

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 ЛАБОРАТОРНОЙ РАБОТЕ №1.3**  
**дисциплины «Программирование на Python»**  
**Вариант \_\_\_\_**

Выполнил:  
Иващенко Олег Андреевич  
2 курс, группа ИВТ-б-о-22-1,  
09.03.02 «Информационные и  
вычислительные машины»,  
направленность (профиль)  
«Программное обеспечение средств  
вычислительной техники и  
автоматизированных систем»

---

(подпись)

Руководитель практики:  
Воронкин Роман Александрович,  
доцент кафедры инфокоммуникаций

---

(подпись)

Отчет защищен с оценкой \_\_\_\_\_ Дата защиты \_\_\_\_\_

Ставрополь, 2023 г.

**Тема:** «Основы ветвления Git»

**Цель:** Исследование базовых возможностей по работе с локальными и удалёнными ветками Git.

Порядок выполнения работы:

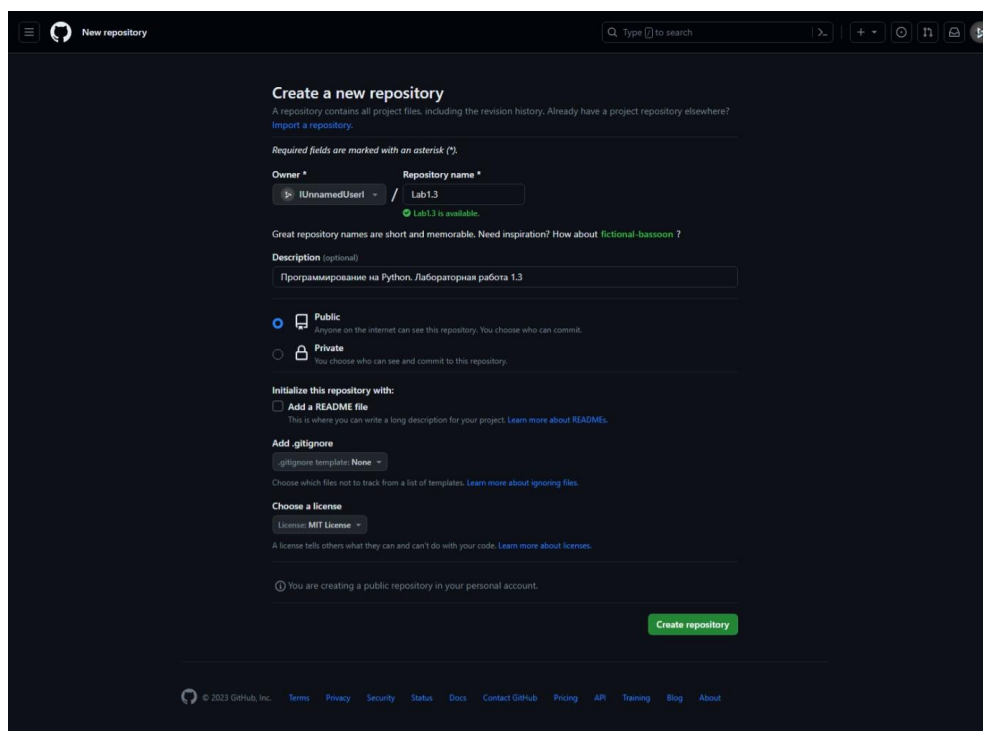


Рисунок 1 – Создание нового репозитория






 1.txt	28.09.2023 14:38	Текстовый докум...	0 КБ
 2.txt	28.09.2023 14:38	Текстовый докум...	0 КБ
 3.txt	28.09.2023 14:38	Текстовый докум...	0 КБ
 LICENSE	28.09.2023 14:35	Файл	2 КБ
 README.md	28.09.2023 14:36	Исходный файл ...	1 КБ

