

**МИНОБРНАУКИ РОССИИ**

**федеральное государственное автономное образовательное учреждение**

**высшего образования**

**«Московский государственный технологический университет «СТАНКИН»**

**(ФГАОУ ВО «МГТУ «СТАНКИН»)**

|  |  |  |
| --- | --- | --- |
| |  |  | | --- | --- | | Институт цифровых интеллектуальных систем | Кафедра  компьютерных систем управления | |

Дисциплина «Основы системного программного обеспечения»

**Отчет по лабораторной работе №1**

|  |  |  |
| --- | --- | --- |
| **Выполнил**  **студент гр. АДБ-22-06:** | \_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_  *(дата) (подпись)* | **Комаров А.О.** |
| **Проверил**  **к.т.н., доцент** | \_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_  *(дата) (подпись)* | **Ковалев И.А.** |

**Москва 2025 г.**

Оглавление

[1. Создание нового репозитория 3](#_Toc198646514)

[2. Установка git 4](#_Toc198646515)

[3. Создание локального репозитория 4](#_Toc198646516)

[4. Фиксация изменений в области заготовленных файлов 5](#_Toc198646517)

[5. Отправка коминат на сервер 6](#_Toc198646518)

[6. Запросим изменения с сервера 6](#_Toc198646519)

[7. Перешлем локальный коммит на сервер 6](#_Toc198646520)

[8. Создание новой ветки 8](#_Toc198646521)

[9. Слияние веток 10](#_Toc198646522)

[10. Просмотр изменений и разрешение конфликтов 13](#_Toc198646523)

[11. Удаление веток на сервере 14](#_Toc198646524)

[12. Возврат к предыдущему состоянию 15](#_Toc198646525)

[13. Исправление коммита 15](#_Toc198646526)

[14. Отправка только нужных файлов на сервер 17](#_Toc198646527)

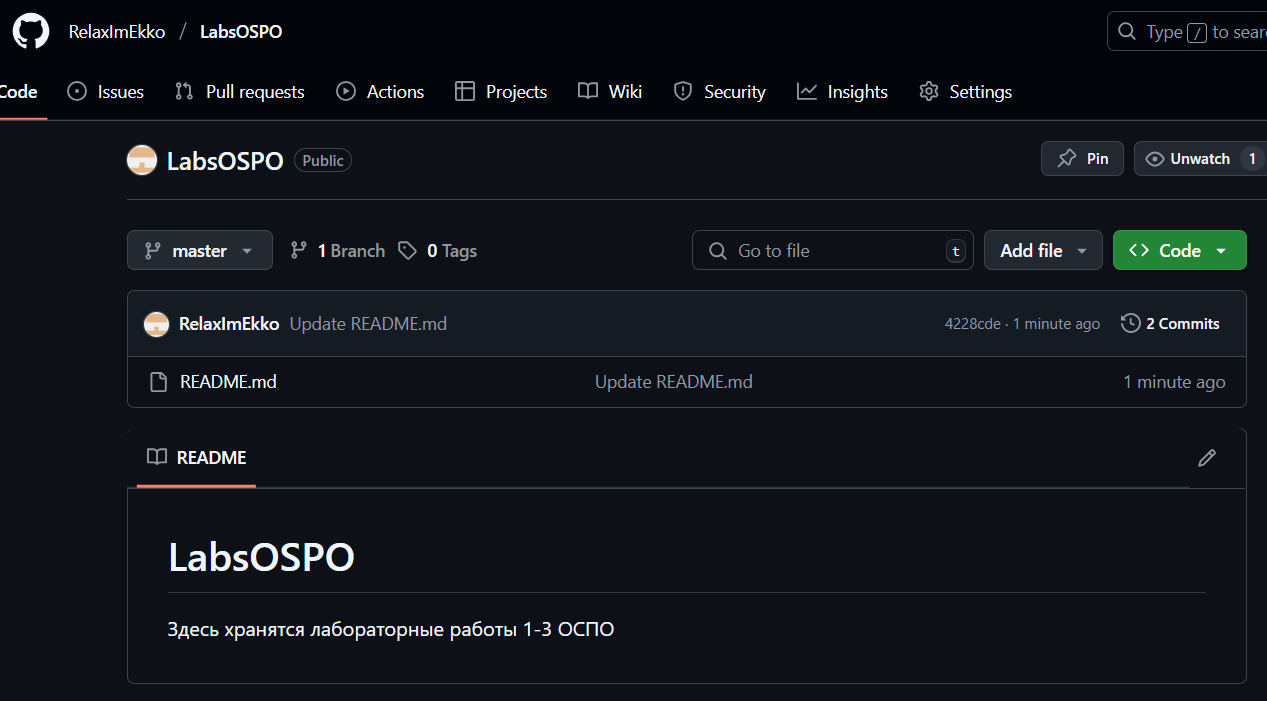
[15. Совместная работа с git 17](#_Toc198646528)

**Работа с системами контроля версий на примере Git Hub**

**Цель работы**: изучение и освоение основ работы с системами контроля версий на примере GitHub. Задача лабораторной работы — научиться создавать и управлять репозиториями, осуществлять фиксацию изменений, работать с ветками, синхронизировать локальные и удалённые репозитории, а также решать конфликты при совместной разработке с использованием системы Git.

