информации.

#### 7. Что такое ветка отслеживания?

Ветки слежения — это ссылки на определённое состояние удалённых веток. Это локальные ветки, которые нельзя перемещать; Git перемещает их автоматически при любой коммуникации с удалённым репозиторием, чтобы гарантировать точное соответствие с ним.

#### 8. Как создать ветку отслеживания?

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

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

Когда вы хотите поделиться веткой, вам необходимо отправить её на удалённый сервер, где у вас есть права на запись. Ваши локальные ветки автоматически не синхронизируются с удалёнными при отправке — вам нужно явно указать те ветки, которые вы хотите отправить.

Таким образом, вы можете использовать свои личные ветки для работы, которую не хотите показывать, а отправлять только те тематические ветки, над которыми вы хотите работать с кем-то совместно. Отправка осуществляется командой `git push <remote> <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`.

Если у вас настроена ветка слежения, или она явно установлена, или она была создана автоматически командами `clone` или `checkout`, `git pull` определит сервер и ветку, за которыми следит ваша текущая ветка, получит данные с этого сервера и затем попытается слить удалённую ветку.

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

Для удаления локальной ветки выполните команду `git branch` с параметром `-d`.

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

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`?

В модели ветвления `git-flow`, основные типы веток это:

`master` - главная ветка проекта, которая содержит только стабильный код и используется для создания релизов.

`develop` - ветка, в которой ведется основная разработка проекта. Она содержит все изменения, которые были сделаны разработчиками.

`feature` - ветки для добавления новой функциональности в проект. Они ветвятся от ветки `develop`, и после завершения работы ветки объединяются с `develop` веткой.

`release` - ветки для подготовки новой версии проекта. Они ветвятся от ветки `develop`, содержат минимальный набор изменений и используются для подготовки релиза. После тестирования и отладки, ветка объединяется с веткой `master` и `develop`.

`hotfix` - ветки для исправления критических ошибок на производственной версии проекта. Они ветвятся от ветки `master`, и после исправления ошибок объединяются с `master` и `develop` ветками.

Работа с ветками в модели `git-flow` организована следующим образом:

Начало разработки новой функциональности начинается с ветвления от ветки `develop` ветки `feature`, на которой работает разработчик.

После завершения работы, все изменения ветки `feature` тестируются, затем вливаются обратно в ветку `develop`.

В момент подготовки новой версии программного продукта ветка `release` создается из `develop`, и она используется для проведения основных тестов наиболее важных функций.

Ветка `hotfix` создается из ветки `master`, если в производственной версии обнаружена критическая ошибка, и на этой ветке выполняется исправление.

Недостатки `git-flow` включают в себя:

Сложность и необходимость управления множеством веток, что может быть трудным для маленьких команд.

Не слишком хорошо подходит для быстрой разработки и быстрой реализации необходимых исправлений или функций в связи с наличием большого количества ветвлений.

Накладывает значительный набор процедур и правил для разработки и управления релизами.

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.

В Git клиенте `codeberg.org` присутствует возможность работы с ветками Git. Инструменты для работы с ветками в `codeberg.org` соответствуют стандартным инструментам Git и включают в себя:

Создание новой ветки из текущей ветки.

Переключение между ветками.

Объединение веток (с помощью команды `merge`), которые могут быть выполнены из интерфейса Git клиента.

Возможность создавать теги и выполнять релизы (включая создание новой ветки для релиза и склеивание веток для внесения исправлений в производственную версию приложения).

Вывод: в результате выполнения работы были исследованы базовые возможности по работе с локальными и удаленными ветками Git.