Рисунок 2 – Создание файлов

```
C:\Users\PackardBell\PythonLab\Lab1.3>git add 1.txt
C:\Users\PackardBell\PythonLab\Lab1.3>git commit -m "Add 1.txt file"
[my_first_branch 62583c9] Add 1.txt file
2 files changed, 0 insertions(+), 0 deletions(-)
delete mode 100644 2.txt
delete mode 100644 3.txt
C:\Users\PackardBell\PythonLab\Lab1.3>
```

Рисунок 3 – Добавление файла 1.txt и коммит

```

C:\Users\PackardBell\PythonLab\Lab1.3>git add 2.txt
C:\Users\PackardBell\PythonLab\Lab1.3>git add 3.txt
C:\Users\PackardBell\PythonLab\Lab1.3>git commit -m "Add 2.txt and 3.txt"
[my_first_branch 4dbe8f5] Add 2.txt and 3.txt
 2 files changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 2.txt
 create mode 100644 3.txt
C:\Users\PackardBell\PythonLab\Lab1.3>

```

Рисунок 4 – Добавление файлов 2.txt и 3.txt, перезапись коммита

```

C:\Users\PackardBell\PythonLab\Lab1.3>git branch my_first_branch
C:\Users\PackardBell\PythonLab\Lab1.3>git pull
Already up to date.
C:\Users\PackardBell\PythonLab\Lab1.3>git branch
* main
  my_first_branch
C:\Users\PackardBell\PythonLab\Lab1.3>

```

Рисунок 5 – Создание новой ветки с именем «my\_first\_branch»

```

C:\Users\PackardBell\PythonLab\Lab1.3>git branch
* main
  my_first_branch
C:\Users\PackardBell\PythonLab\Lab1.3>git checkout my_first_branch
Switched to branch 'my_first_branch'
C:\Users\PackardBell\PythonLab\Lab1.3>git branch
  main
* my_first_branch
C:\Users\PackardBell\PythonLab\Lab1.3>

```

Рисунок 6 – Смена текущей ветки

```

C:\Users\PackardBell\PythonLab\Lab1.3>git branch
  main
* my_first_branch
C:\Users\PackardBell\PythonLab\Lab1.3>git status
On branch my_first_branch
Untracked files:
  (use "git add <file>..." to include in what will be committed)
  in_branch.txt

nothing added to commit but untracked files present (use "git add" to track)
C:\Users\PackardBell\PythonLab\Lab1.3>git add in_branch.txt
C:\Users\PackardBell\PythonLab\Lab1.3>git commit -m "Add in_branch.txt file"
[my_first_branch 339f67f] Add in_branch.txt file
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 in_branch.txt

```

Рисунок 7 – Добавление и коммит нового файла в ветку

```

C:\Users\PackardBell\PythonLab\Lab1.3>git checkout -b new_branch
Switched to a new branch 'new_branch'
C:\Users\PackardBell\PythonLab\Lab1.3>git branch
  main
  my_first_branch
* new_branch

```

Рисунок 8 – Создание и переход в новую ветку

```

C:\Users\PackardBell\PythonLab\Lab1.3>git branch
main
my_first_branch
* new_branch

C:\Users\PackardBell\PythonLab\Lab1.3>git status
On branch new_branch
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   1.txt

no changes added to commit (use "git add" and/or "git commit -a")

C:\Users\PackardBell\PythonLab\Lab1.3>git add 1.txt

C:\Users\PackardBell\PythonLab\Lab1.3>git commit -m "Change 1.txt file"
[new_branch d696356] Change 1.txt file
1 file changed, 1 insertion(+)

C:\Users\PackardBell\PythonLab\Lab1.3>

```

Рисунок 9 – Изменение файла и коммит

```

C:\Users\PackardBell\PythonLab\Lab1.3>git merge my_first_branch
Updating 1bbe05b..339f67f
Fast-forward
 in_branch.txt | 0
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 in_branch.txt