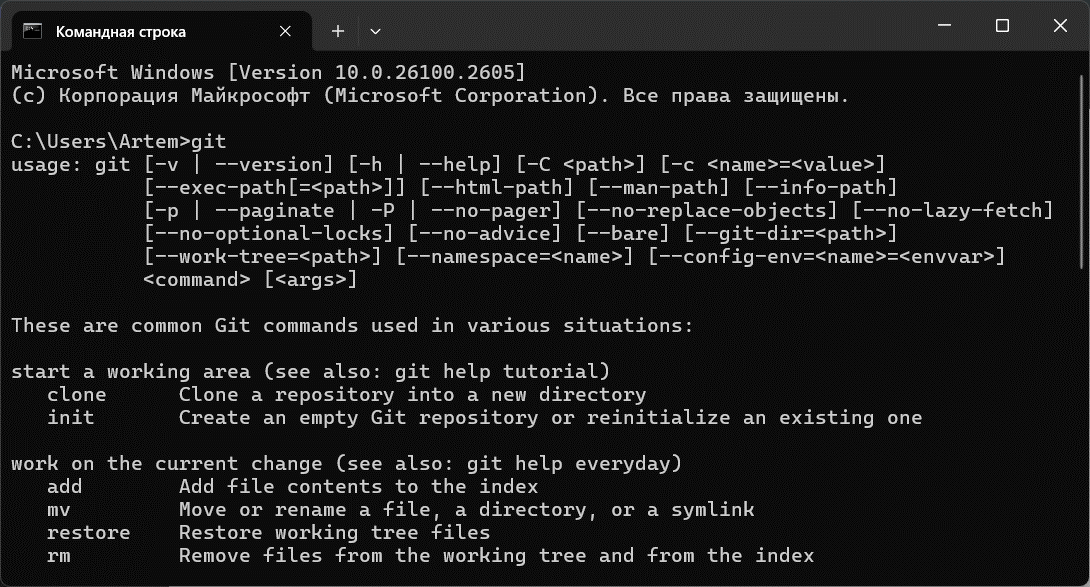
# 1. Создание нового репозитория

Этот пункт включает создание нового репозитория на GitHub. Необходимо зарегистрироваться на GitHub, создать новый репозиторий, добавить описание и выбрать тип репозитория. Рекомендуется создать файл README, который будет автоматически загружаться при открытии репозитория.



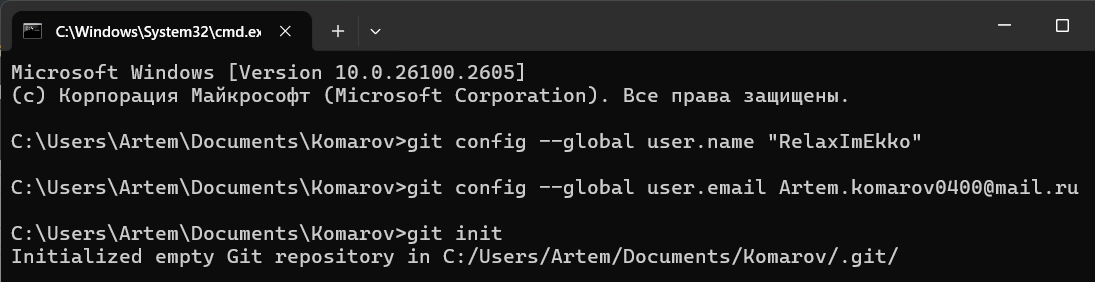
# 2. Установка git

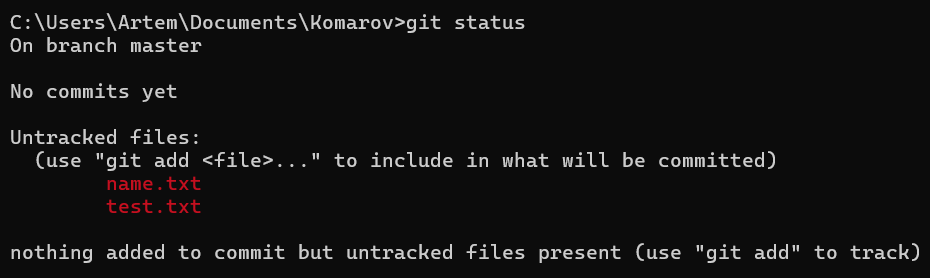
На этом этапе происходит установка Git на локальный компьютер. Нужно скачать установочный файл с официального сайта Git и проверить успешность установки, введя команду git в командной строке.



# 3. Создание локального репозитория

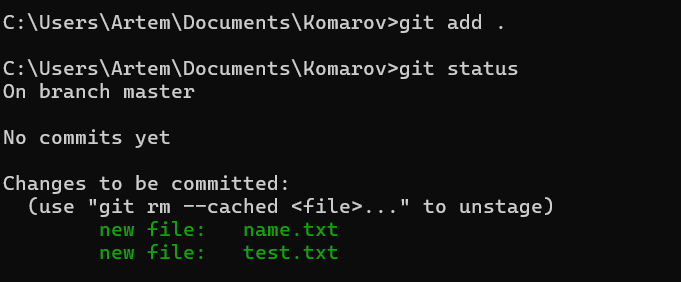
В данном шаге создается локальный репозиторий на компьютере. Для этого создается папка, в которой и будет храниться проект, после чего выполняется команда git init для инициализации репозитория.

