# МИНИCTEPCTBO 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 программной инженерии»

Тема: «Основы ветвления GIT»

Выполнила: студентка 2 курса группы Пиж-б-о-21-1

Джолдошова Мээрим Бекболотовна

Ставрополь 2022

**Цель:** исследование базовых возможностей по работе с локальными и удаленными ветками GIT

**Выполнение работы**

1. Был создан репозиторий с использованием лицензии MIT и файлами gitignor для выбранного языка программирования C++. Далее он был клонирован по ноутбук.

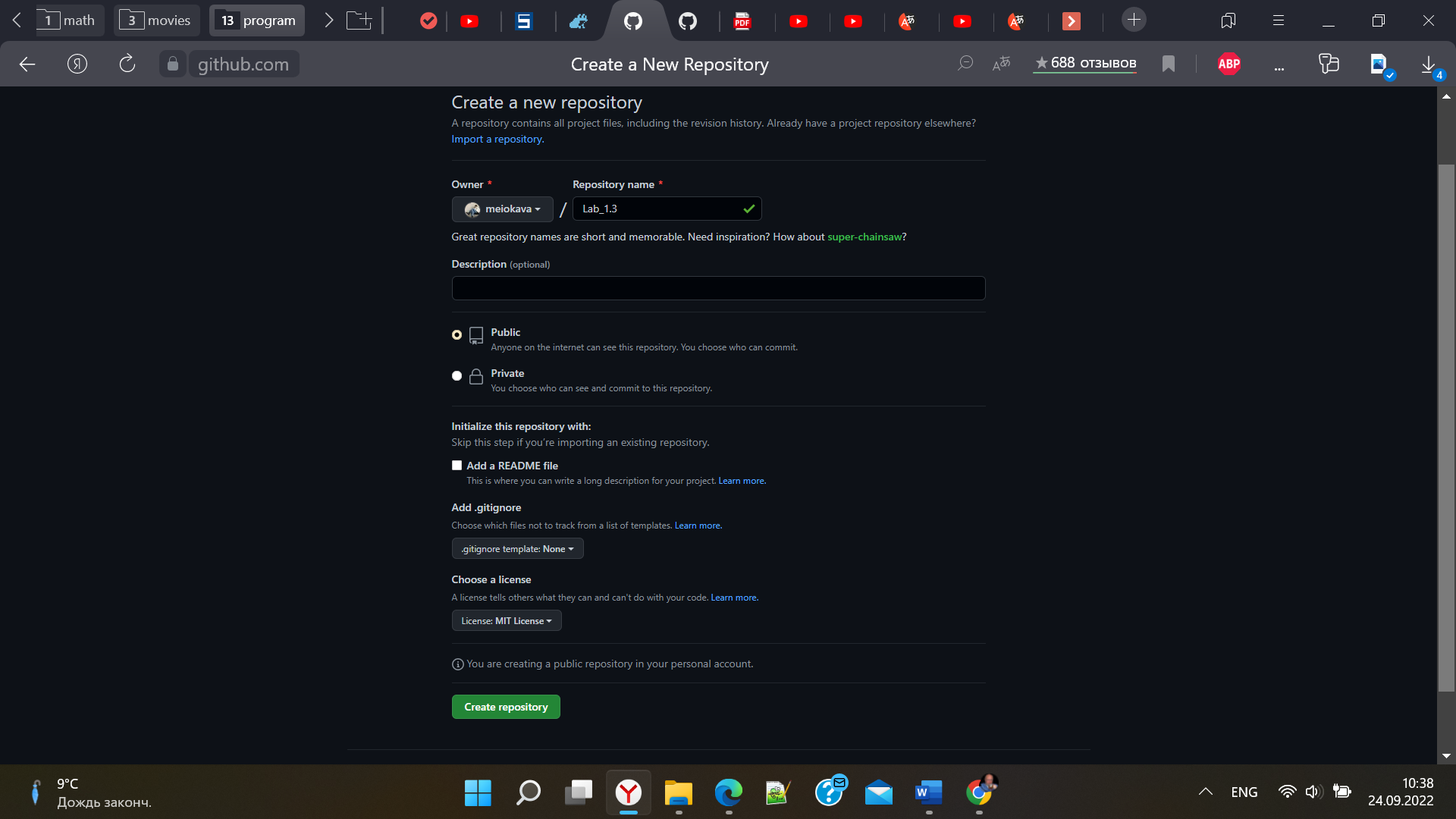


Рисунок 1.1 – Создание репозитория

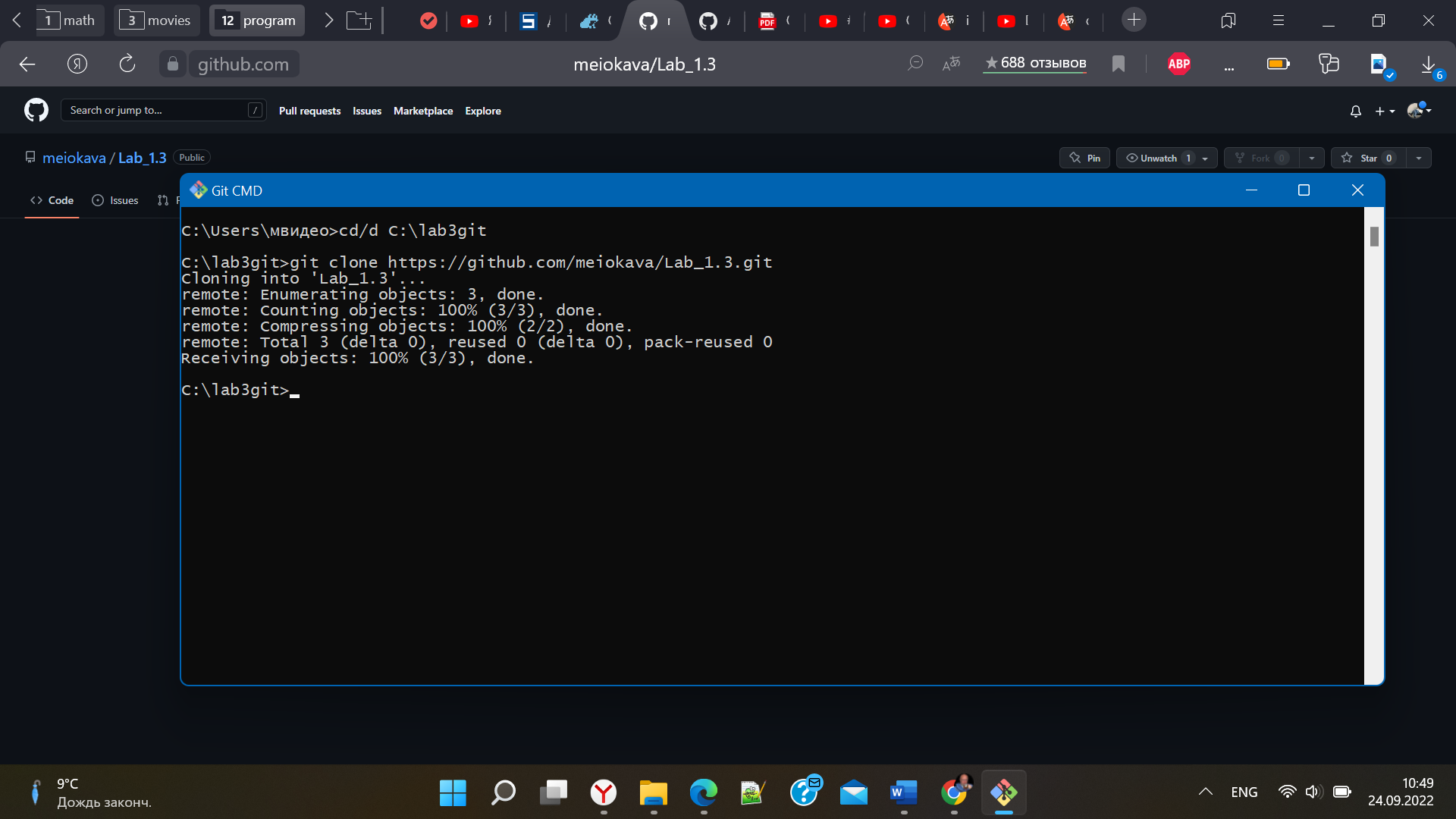


