

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ  
ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение  
высшего образования «СЕВЕРОКАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ  
УНИВЕРСИТЕТ»

Кафедра инфокоммуникаций

Институт цифрового развития

ОТЧЁТ

по лабораторной работе №1.3

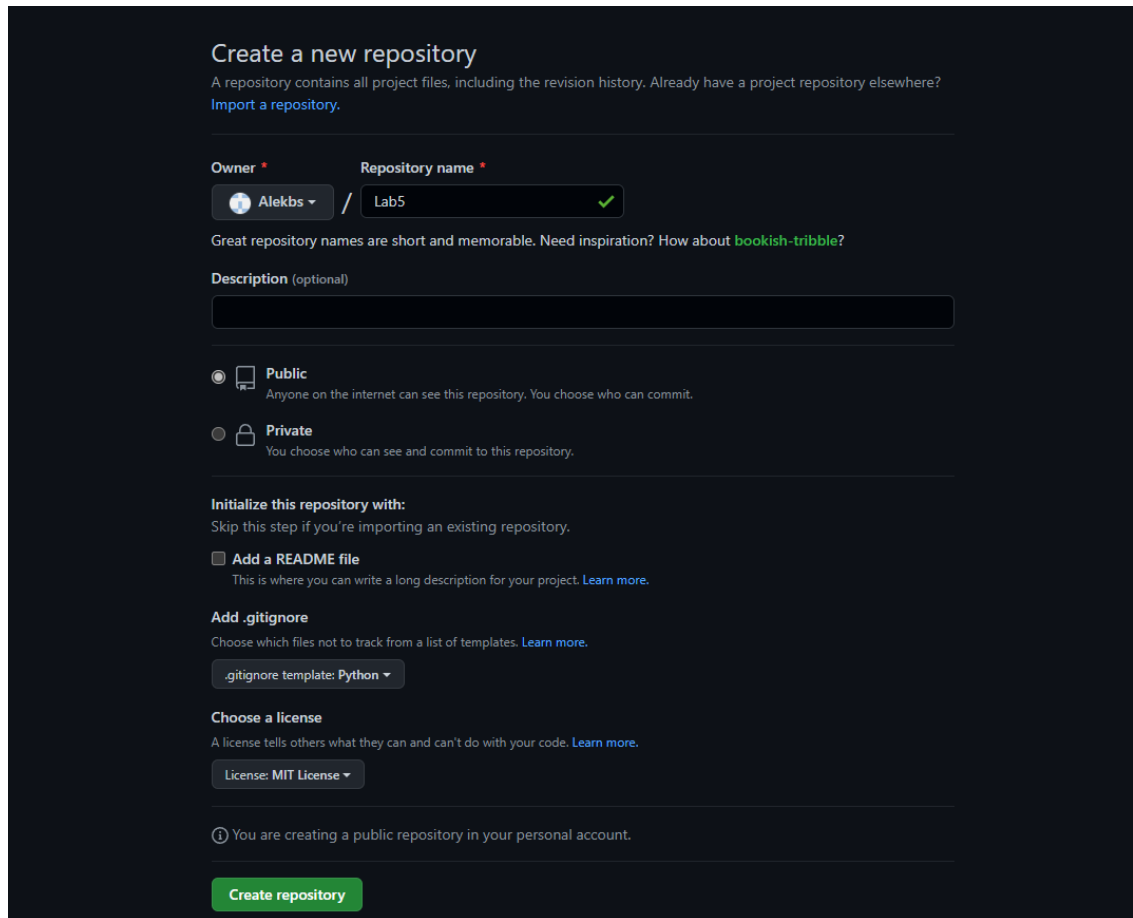
Дисциплина: «Основы кроссплатформенного программирования»

Тема: «Основы ветвления Git»