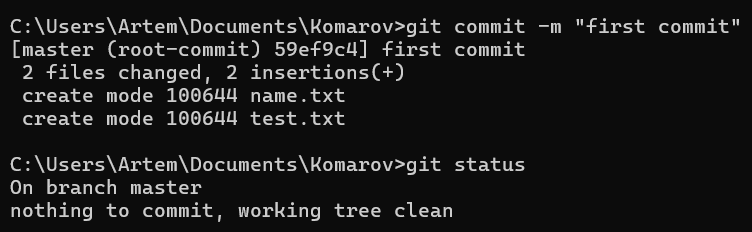
****

****

# 4. Фиксация изменений в области заготовленных файлов

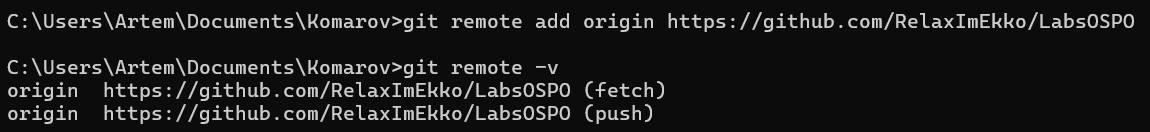
В этом пункте работы осуществляется подготовка файлов для коммита. Необходимо добавить измененные файлы в область подготовленных файлов с помощью команды git add, а затем зафиксировать изменения с помощью команды git commit.

****

****

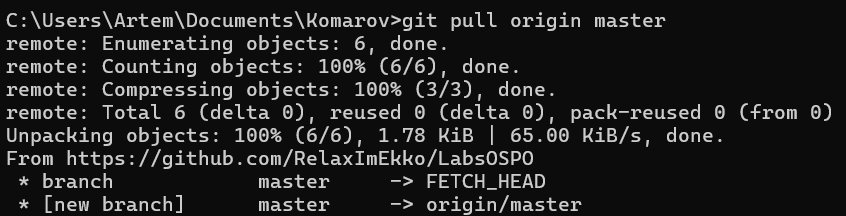
# 5. Отправка коминат на сервер

После фиксации изменений локально, необходимо отправить коммиты в удалённый репозиторий на GitHub с помощью команды git push.

****

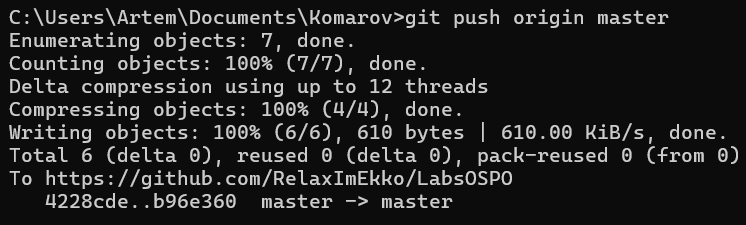
# 6. Запросим изменения с сервера

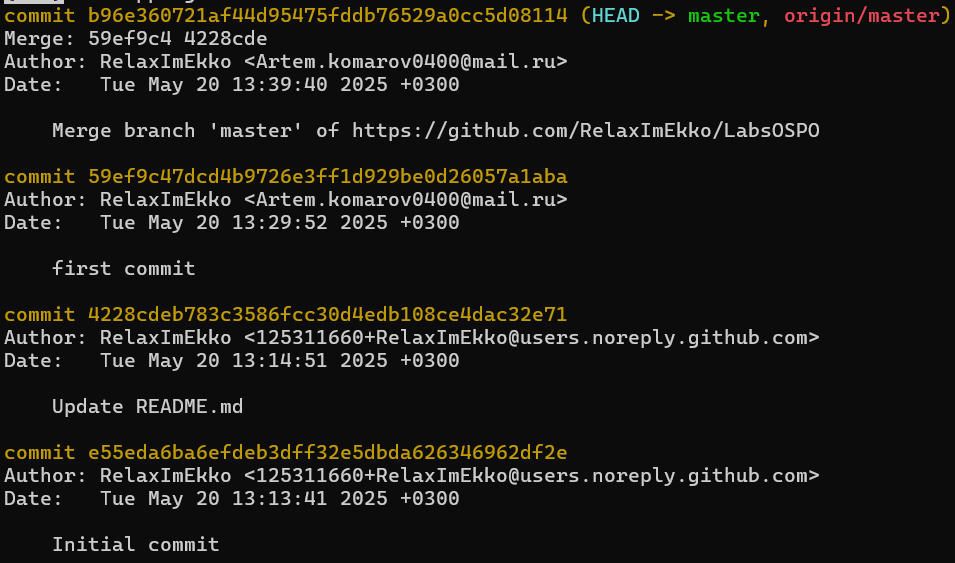
Для того чтобы получить последние изменения из удаленного репозитория, используется команда git pull, которая загружает обновления с сервера в локальный репозиторий.