Рисунок 1.2 – Клонирование репозитория

1. Создание трех текстовых файлов

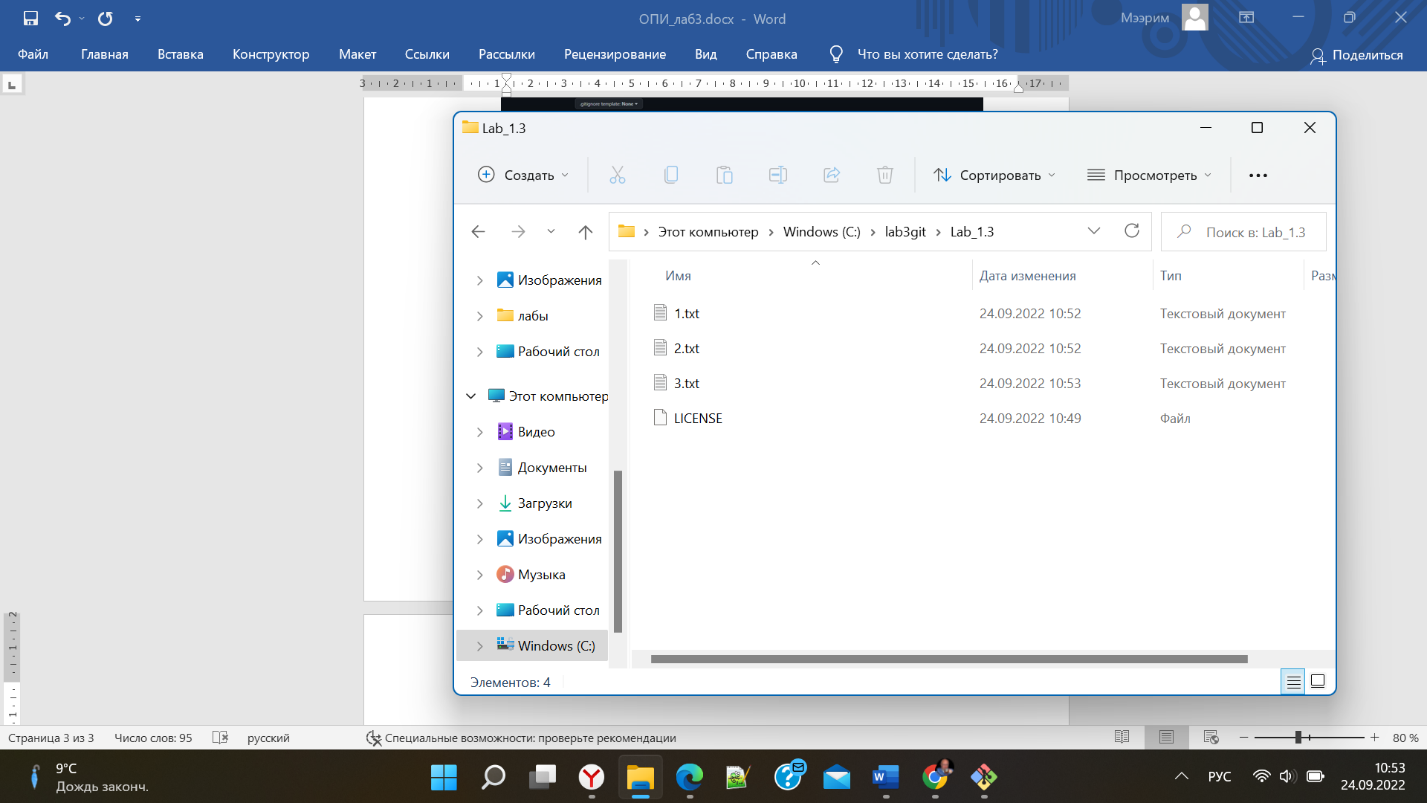


Рисунок 2.1 – Создание трех текстовых файлов

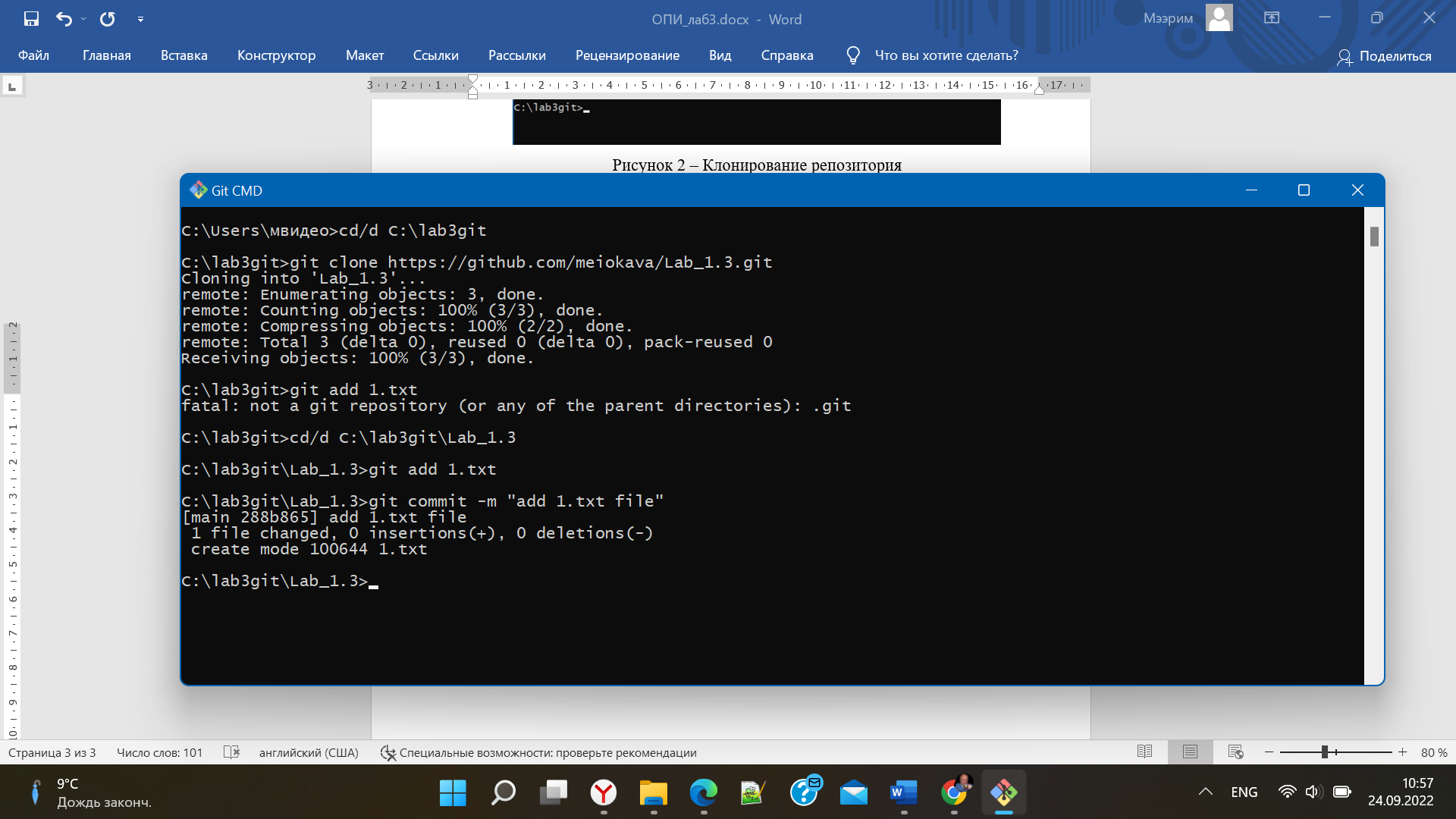


Рисунок 2.2 – Коммит с комментарием add 1.txt file

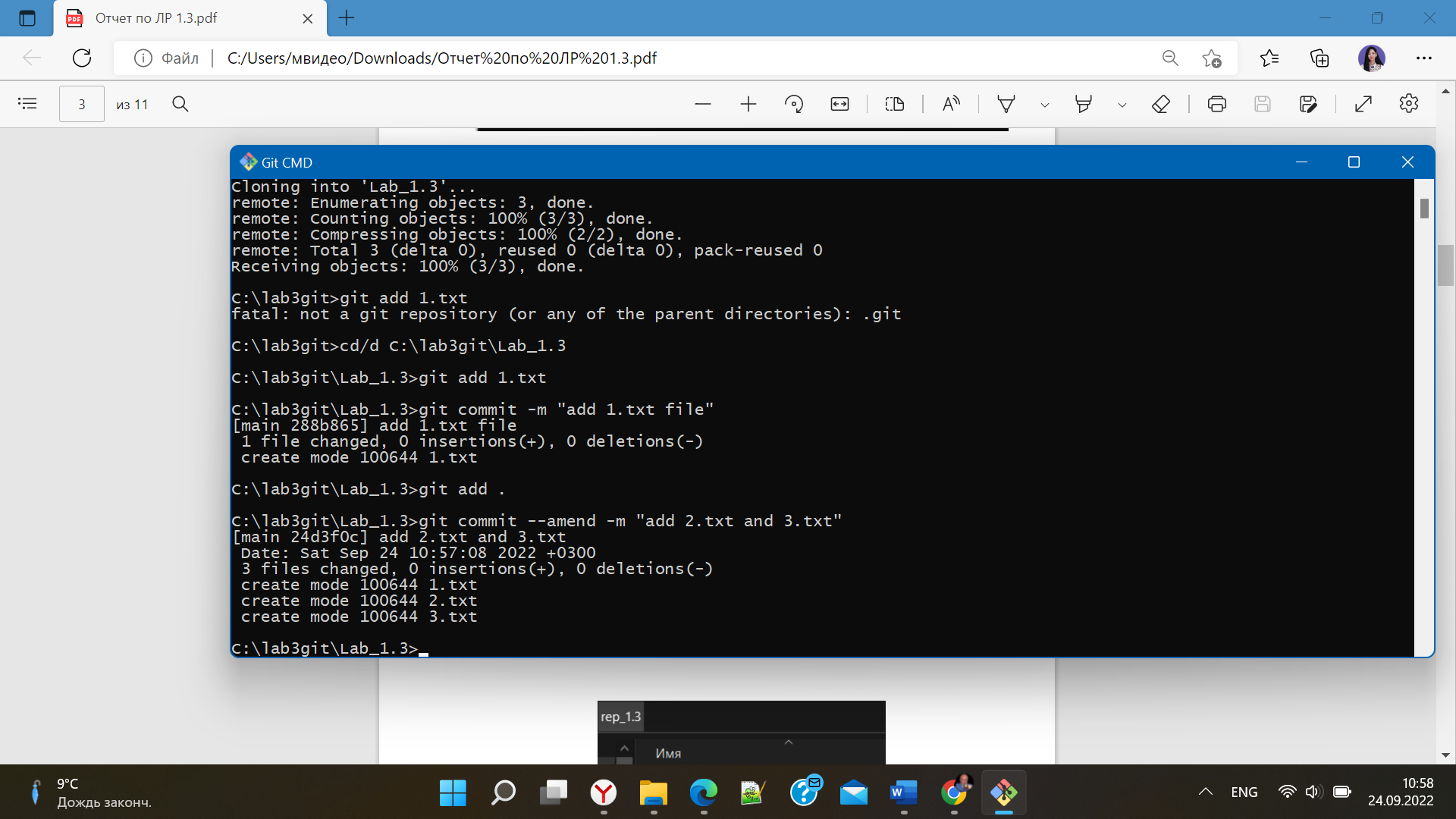


Рисунок 2.3 – индексация второго и третьего файла

1. Была создана новая ветка my\_first\_branch, был осуществлен переход на нее и создан новый файл в нем in\_branch.txt, были закоммичены изменения. После этого произошел возврат на ветку master, и переход на ветку new\_branch, были осуществлены изменения в файле 1.txt и закоммичены изменения.