Выполнил: студент 1 курса,  
группы ИВТ-б-о-21-1  
Богдан Александр Анатольевич

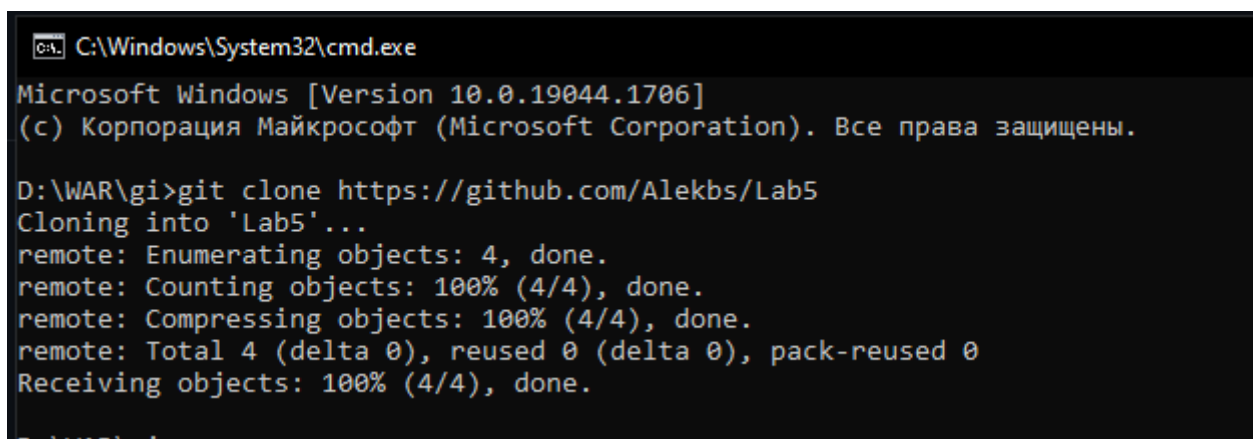
Ставрополь 2022

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



The screenshot shows the GitHub 'Create a new repository' interface. At the top, it says 'Create a new repository' and 'A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)'. Below this, there are two input fields: 'Owner' with a dropdown menu showing 'Alekbs' and 'Repository name' with a text input 'Lab5' and a green checkmark. A note says 'Great repository names are short and memorable. Need inspiration? How about [bookish-tribble?](#)'. There is a 'Description (optional)' text area. Below that, there are two radio buttons for visibility: 'Public' (selected) and 'Private'. The 'Public' option has a subtext: 'Anyone on the internet can see this repository. You choose who can commit.' The 'Private' option has a subtext: 'You choose who can see and commit to this repository.' Below the visibility options, there is a section 'Initialize this repository with:' with a subtext: 'Skip this step if you're importing an existing repository.' There are three checkboxes: 'Add a README file' (checked), 'Add .gitignore' (checked), and 'Choose a license' (checked). The 'Add a README file' checkbox has a subtext: 'This is where you can write a long description for your project. [Learn more.](#)'. The 'Add .gitignore' checkbox has a subtext: 'Choose which files not to track from a list of templates. [Learn more.](#)'. The 'Choose a license' checkbox has a subtext: 'A license tells others what they can and can't do with your code. [Learn more.](#)'. Below the checkboxes, there is a dropdown menu for '.gitignore template: Python'. Below the dropdown menu, there is a section 'Choose a license' with a subtext: 'A license tells others what they can and can't do with your code. [Learn more.](#)'. There is a dropdown menu for 'License: MIT License'. At the bottom, there is a green button 'Create repository' and a note: 'You are creating a public repository in your personal account.'