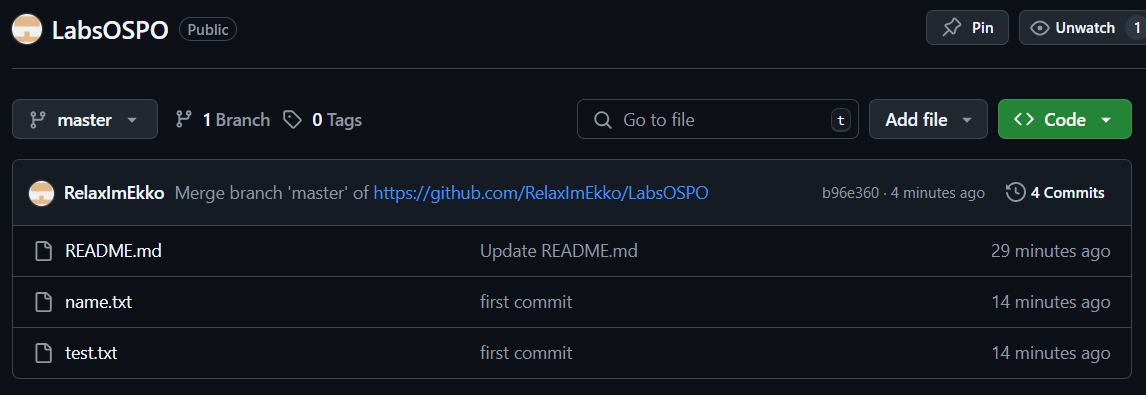
****

# 7. Перешлем локальный коммит на сервер

На этом шаге происходит отправка локальных изменений на сервер, используя команду git push. Это позволяет синхронизировать локальный репозиторий с удалённым.