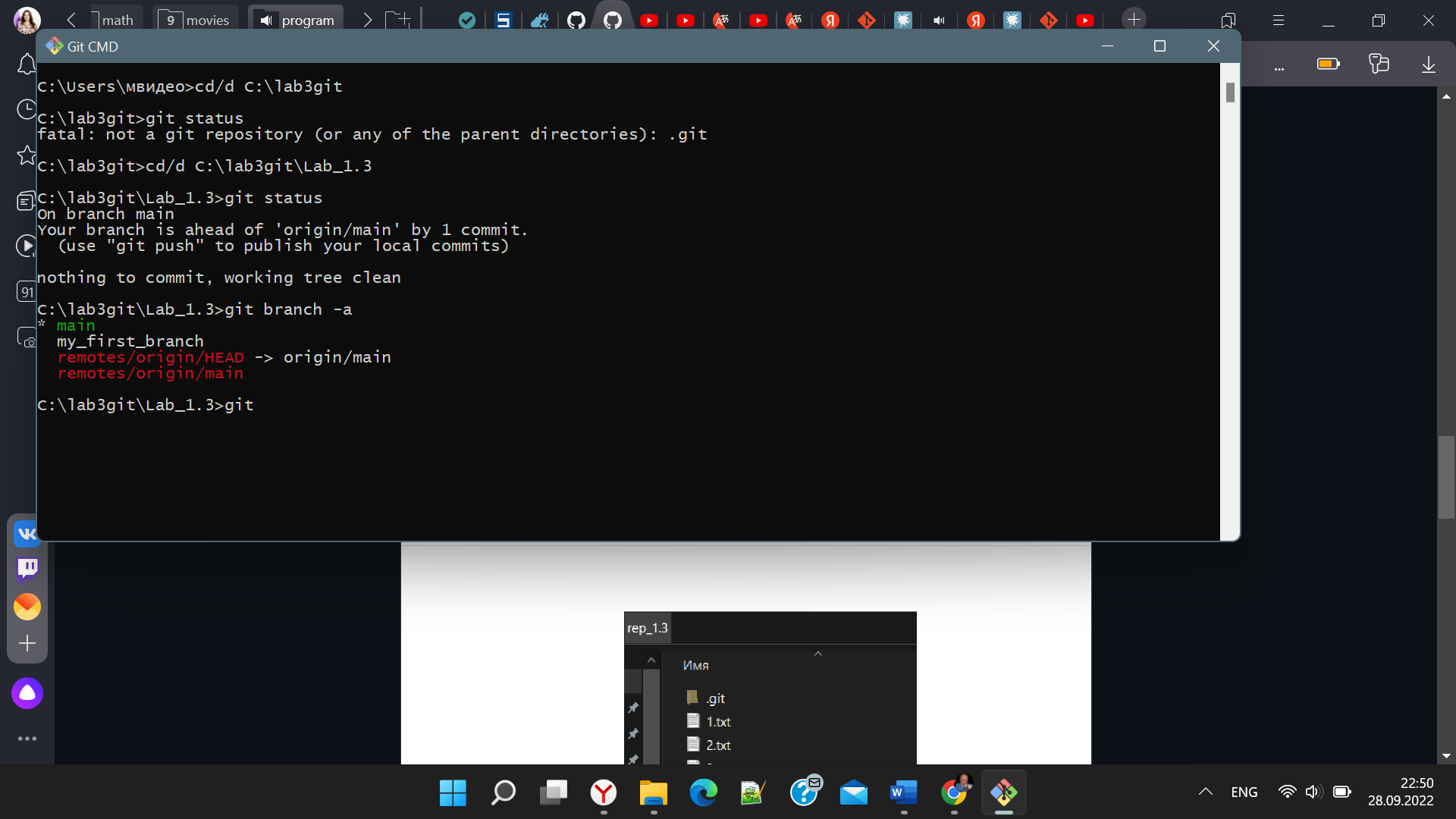


Рисунок 3.1 – Отображение веток, а также текущей ветки

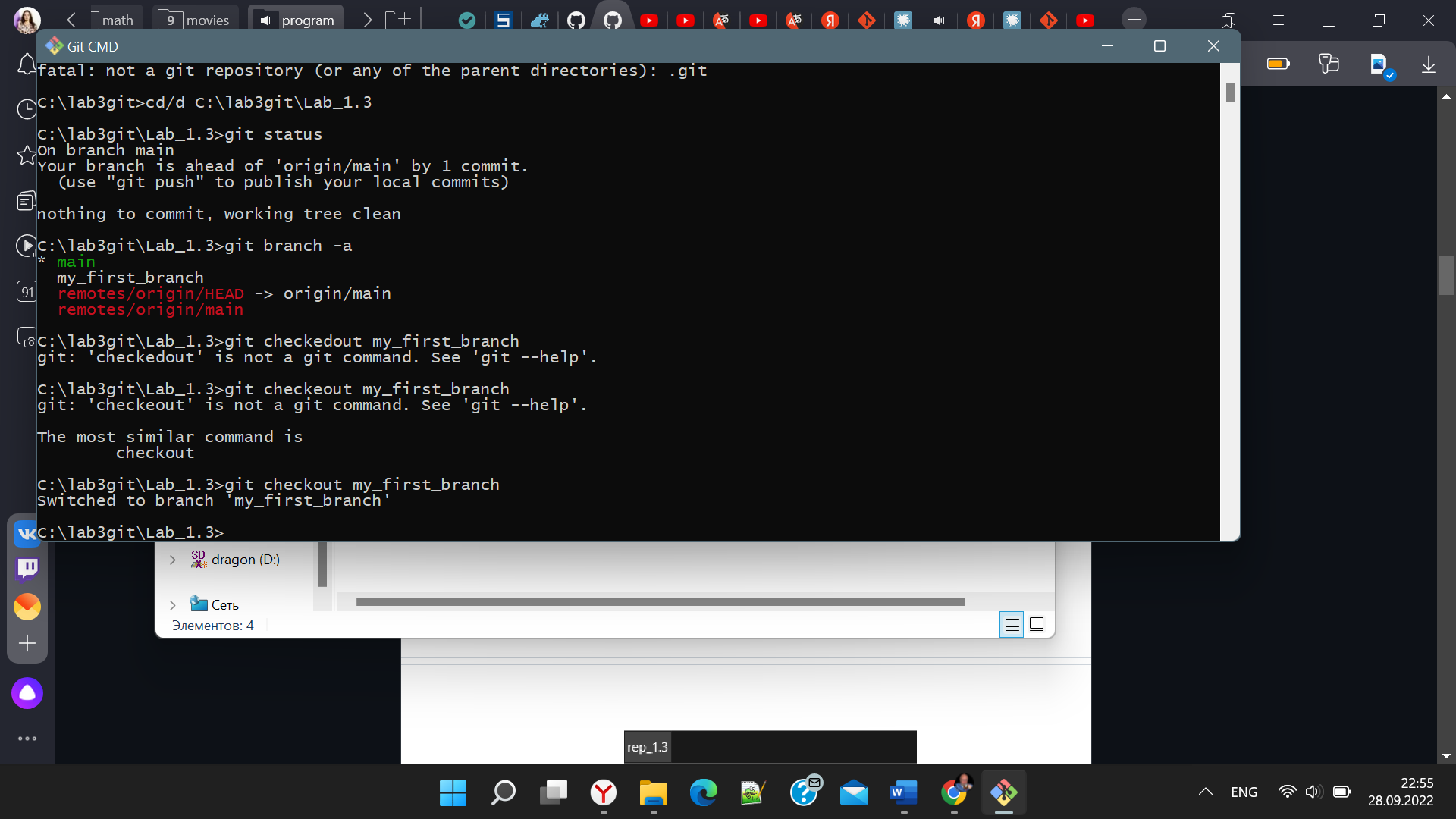


Рисунок 3.2 – Переход на ветку my\_first\_branch

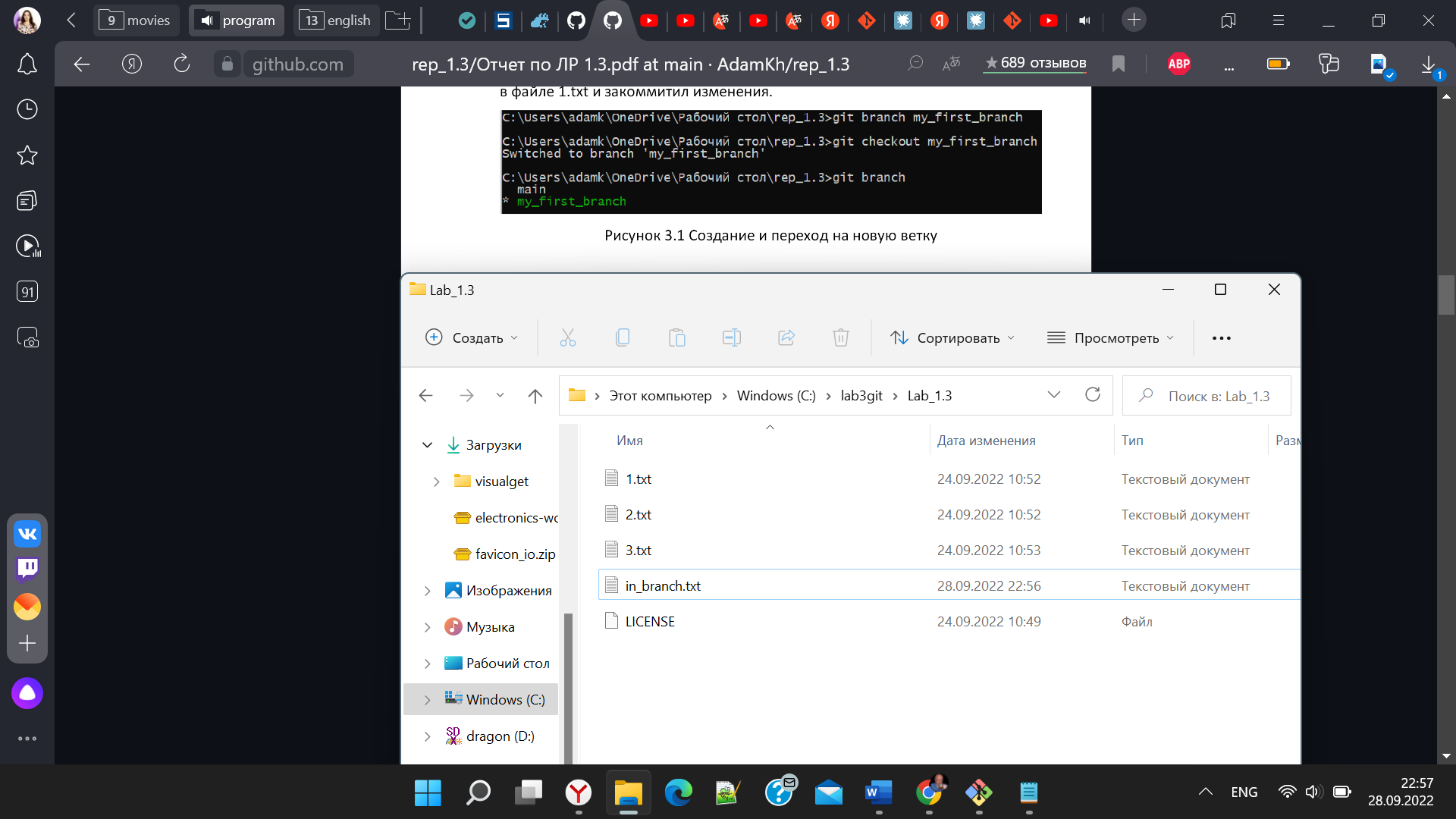


Рисунок 3.3 – Добавление нового текстового файла в репозиторий