```

Рисунок 10 – Слияние ветвей main и my\_first\_branch

```

C:\Users\PackardBell\PythonLab\Lab1.3>git merge new_branch
Merge made by the 'ort' strategy.
 1.txt | 1 +
 1 file changed, 1 insertion(+)

```

Рисунок 11 – Слияние ветвей main и new\_branch

```

C:\Users\PackardBell\PythonLab\Lab1.3>git branch
* main
  my_first_branch
  new_branch

C:\Users\PackardBell\PythonLab\Lab1.3>git branch -d my_first_branch
Deleted branch my_first_branch (was 339f67f).

C:\Users\PackardBell\PythonLab\Lab1.3>git branch -d new_branch
Deleted branch new_branch (was d696356).

C:\Users\PackardBell\PythonLab\Lab1.3>git branch
* main

```

Рисунок 12 – Удаление ветвей my\_first\_branch и new\_branch

```

C:\Users\PackardBell\PythonLab\Lab1.3>git branch
* main

C:\Users\PackardBell\PythonLab\Lab1.3>git branch branch_1

C:\Users\PackardBell\PythonLab\Lab1.3>git branch branch_2
fatal: not a valid object name: '_2'

C:\Users\PackardBell\PythonLab\Lab1.3>git branch branch_2

C:\Users\PackardBell\PythonLab\Lab1.3>git branch
  branch_1
  branch_2
* main

```

Рисунок 13 – Создание ветвей branch\_1 и branch\_2

```
C:\Users\PackardBell\PythonLab\Lab1.3>git checkout branch_1
Switched to branch 'branch_1'

C:\Users\PackardBell\PythonLab\Lab1.3>git branch
* branch_1
  branch_2
  main

C:\Users\PackardBell\PythonLab\Lab1.3>git status
On branch branch_1
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   1.txt
        modified:   3.txt

no changes added to commit (use "git add" and/or "git commit -a")

C:\Users\PackardBell\PythonLab\Lab1.3>git add .

C:\Users\PackardBell\PythonLab\Lab1.3>git commit -m "Fix 1.txt and 3.txt"
[branch_1 d7ae118] Fix 1.txt and 3.txt
2 files changed, 2 insertions(+), 1 deletion(-)
```

Рисунок 14 – Изменение файлов и сохранение изменений в ветке  
branch\_1

```
C:\Users\PackardBell\PythonLab\Lab1.3>git checkout branch_2
Switched to branch 'branch_2'

C:\Users\PackardBell\PythonLab\Lab1.3>git branch
  branch_1
* branch_2
  main

C:\Users\PackardBell\PythonLab\Lab1.3>git status
On branch branch_2
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   1.txt
        modified:   3.txt

no changes added to commit (use "git add" and/or "git commit -a")

C:\Users\PackardBell\PythonLab\Lab1.3>git add .

C:\Users\PackardBell\PythonLab\Lab1.3>git commit -m "My fix in the 1.txt and 3.txt"
[branch_2 ab7100d] My fix in the 1.txt and 3.txt
2 files changed, 2 insertions(+), 1 deletion(-)
```

Рисунок 15 – Изменение файлов и сохранение изменений в ветке  
branch\_2

```

C:\Users\PackardBell\PythonLab\Lab1.3>git checkout branch_1
Switched to branch 'branch_1'

C:\Users\PackardBell\PythonLab\Lab1.3>git branch
* branch_1
  branch_2
  main

C:\Users\PackardBell\PythonLab\Lab1.3>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.

C:\Users\PackardBell\PythonLab\Lab1.3>git status
On branch branch_1
You have unmerged paths.
  (fix conflicts and run "git commit")
  (use "git merge --abort" to abort the merge)

Unmerged paths:
  (use "git add <file>..." to mark resolution)
    both modified:   1.txt
    both modified:   3.txt

no changes added to commit (use "git add" and/or "git commit -a")