****

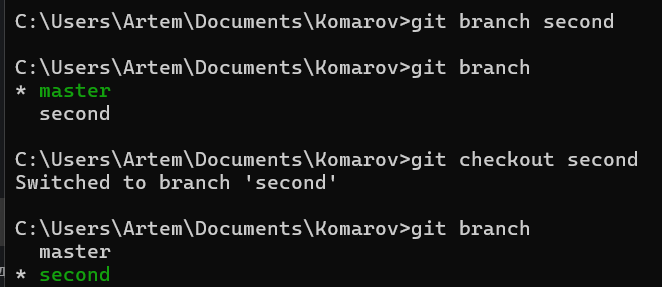
****

****

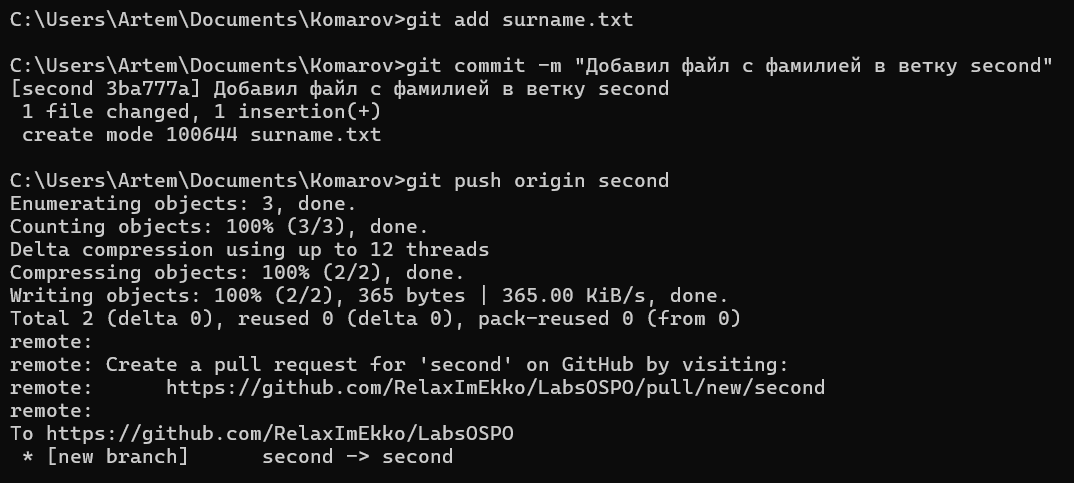
****

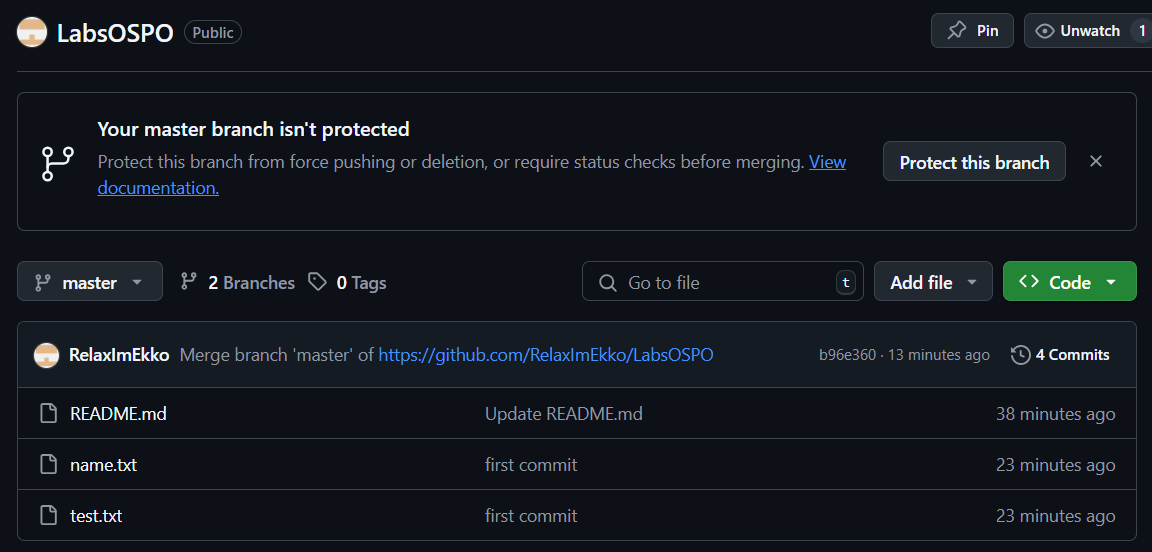
# 8. Создание новой ветки

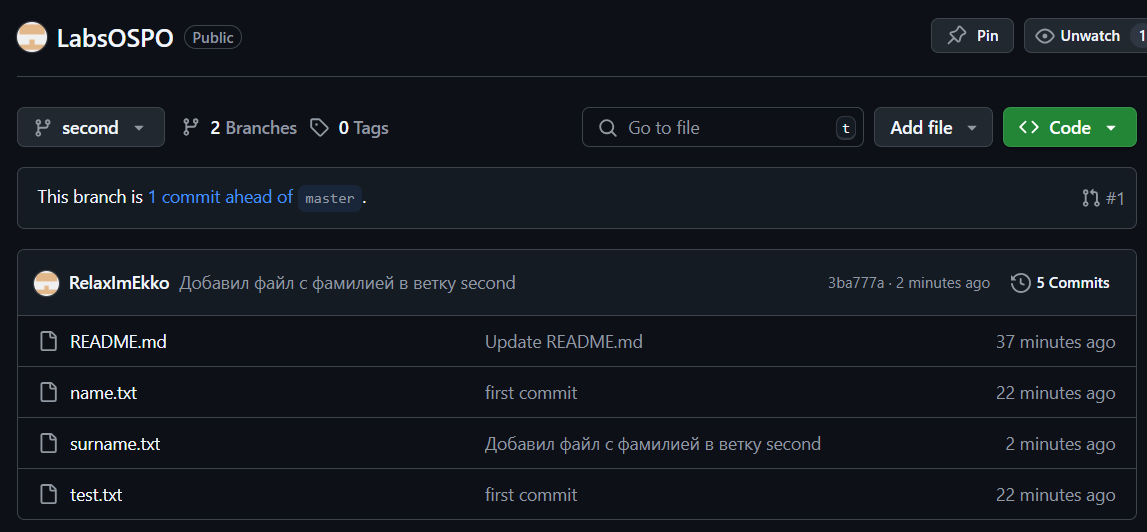
Создается новая ветка для разработки, чтобы не нарушать основную (master) ветку. Это позволяет изолировать изменения и работать над новой функциональностью без риска повредить основной код. Для создания ветки используется команда git branch, а для переключения на неё — git checkout.

****

Создадим новый файл в нашем локальном репозитории и напишем в нем свою фамилию, добавим в область подготовленных файлов, закомитим и отправим на сервер.

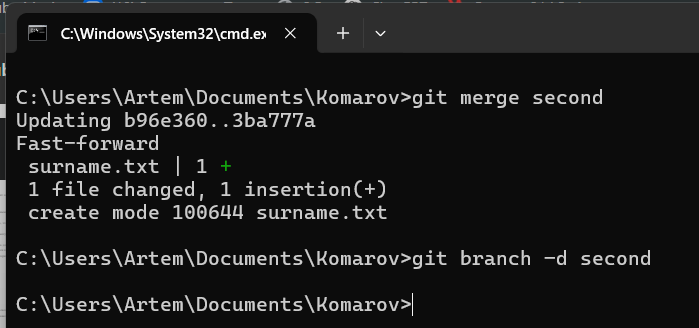
****

****

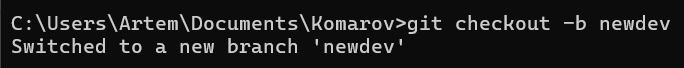
****

# 9. Слияние веток

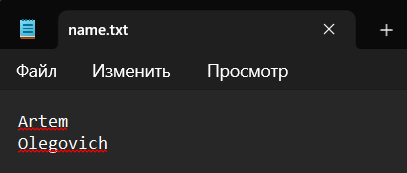
После того как изменения в новой ветке завершены, они сливаются с основной веткой (master) с помощью команды git merge. Это позволяет интегрировать изменения из одной ветки в другую.