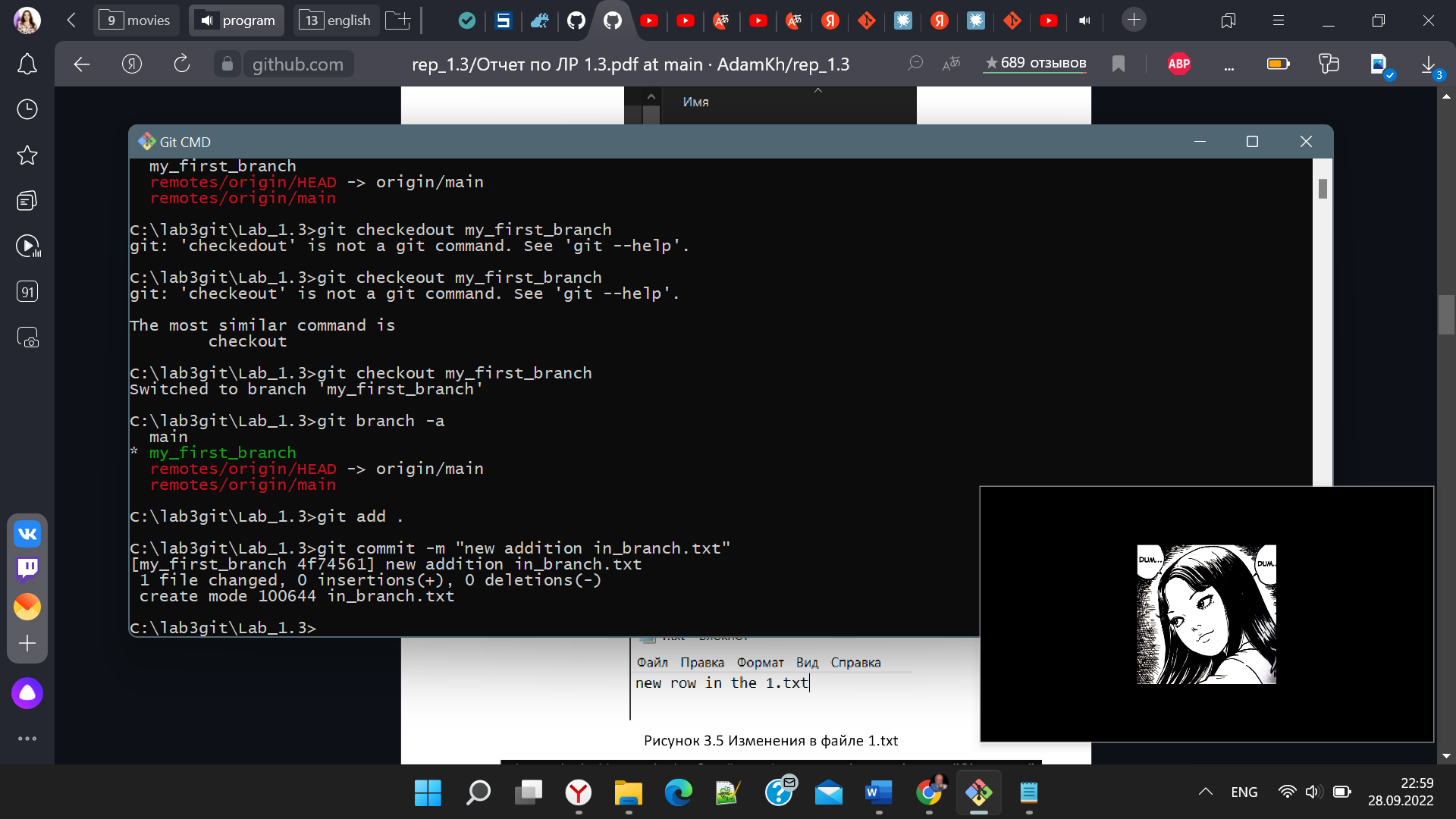


Рисунок 3.4 – Коммит сделанных изменений

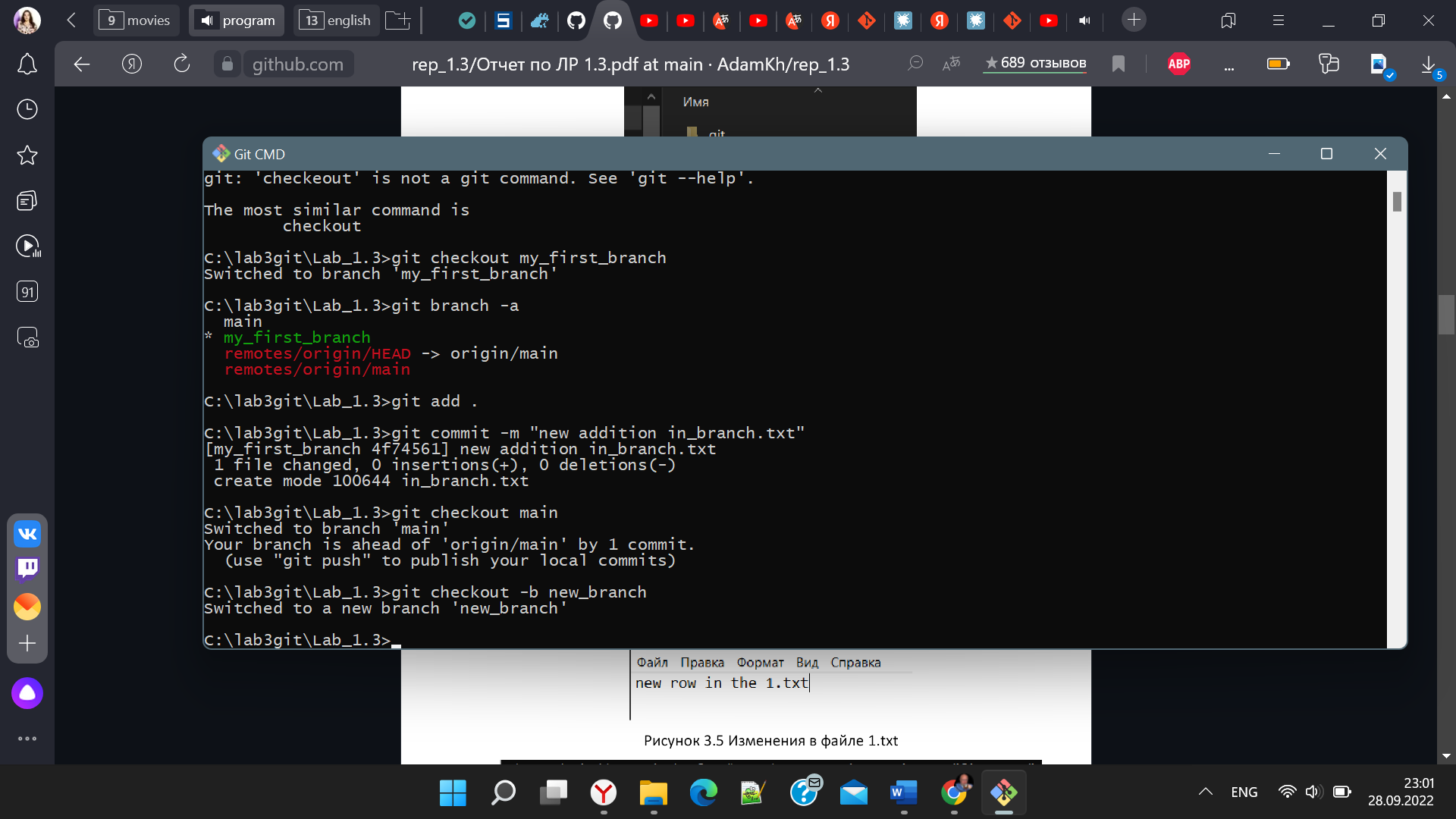


Рисунок 3.5 – Переход на главную ветвь и создание с переходом на новую ветку

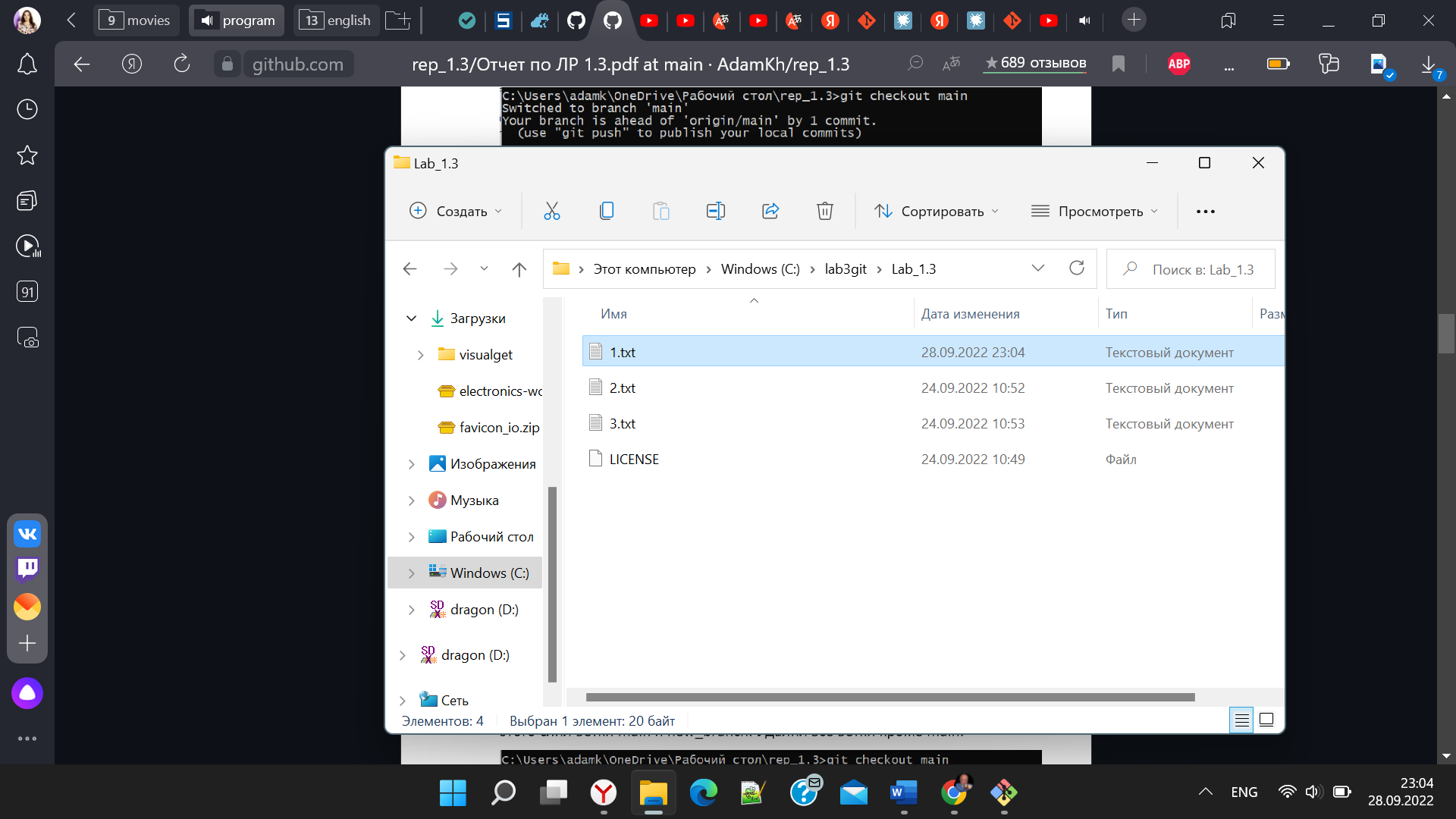


Рисунок 3.6 – В новой ветви нет добавленного файла в ветви my\_first\_branch