Рисунок 1. Создание репозитория



```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.19044.1706]
(c) Корпорация Майкрософт (Microsoft Corporation). Все права защищены.

D:\WAR\gi>git clone https://github.com/Alekbs/Lab5
Cloning into 'Lab5'...
remote: Enumerating objects: 4, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 4 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (4/4), done.
```

Рисунок 2. Клонирование репозитория






 .gitignore	25.05.2022 0:56	Текстовый докум...	2 КБ
 1.txt	25.05.2022 0:56	Текстовый докум...	0 КБ
 2.txt	25.05.2022 0:56	Текстовый докум...	0 КБ
 3.txt	25.05.2022 0:57	Текстовый докум...	0 КБ
 LICENSE	25.05.2022 0:56	Файл	2 КБ

Рисунок 3. Создание текстовых файлов

```
D:\WAR\gi\Lab5>git add 1.txt
D:\WAR\gi\Lab5>git commit -m "add 1.txt file"
[main ede1015] add 1.txt file
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 1.txt
D:\WAR\gi\Lab5>git add .
D:\WAR\gi\Lab5>git commit --amend -m "add 2.txt and 3.txt file"
[main 3d16218] add 2.txt and 3.txt file
Date: Wed May 25 00:59:18 2022 +0300
3 files changed, 0 insertions(+), 0 deletions(-)
create mode 100644 1.txt
create mode 100644 2.txt
create mode 100644 3.txt
```

Рисунок 4. Сохранение текстовых файлов

```
D:\WAR\gi\Lab5>git branch my_first_branch
D:\WAR\gi\Lab5>git branch
* main
  my_first_branch
D:\WAR\gi\Lab5>git checkout my_first_branch
Switched to branch 'my_first_branch'
```

Рисунок 5. Создание ветки my\_first\_branch

```

D:\WAR\gi\Lab5>git add .

D:\WAR\gi\Lab5>git commit -m "branch"
[my_first_branch 1cec252] branch
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 in_branch.txt

D:\WAR\gi\Lab5>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 6. Изменение файлов в ветке my\_first\_branch

```

D:\WAR\gi\Lab5>git branch new_branch

D:\WAR\gi\Lab5>git checkout new_branch
Switched to branch 'new_branch'

```

Рисунок 7. Создание ветки new\_branch

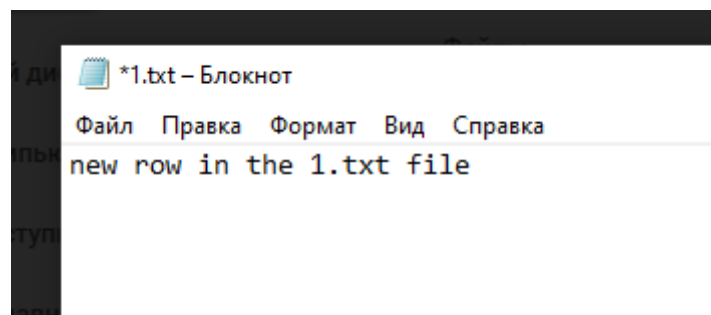


Рисунок 8. Изменение файла 1.txt

```

D:\WAR\gi\Lab5>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)

D:\WAR\gi\Lab5>git merge my_first_branch
Updating 3d16218..1cec252
Fast-forward
 in_branch.txt | 0
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 in_branch.txt

D:\WAR\gi\Lab5>git merge new_branch
Merge made by the 'ort' strategy.
1.txt | 1 +
1 file changed, 1 insertion(+)

D:\WAR\gi\Lab5>

```

Рисунок 9. Объединение веток main с my\_first\_branch и new\_branch

```
D:\WAR\gi\Lab5>git branch -d my_first_branch
Deleted branch my_first_branch (was 1cec252).