```

Рисунок 16 – Попытка слить ветки branch\_1 и branch\_2

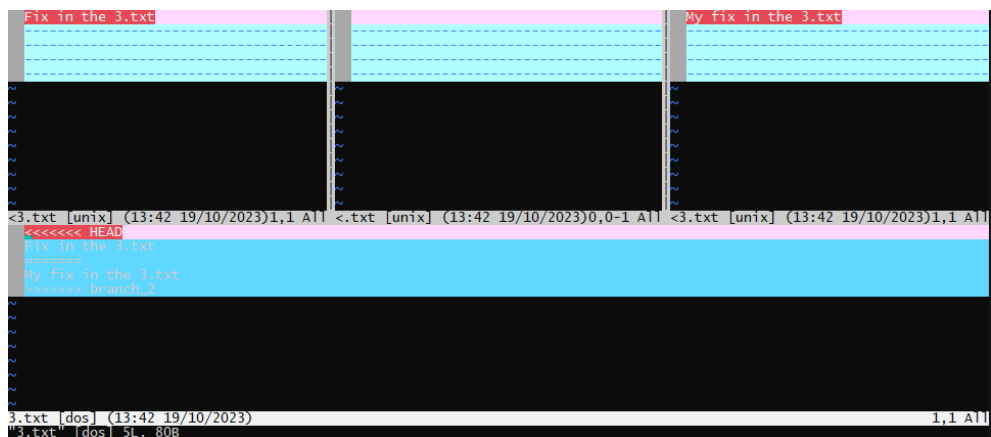


Рисунок 17 – Исправление конфликтов

```

Normal merge conflict for '3.txt':
  {local}: modified file
  {remote}: modified file
Hit return to start merge resolution tool (vimdiff): meld
4 files to edit

C:\Users\PackardBell\PythonLab\Lab1.3>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

```

Рисунок 18 – Результат решения конфликтов двумя способами

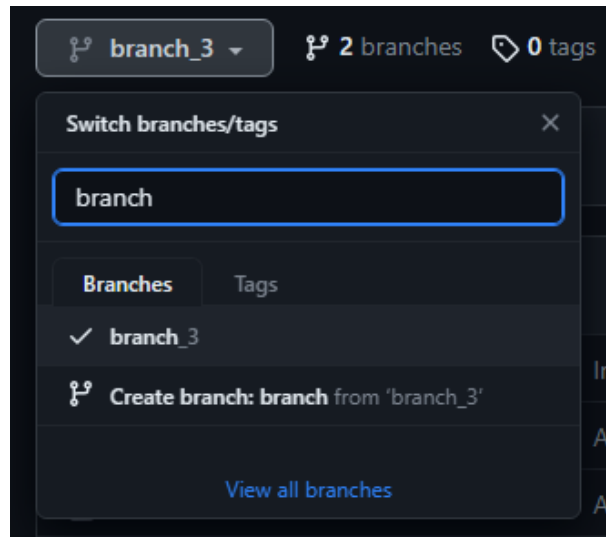


Рисунок 19 – Создание новой ветки средствами GitHub

```
C:\Users\PackardBell\PythonLab\Lab1.3>git checkout --track origin/branch_3
Switched to a new branch 'branch_3'
branch 'branch_3' set up to track 'origin/branch_3'.
```

Рисунок 20 – Создание ветки отслеживания удалённой ветки branch\_3

```
C:\Users\PackardBell\PythonLab\Lab1.3>git branch
  branch_1
  branch_2
* branch_3
  main

C:\Users\PackardBell\PythonLab\Lab1.3>git checkout branch_2
Switched to branch 'branch_2'

C:\Users\PackardBell\PythonLab\Lab1.3>git rebase main
Current branch branch_2 is up to date.

C:\Users\PackardBell\PythonLab\Lab1.3>_
```