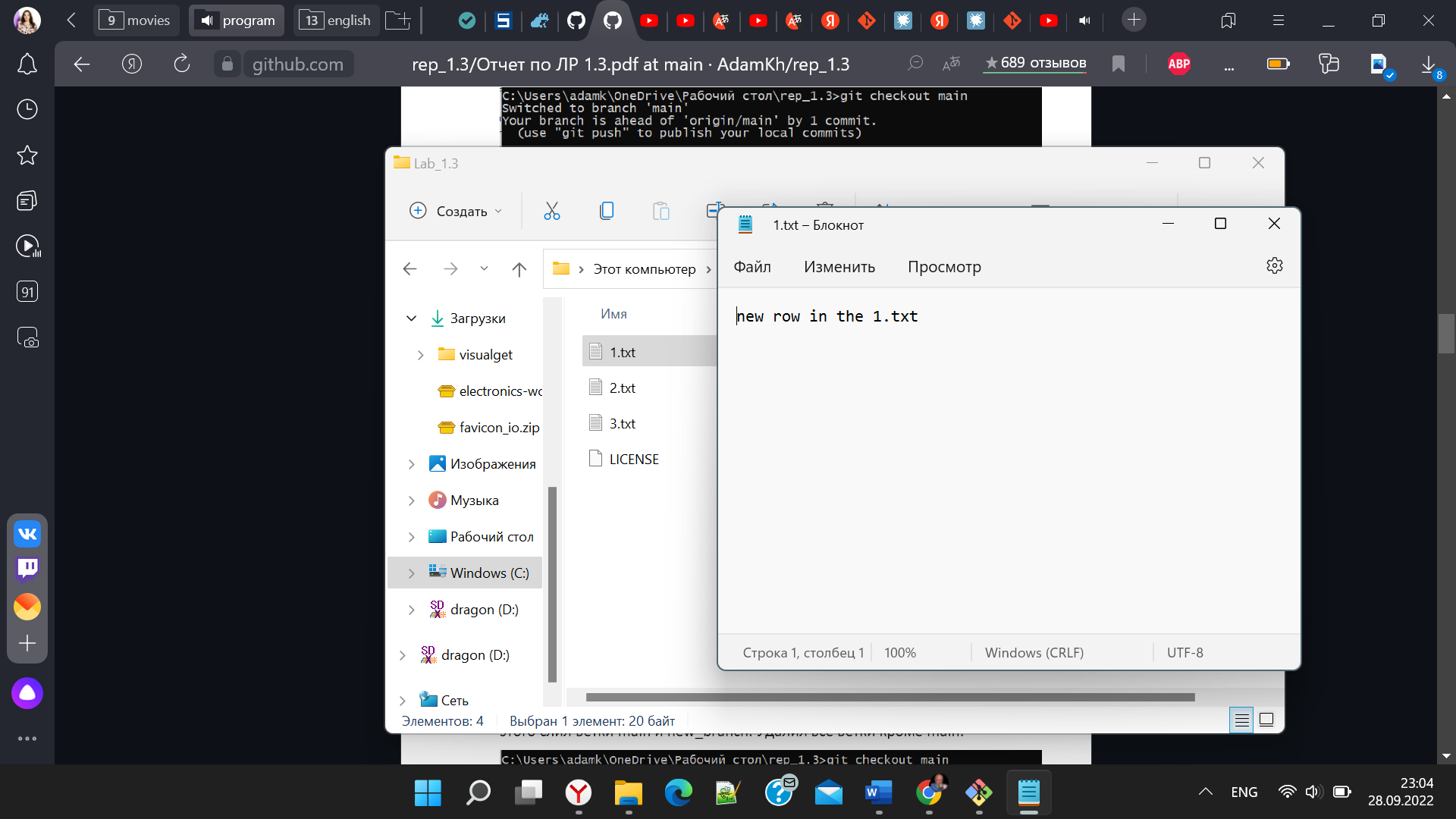


Рисунок 3.7 – Изменения в файле 1.txt

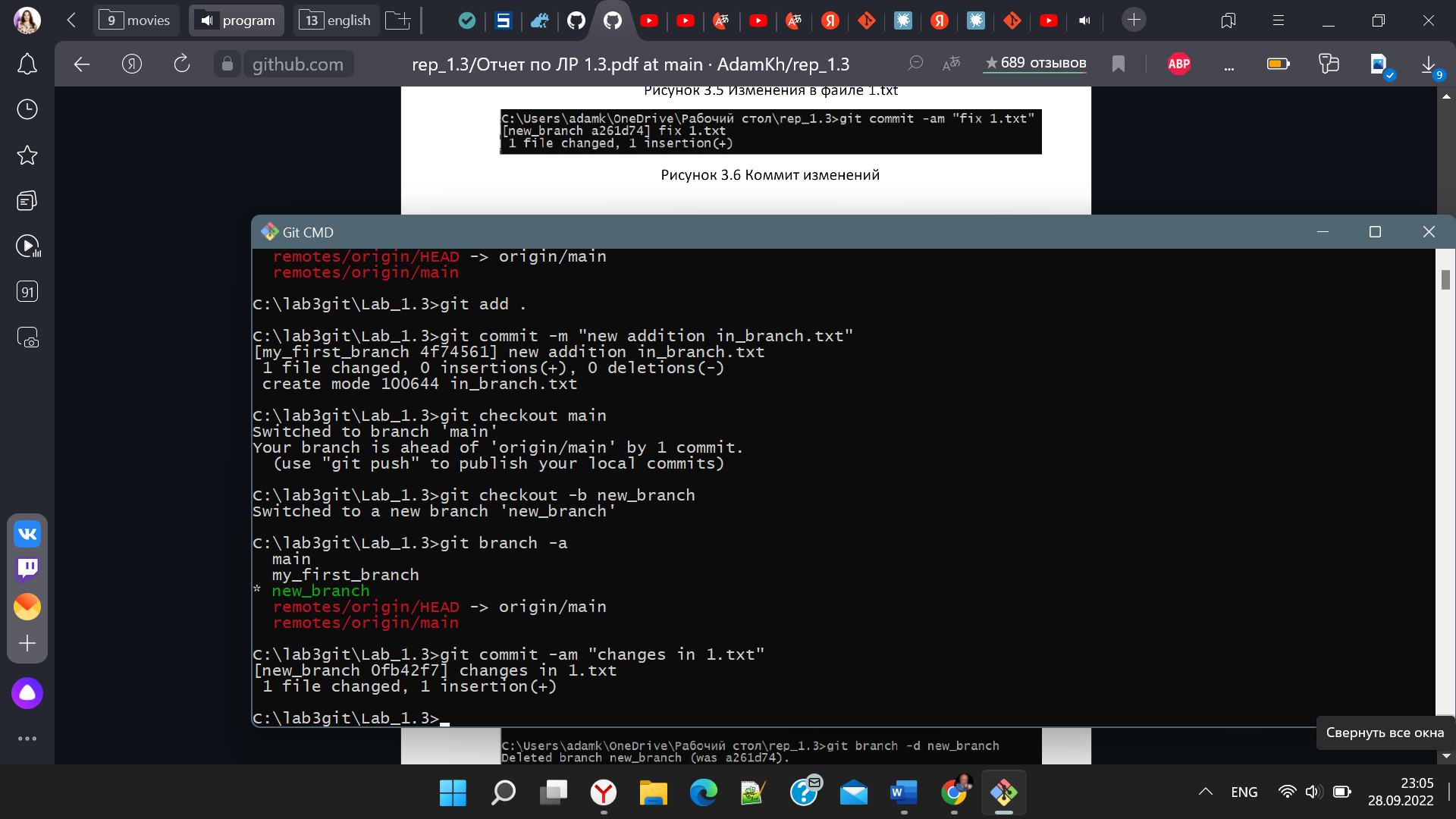


Рисунок 3.8 – Коммит изменений

1. Был осуществлен переход на ветку main также были слиты ветки main и my\_first\_branch. После были удалены все ветки кроме main.