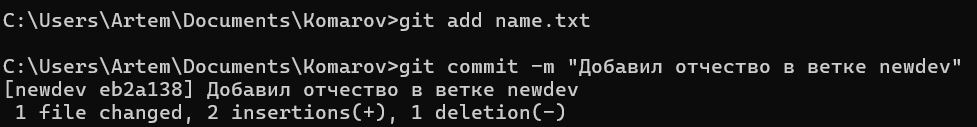
****

Создадим ветку с названием newdev и переключимся на нее

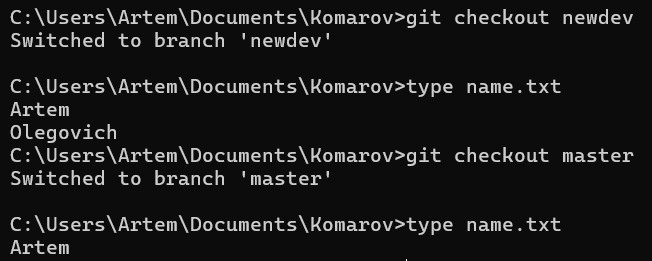
****

Добавим в файл с именем отчество

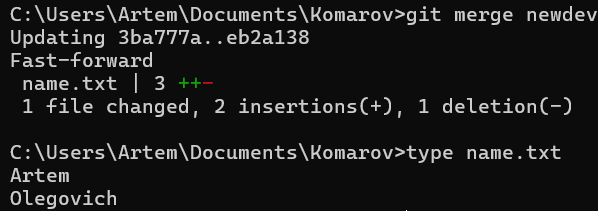


Фиксируем изменения и делаем коммит

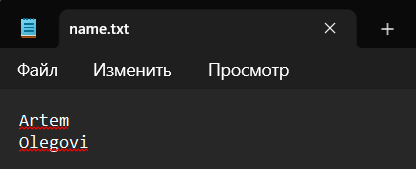
Переключаемся обратно на master и проверяем файл

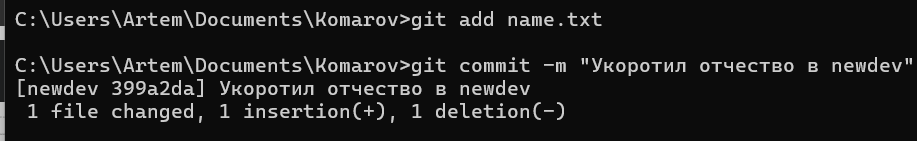


Сливаем ветку newdev в master

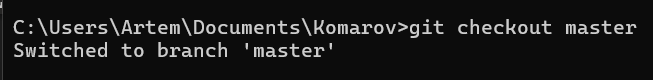


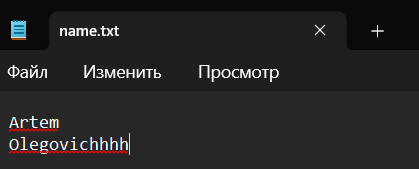
Переключаемся на newdev и удаляем часть отчества

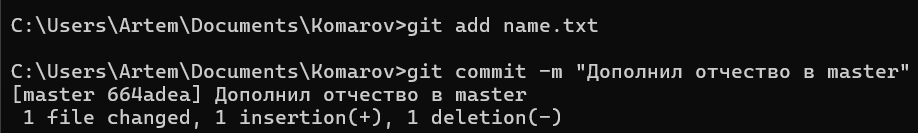




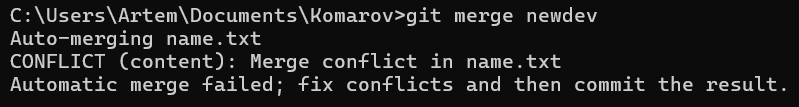
Переключаемся на master и дописываем отчество





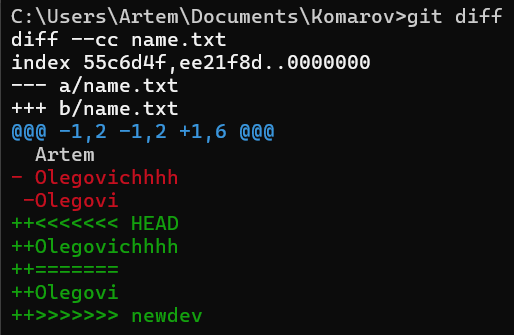


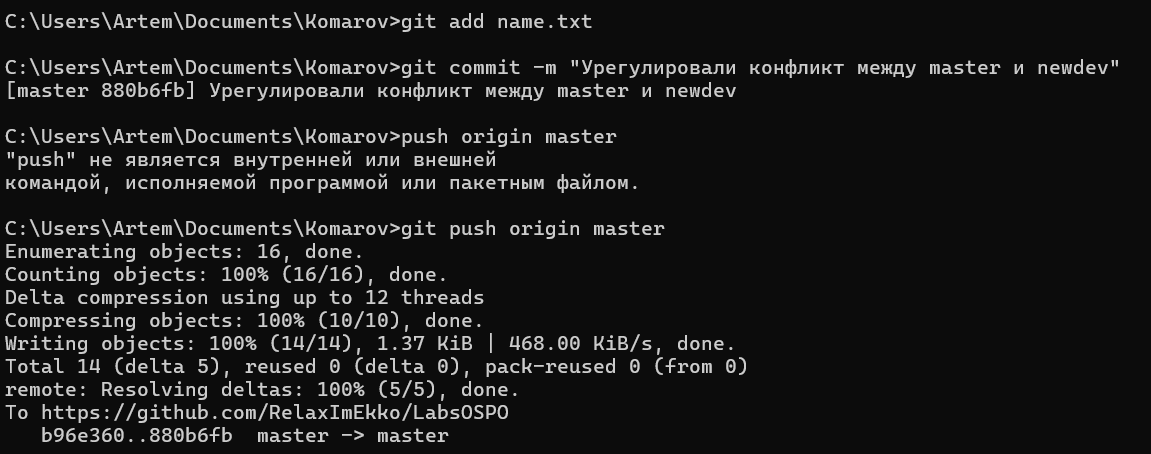
Пытаемся слить newdev в master

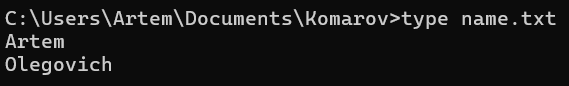


# 10. Просмотр изменений и разрешение конфликтов

В случае конфликта при слиянии веток необходимо разрешить его вручную. Команда git diff позволяет просмотреть различия между файлами, а также выявить конфликтующие изменения. Разработчик вручную выбирает, какие изменения оставить, после чего коммитит результат.