D:\WAR\gi\Lab5>git branch -d new_branch
Deleted branch new_branch (was 16ef4be).
```

Рисунок 10. Удаление веток

```
D:\WAR\gi\Lab5>git branch branch_1
D:\WAR\gi\Lab5>git branch branch_2
```

Рисунок 11. Создание новых веток

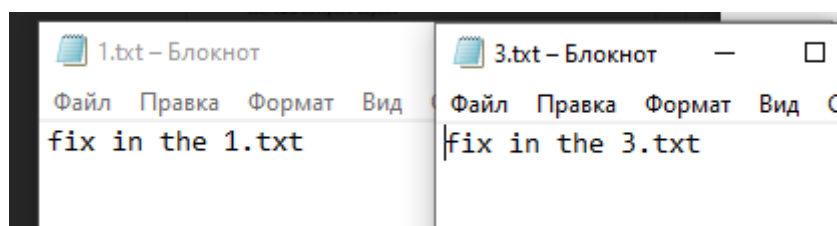


Рисунок 12. Изменение файлов 1 и 3

```
D:\WAR\gi\Lab5>git commit -m "1 and 3"
[branch_1 415e9a7] 1 and 3
2 files changed, 2 insertions(+), 1 deletion(-)
```

Рисунок 13. Сохранение изменений на ветке branch\_1

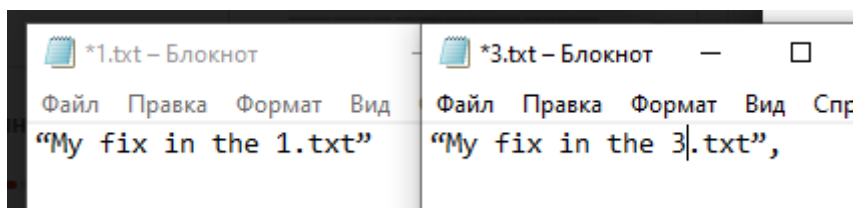


Рисунок 14. Изменение файлов 1 и 3

```
D:\WAR\gi\Lab5>git add .

D:\WAR\gi\Lab5>git commit -m "1 and 3 again"
[branch_2 1d8bb0e] 1 and 3 again
2 files changed, 2 insertions(+), 1 deletion(-)
```

Рисунок 15. Сохранение изменений на ветке branch\_2

```
D:\WAR\gi\Lab5>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. Объединение веток и появление конфликта