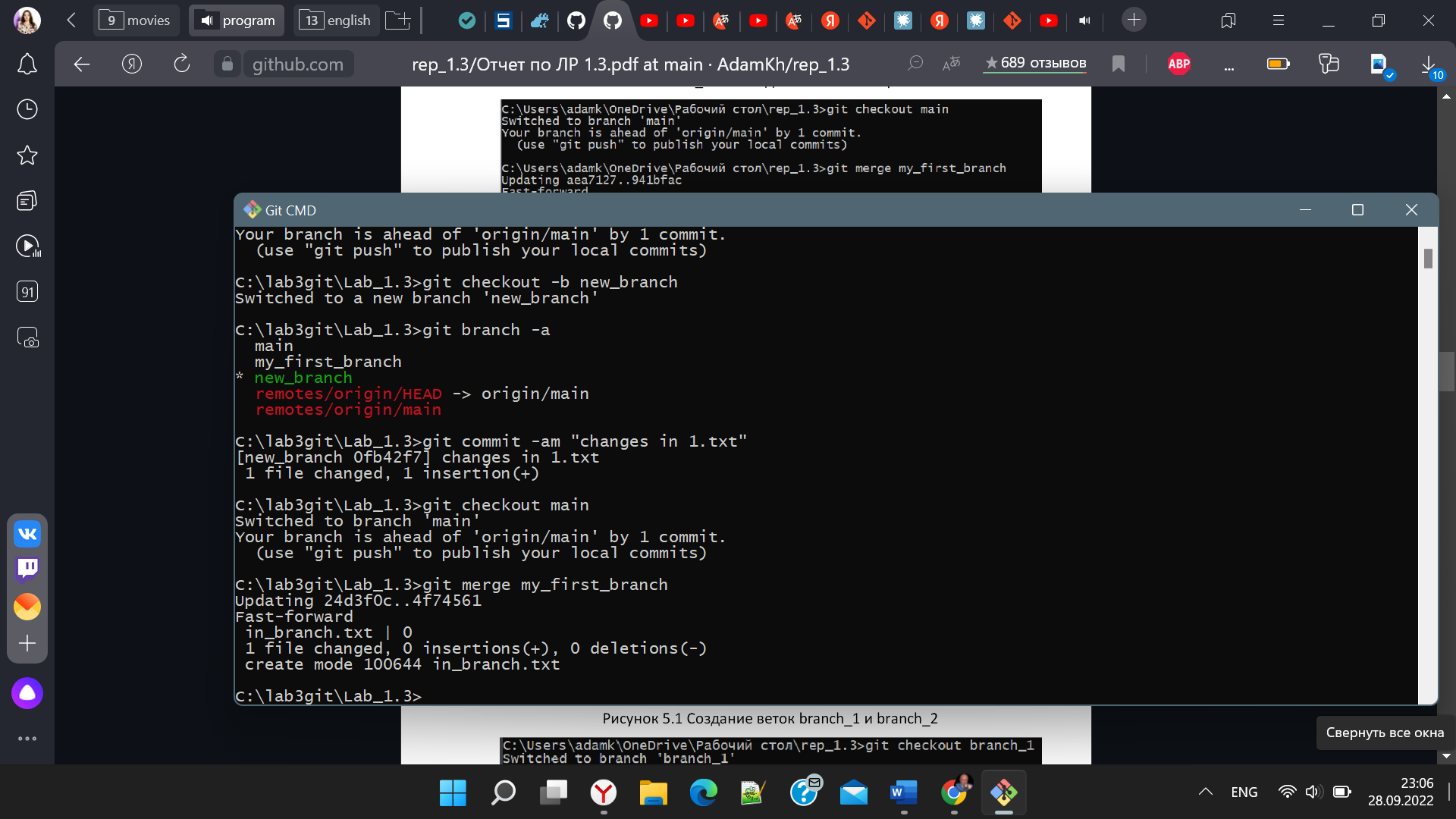


Рисунок 4.1 – Переход на ветку main и слияние ее с веткой my\_first\_branch

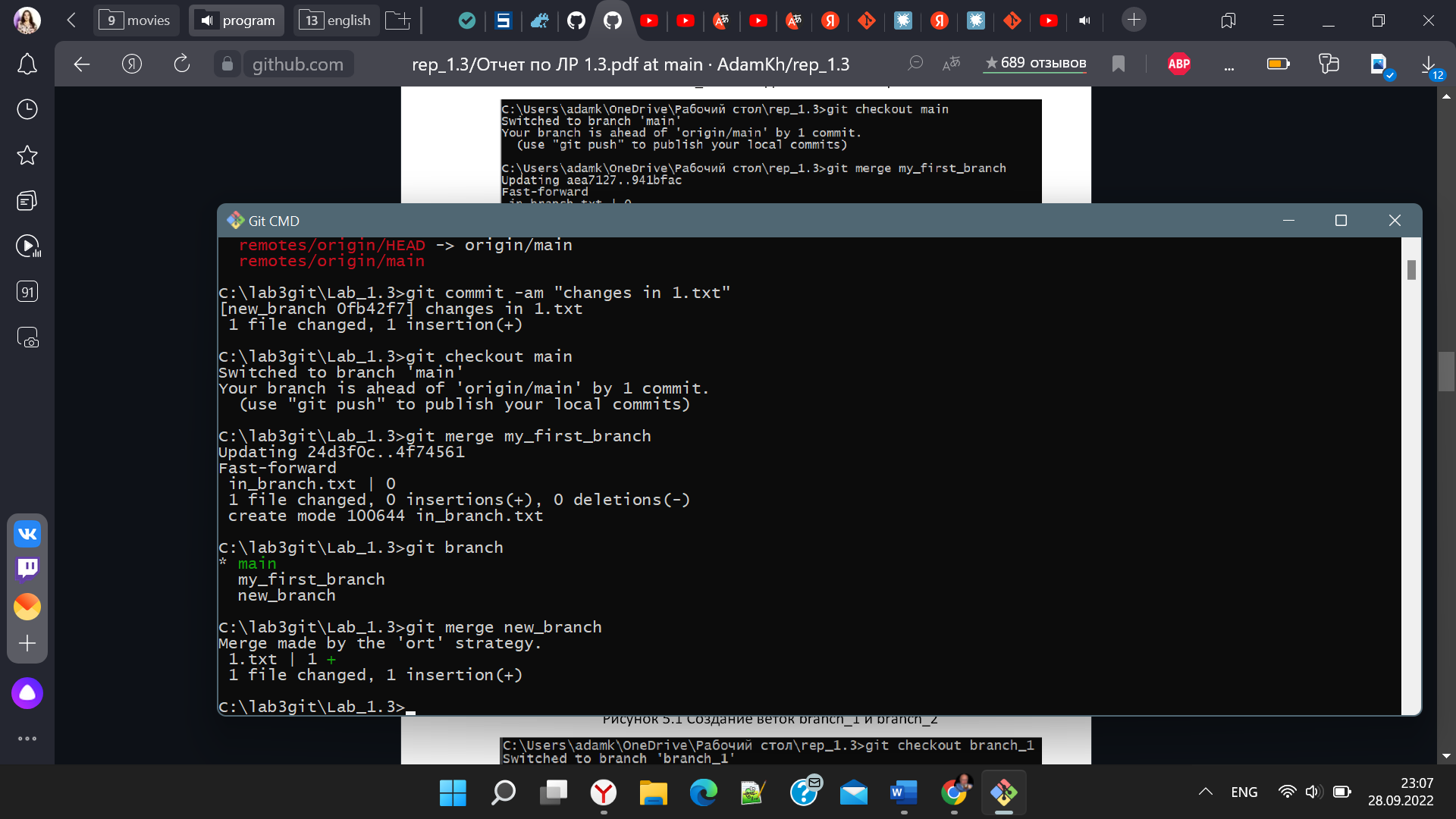


Рисунок 4.2 – Слияние ветки main и ветки new\_branch

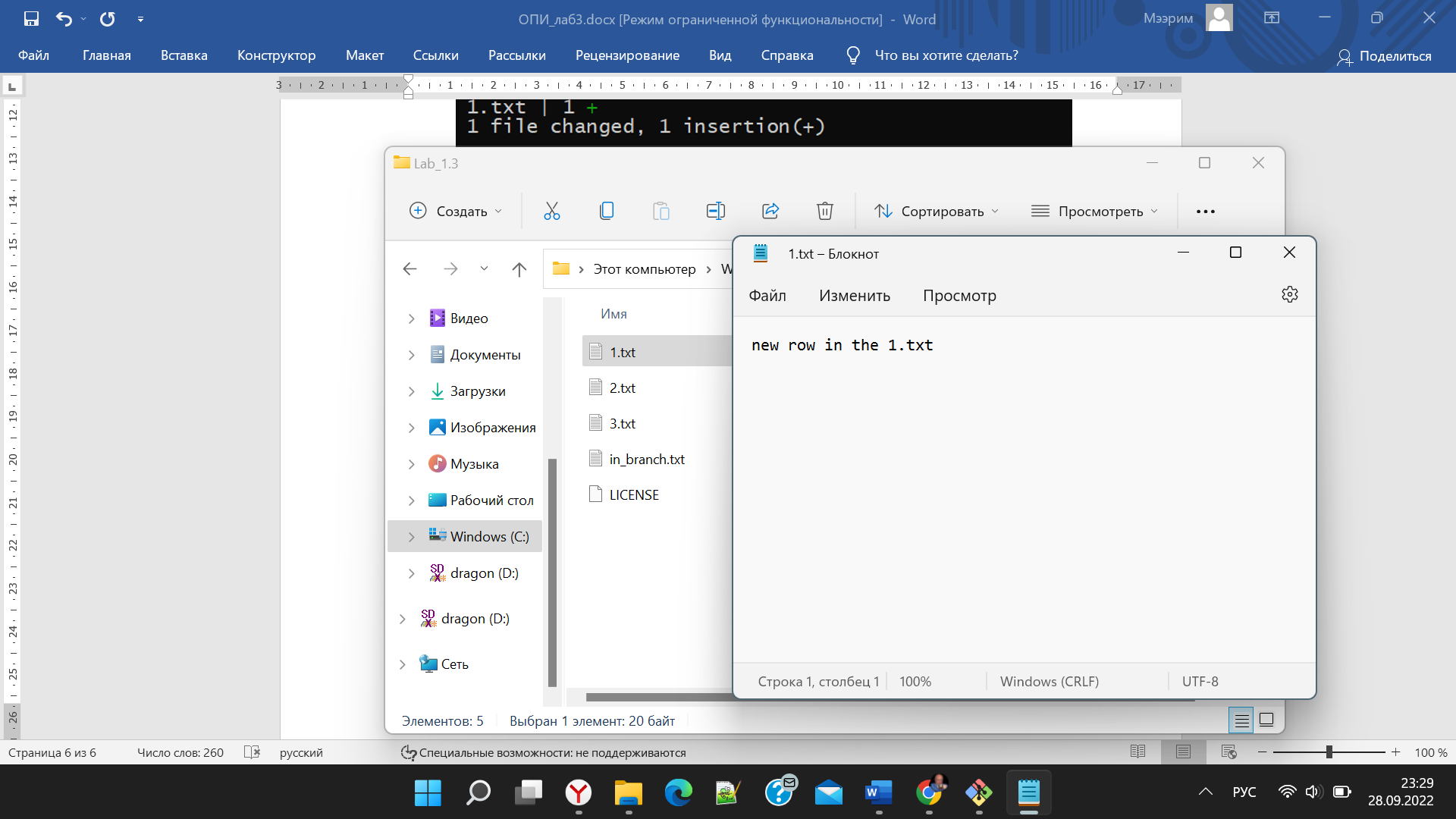


Рисунок 4.3 – Вид репозитория на ветви main

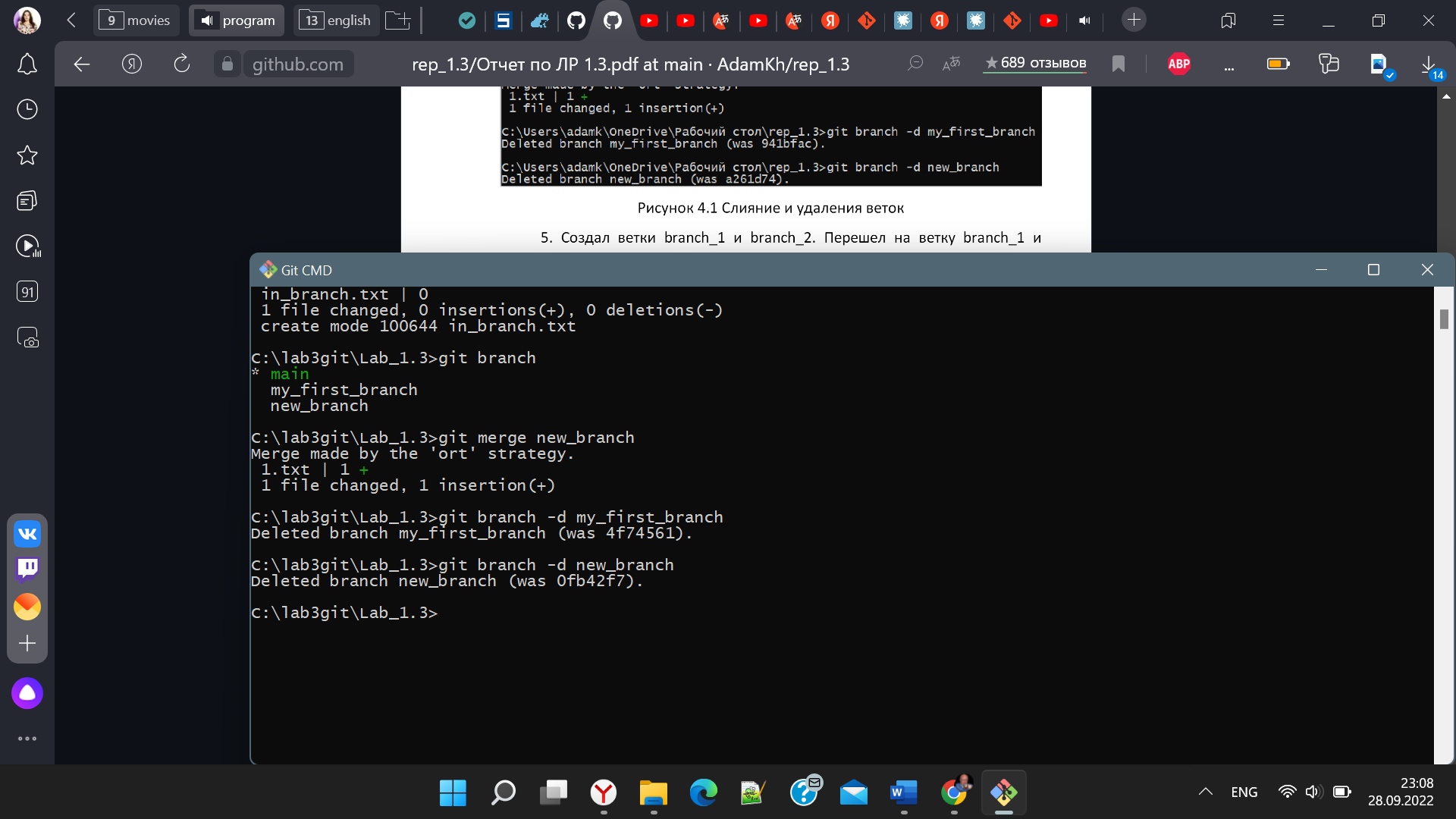


Рисунок 4.4 – Удаление ветвей my\_first\_branch и new\_branch

1. Создание веток branch\_1 и branch\_2. Переход на ветку branch\_1 и изменение файлов 1.txt, 3.txt. Затем коммит изменений. Осуществление перехода на ветку branch\_2 и изменения файлов 1.txt, 3.txt, после чего фиксация коммитов изменения. После слияние изменения ветки branch\_2 в ветку branch\_2.