Рисунок 21 – Перемещение ветки master на ветку branch\_2

```
C:\Users\PackardBell\PythonLab\Lab1.3>git push --set-upstream origin branch_2
Enumerating objects: 14, done.
Counting objects: 100% (13/13), done.
Delta compression using up to 4 threads
Compressing objects: 100% (8/8), done.
Writing objects: 100% (11/11), 1022 bytes | 511.00 KiB/s, done.
Total 11 (delta 4), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (4/4), done.
remote:
remote: Create a pull request for 'branch_2' on GitHub by visiting:
remote:   https://github.com/IUnnamedUserI/Lab1.3/pull/new/branch_2
remote:
To https://github.com/IUnnamedUserI/Lab1.3
 * [new branch]      branch_2 -> branch_2
branch 'branch_2' set up to track 'origin/branch_2'.
```

Рисунок 22 – Отправка изменений

## Контрольные вопросы

1. Что такое ветка?

Ветка в Git – это простой перемещаемый указатель на один из коммитов.  
По умолчанию имя основной ветки в Git – main.

2. Что такое HEAD?

HEAD – это указатель, задача которого ссылаться на определённый коммит в репозитории.

3. Способы создания веток.

Создать ветку можно с помощью команды `git branch <имя_ветки>`, а также можно с помощью графического интерфейса.

4. Как узнать текущую ветку?

Для вывода текущей активной ветки используется команда `git branch`.

5. Как переключаться между ветками?

Для переключения между ветками используется команда `git checkout`.

6. Что такое удалённая ветка?

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

7. Что такое ветка отслеживания?

Ветки слежения – это ссылки на определённое состояние удалённых веток. Это локальные ветки, которые нельзя перемещать.

Ветки слежения – это локальные ветки, которые напрямую связаны с удалённой веткой.

8. Как создать ветку отслеживания?

Создать ветку отслеживания можно выполнив команду `git checkout --track origin/<название_ветки>`

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

Отправка изменений из локальной ветки в удалённую производится командой `git push origin <название_ветки>`

10. В чём отличие команд `git fetch` и `git pull`?

Команда `git fetch` получает с сервера все изменения, которых ещё нет, но не будет изменять состояние рабочей директории. Эта команда просто получает данные и позволяет самостоятельно сделать слияние.



Команда `git pull` определяет сервер и ветку, за которыми следит текущая ветка, получает данные с этого сервера и затем попытается сдвинуть удалённую ветку.

11. Как удалить локальную и удалённую ветки?

Удаление веток осуществляется командами `git push origin --delete <имя_ветки>` или `git branch -d <имя_ветки>`

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

Модель `git-flow` представляет собой стратегию организации веток в репозитории Git, предназначенную для эффективной организации и управления разработкой программного обеспечения. Основными типами веток в модели `git-flow` являются:

- **Main** – в этой ветке находится стабильная версия продукта. Новый код интегрируется в эту ветку только после тщательного тестирования и убеждения в его стабильности.
- **Develop** – в этой ветке находится код, который находится в процессе разработки. Это рабочая ветка, где разработчики интегрируют свой код и решают конфликты.
- **Feature** – ветки этого типа используются для разработки новых функций или особенностей продукта. Они отходят от ветки **Develop** и интегрируются обратно после завершения работы над функцией.
- **Release** – ветки этого типа создаются перед выпуском новой версии продукта. В этой ветке происходит подготовка к релизу, включая исправления и подготовку документации.
- **Hotfix** – если в стабильной версии обнаруживается критическая ошибка, создаётся ветка **Hotfix** для её немедленного исправления.

Недостатки модели `git-flow` включают в себя:

- **Сложность**: `git-flow` может показаться слишком сложной для небольших команд или больших проектов. Она вводит

множество веток и дополнительных шагов, которые могут быть избыточными.

- Замедленный релиз – подготовка к релизу в модели git-flow может занять много времени, так как требуется создание отдельной ветки Release и процедура тестирования.
- Конфликты и слияния – в процессе интеграции веток могут возникать конфликты слияния, особенно при длительной разработке веток Feature.

**Выводы:** В процессе выполнения лабораторной работы исследовал базовые возможности по работе с локальными и удалёнными ветками Git.