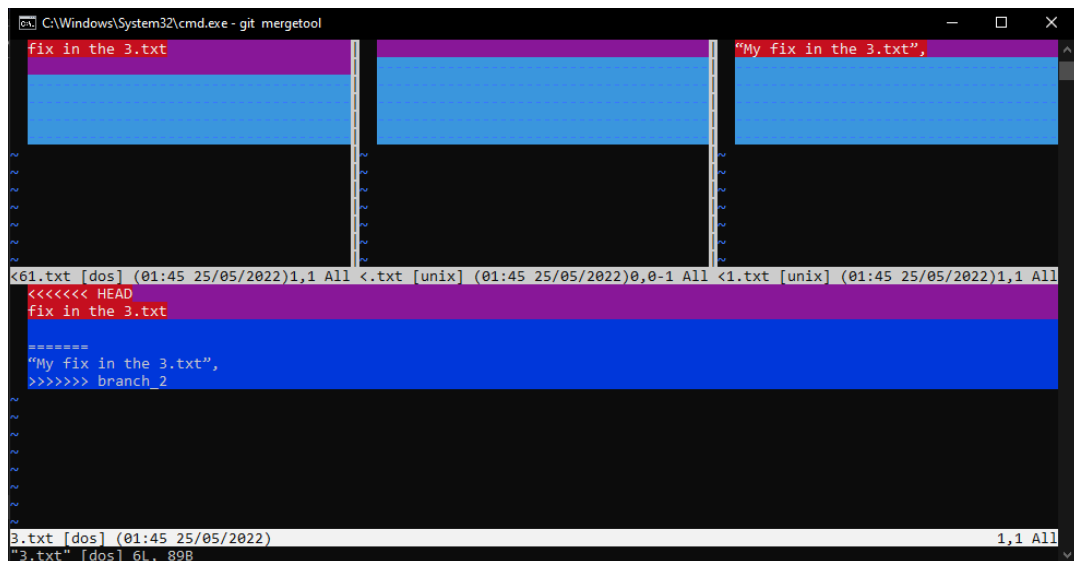


Рисунок 17. Разрешение конфликта с помощью mergetool

```
D:\WAR\gi\Lab5>git push origin branch_1
Enumerating objects: 30, done.
Counting objects: 100% (30/30), done.
Delta compression using up to 8 threads
Compressing objects: 100% (19/19), done.
Writing objects: 100% (29/29), 2.19 KiB | 559.00 KiB/s, done.
Total 29 (delta 10), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (10/10), done.
remote:
remote: Create a pull request for 'branch_1' on GitHub by visiting:
remote:   https://github.com/Alekbs/Lab5/pull/new/branch_1
remote:
To https://github.com/Alekbs/Lab5
 * [new branch]      branch_1 -> branch_1
```

Рисунок 18. Отправка ветки branch\_1

```
D:\WAR\gi\Lab5>git checkout branch_3
Switched to a new branch 'branch_3'
branch 'branch_3' set up to track 'origin/branch_3'.

D:\WAR\gi\Lab5>
```

Рисунок 19. Создание ветки отслеживания

```
D:\WAR\gi\Lab5>git add .
D:\WAR\gi\Lab5>git commit -m "final"
[branch_3 eed0d38] final
1 file changed, 1 insertion(+)
create mode 100644 2.txt
D:\WAR\gi\Lab5>
```

Рисунок 20. Изменение ветки branch\_3

```
D:\WAR\gi\Lab5>git rebase main
Current branch branch_2 is up to date.
```

Рисунок 21. Перемещение ветки main на branch\_2

```
D:\WAR\gi\Lab5>git push --set-upstream origin branch_2
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: Create a pull request for 'branch_2' on GitHub by visiting
remote:   https://github.com/Alekbs/Lab5/pull/new/branch_2
remote:
To https://github.com/Alekbs/Lab5
 * [new branch]      branch_2 -> branch_2
branch 'branch_2' set up to track 'origin/branch_2'.
```

Рисунок 22. Отправка изменений

### Ответы на вопросы:

1. Что такое ветка?

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

3. Способы создания веток.

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

4. Как узнать текущую ветку?

Увидеть текущую ветку можно при помощи простой команды `git log`, которая покажет вам куда указывают указатели веток. Эта опция называется `-decorate`.

## 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 вы можете настроить отслеживание и других веток — следить за ветками на других серверах или отключить слежение за веткой `master`. Вы только что видели простейший пример, что 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. Если у вас есть ветка `serverfix`, над которой вы хотите работать с кем-то ещё, вы можете отправить её точно так же, как вы отправляли вашу первую ветку. Выполните команду `git push <remote> <branch>`.

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

Для синхронизации ваших изменений с удаленным сервером выполните команду `git fetch <remote>` (в нашем случае `git fetch origin`). Эта команда определяет какому серверу соответствует “origin” (в нашем случае это `git.ourcompany.com`), извлекает оттуда данные, которых у вас ещё нет, и обновляет локальную базу данных, сдвигая указатель `origin/master` на новую позицию.

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

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`?

`Git-flow` — это устаревшая версия рабочего процесса `Git`, в свое время ставшая принципиально новой стратегией управления ветками в `Git`. Это альтернативная модель ветвления `Git`, в которой используются функциональные ветки и несколько основных веток.

Основными типами веток являются функциональные ветки, ветки выпуска, ветки исправления.

Последовательность действий при работе по модели Gitflow:

- 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.

Первая проблема: авторам приходится использовать ветку develop вместо master, поскольку master зарезервирован для кода, который отправляется в продакшен. Существует сложившийся обычай называть рабочую ветвь по умолчанию master, и делать ответвления и слияния с ней. Большинство инструментов по умолчанию используют это название для основной ветки и по умолчанию выводят именно ее, и бывает неудобно постоянно переключаться вручную на другую ветку.

Вторая проблема процесса git flow – сложности, возникающие из-за веток для патчей и для релиза.

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