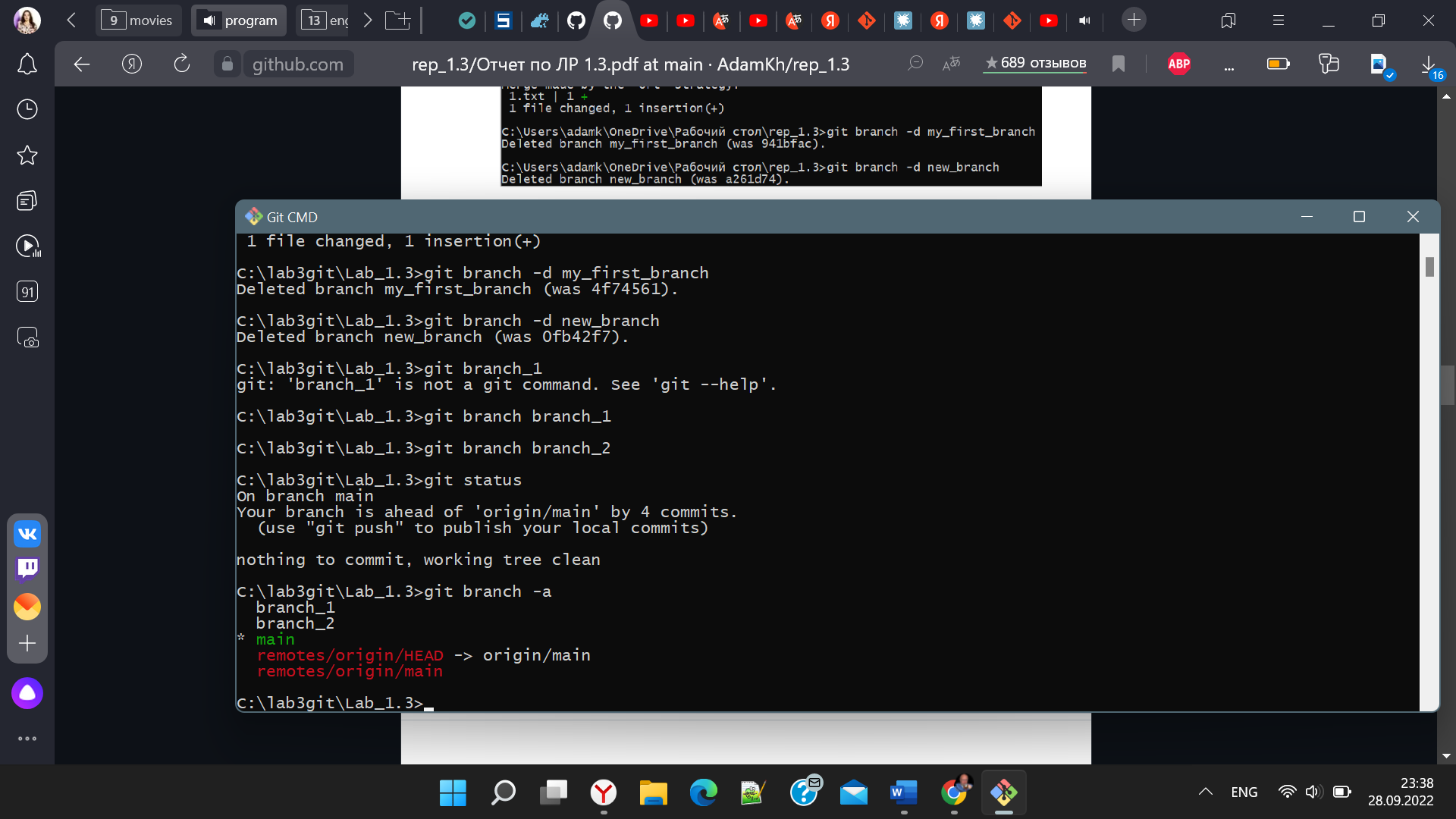


Рисунок 5.1 – Создание веток branch\_1 и branch\_2

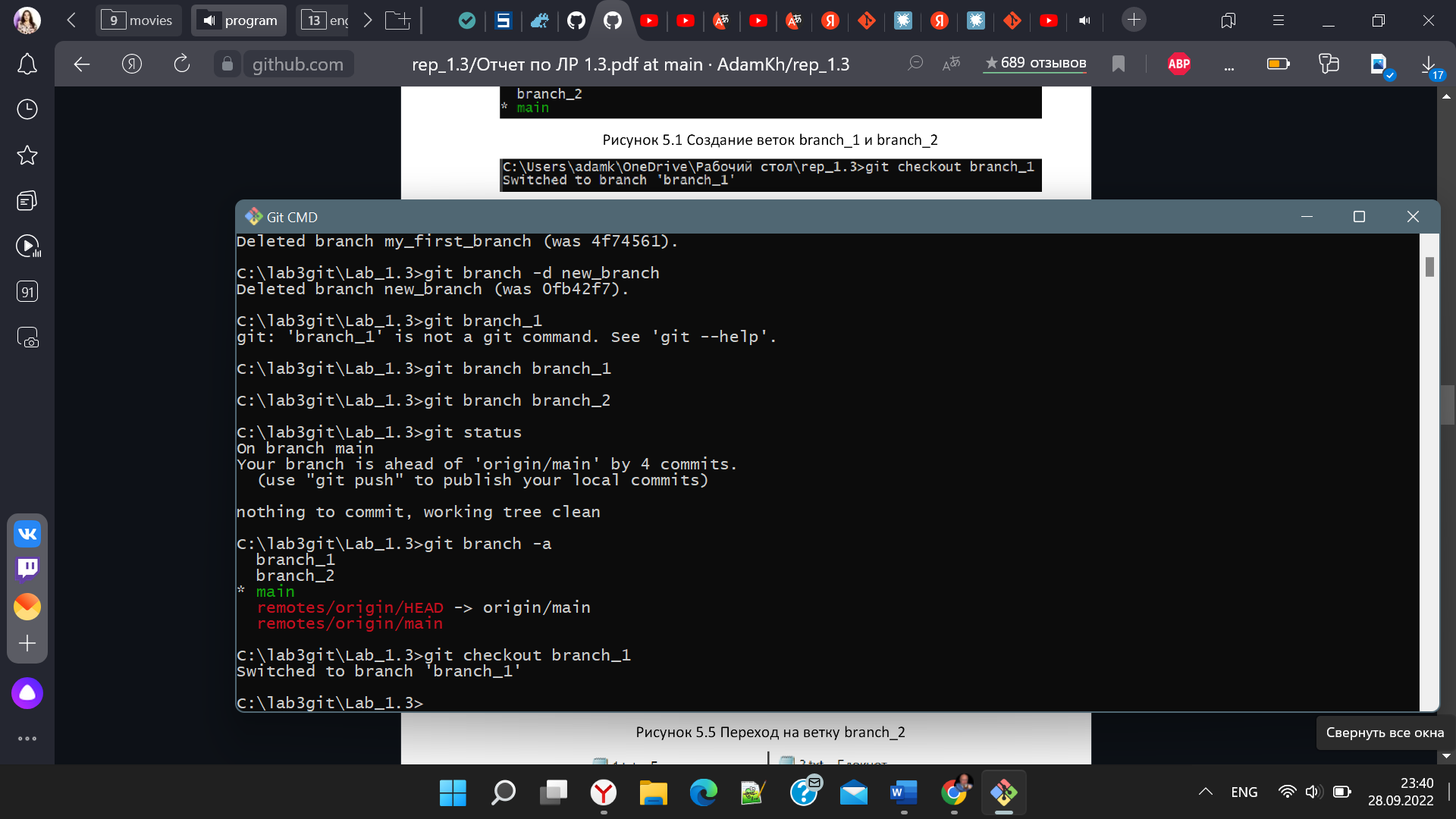


Рисунок 5.2 – Переход на ветвь branch\_1

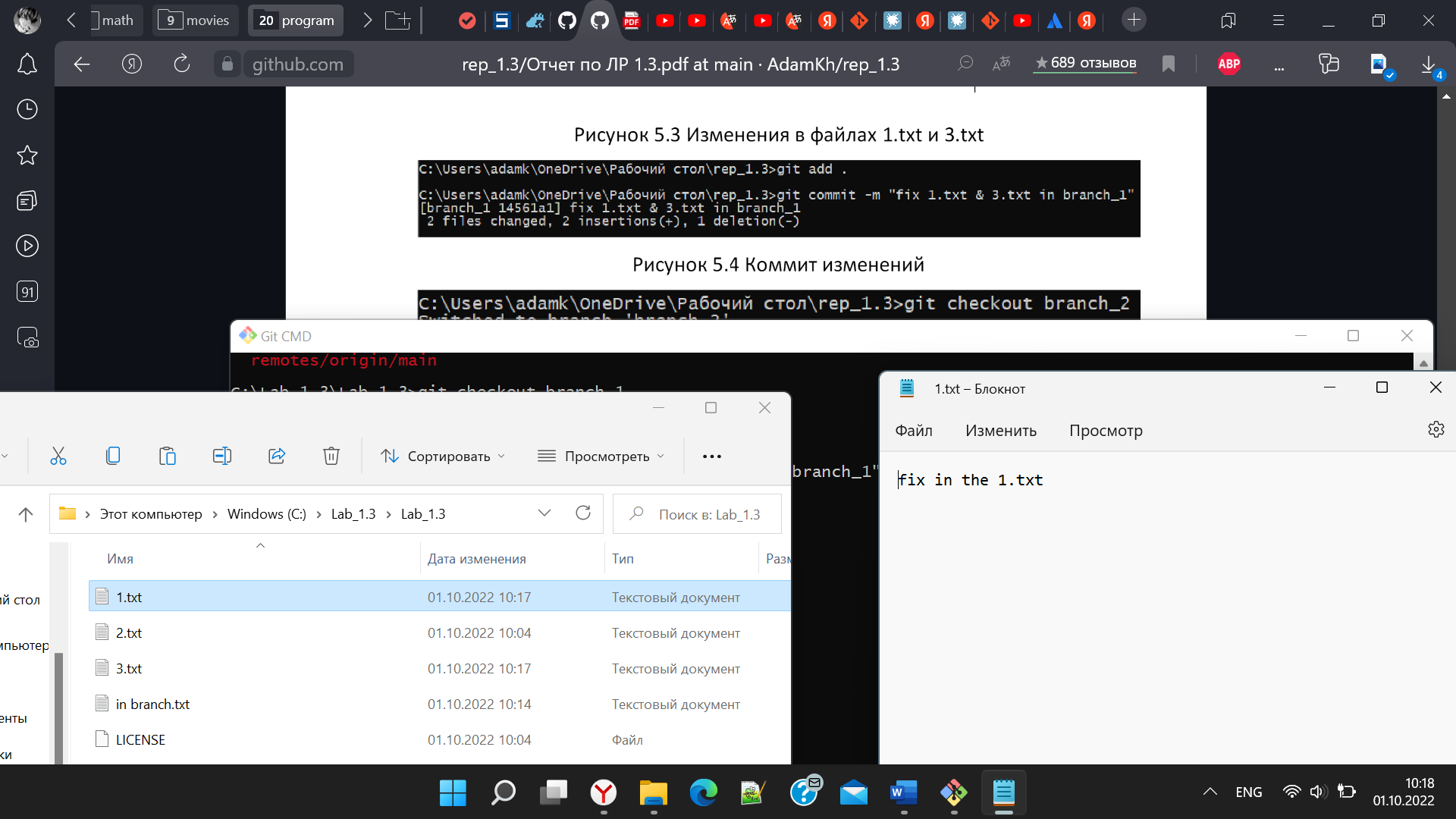