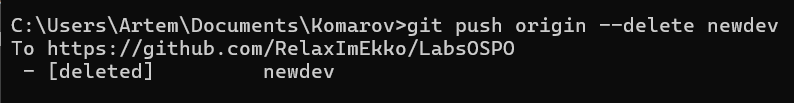
****

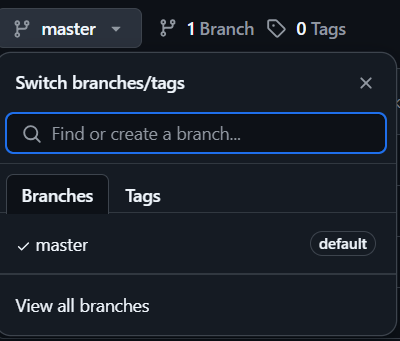
****

****

# 11. Удаление веток на сервере

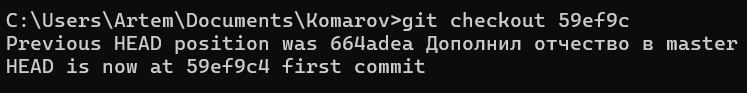
После завершения работы с веткой и слияния её изменений с основной веткой, ветка может быть удалена с удалённого репозитория с помощью команды git push origin --delete имя\_ветки.

****

****

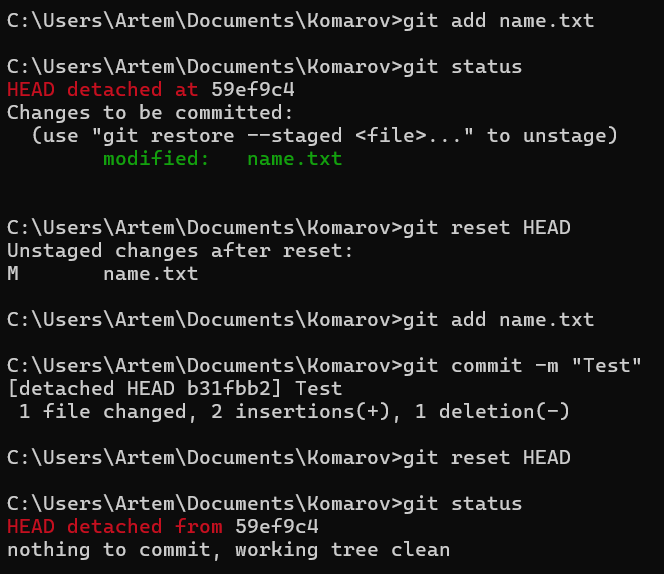
# 12. Возврат к предыдущему состоянию

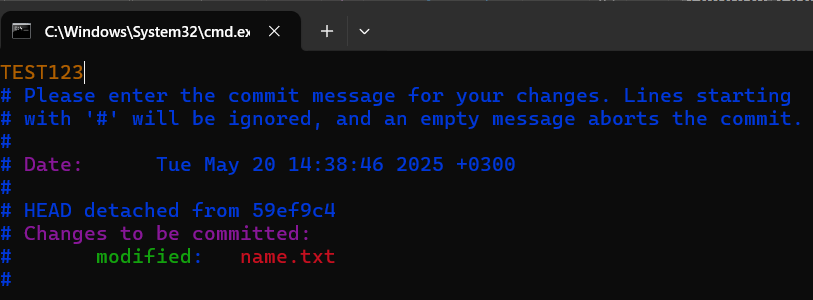
Если нужно вернуться к предыдущему состоянию проекта, например, отменить последний коммит, используется команда git checkout [hash commit], которая позволяет откатиться на выбранный коммит.

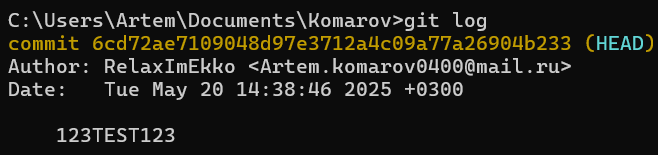
****

# 13. Исправление коммита

Если после коммита нужно внести изменения, это можно сделать с помощью команды git commit --amend, которая позволяет изменить последний коммит, например, добавив забытые файлы или исправив сообщение.

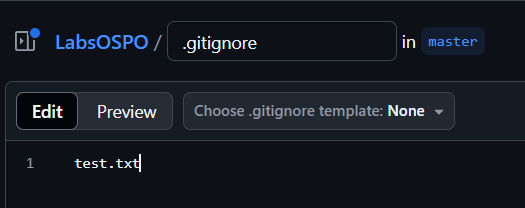
****

****

****

# 14. Отправка только нужных файлов на сервер

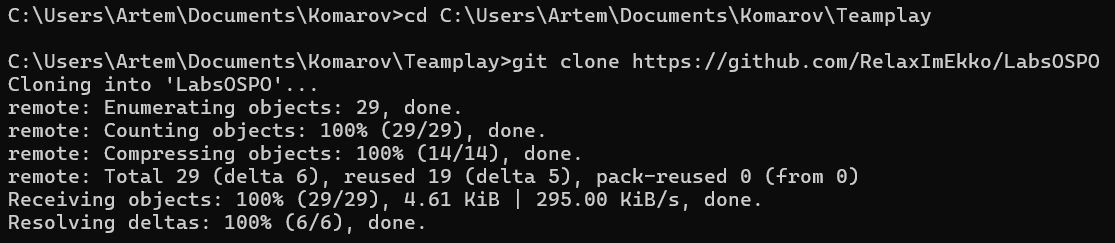
Для отправки только нужных файлов на сервер используется файл .gitignore, в который заносятся файлы и директории, которые не должны попадать в репозиторий.

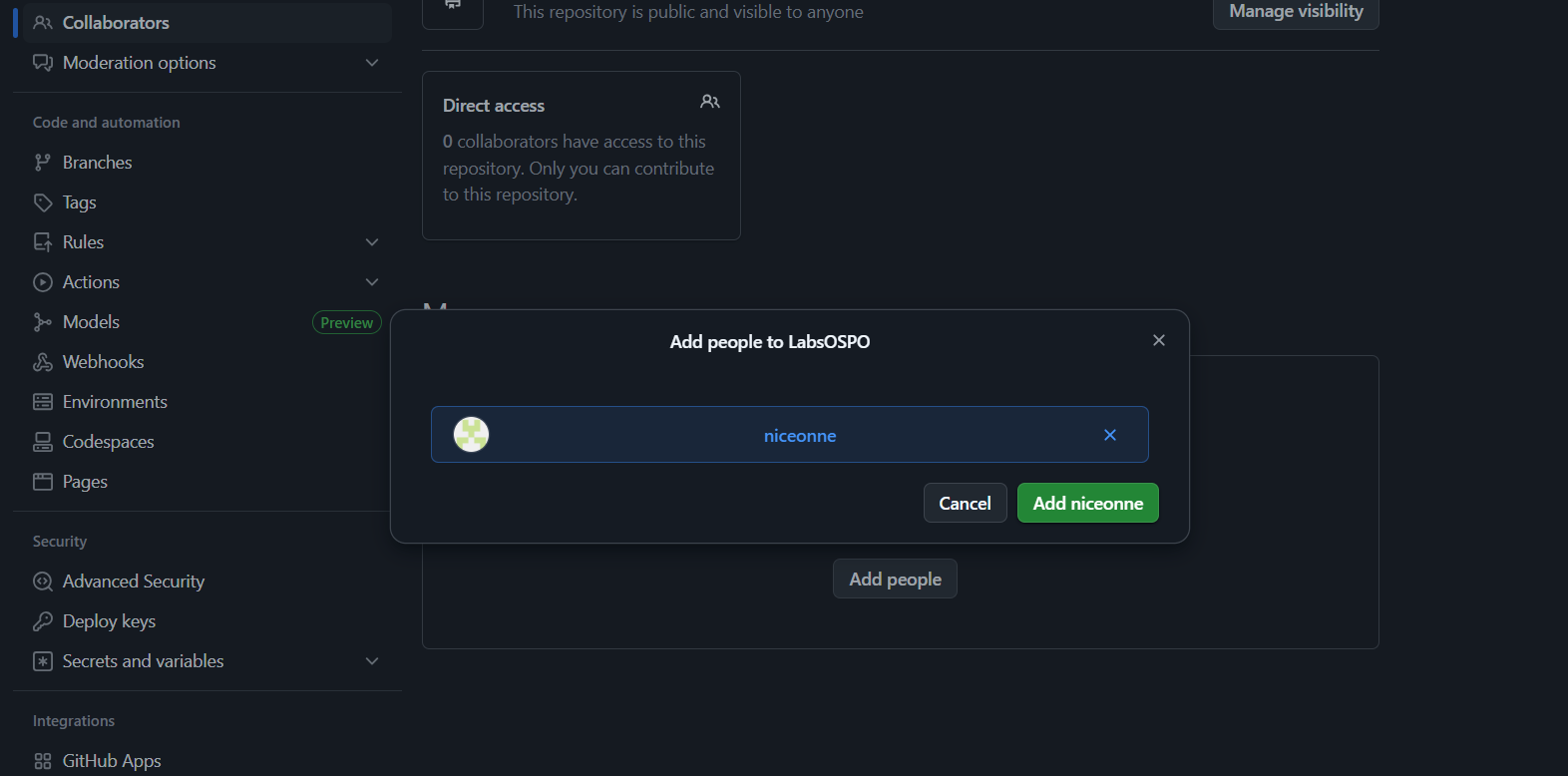
****

****

# 15. Совместная работа с git

На этом этапе происходит совместная работа с другими разработчиками. Каждый участник работы форкает репозиторий, вносит изменения и отправляет их на сервер, после чего другие участники могут синхронизировать свои репозитории, используя команду git pull.

****

****