Рисунок 5.3 – Изменения в файле 1.txt

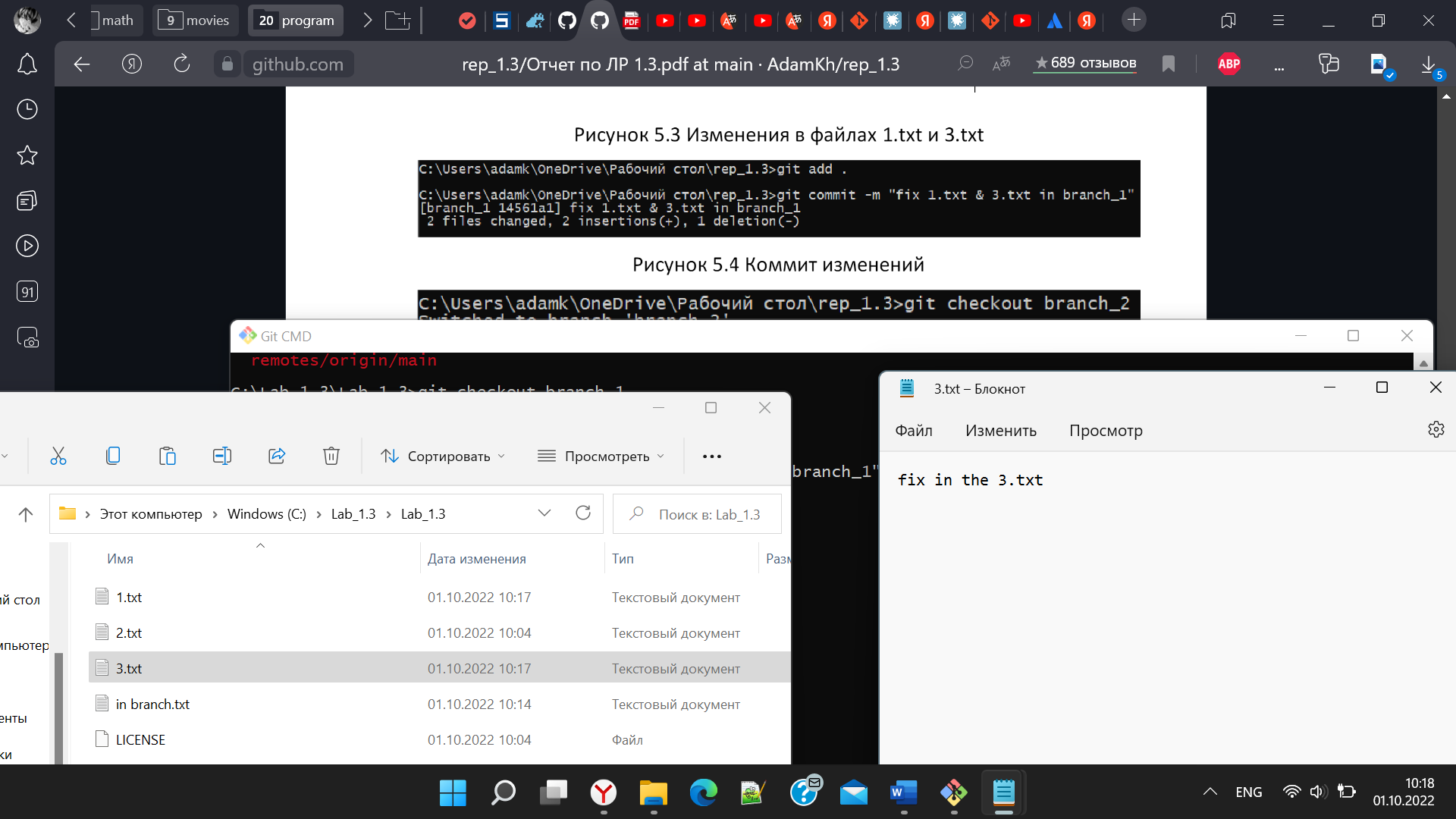


Рисунок 5.4 – Изменения в файле 3.txt

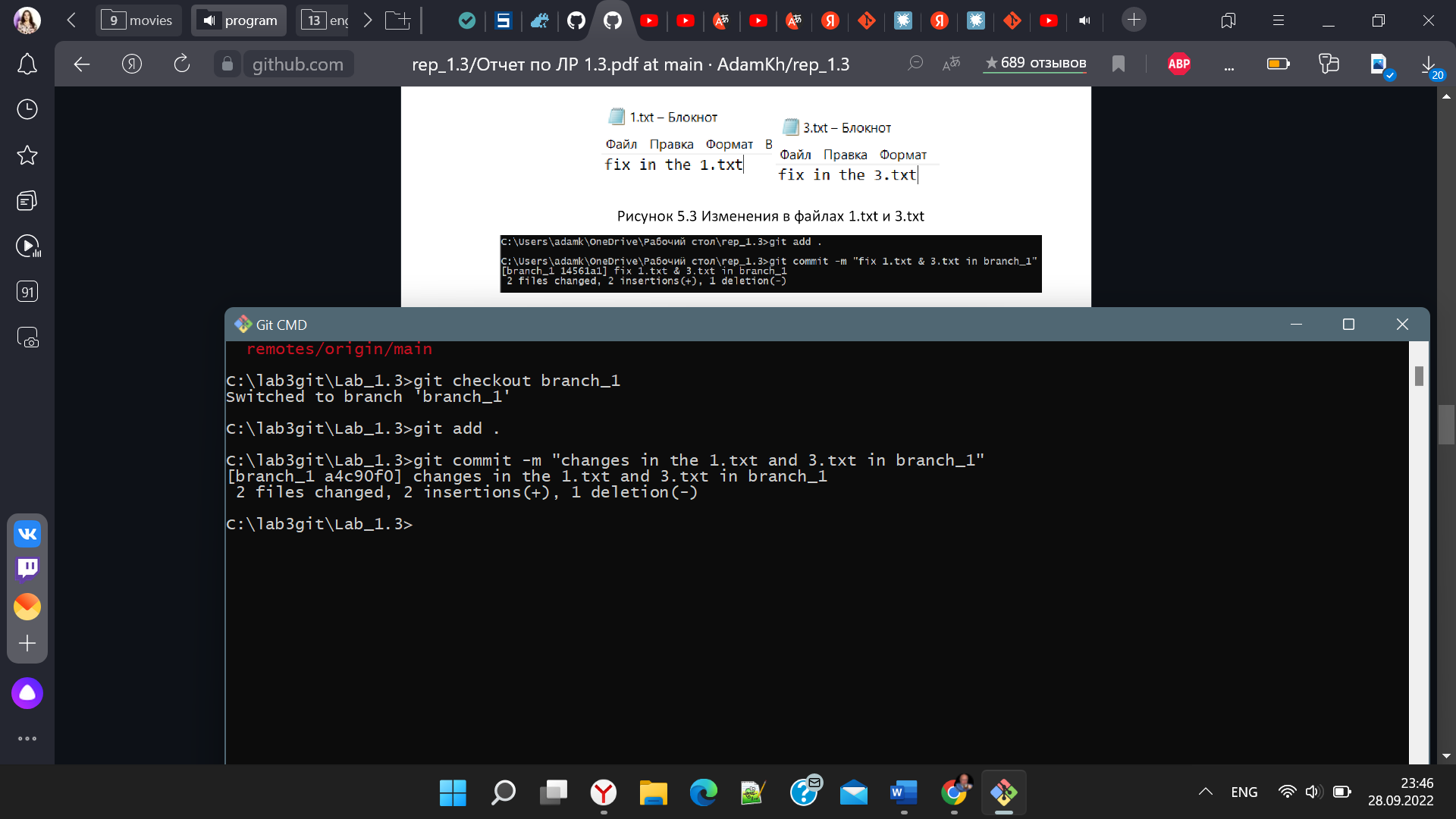


Рисунок 5.5 – Коммит изменений

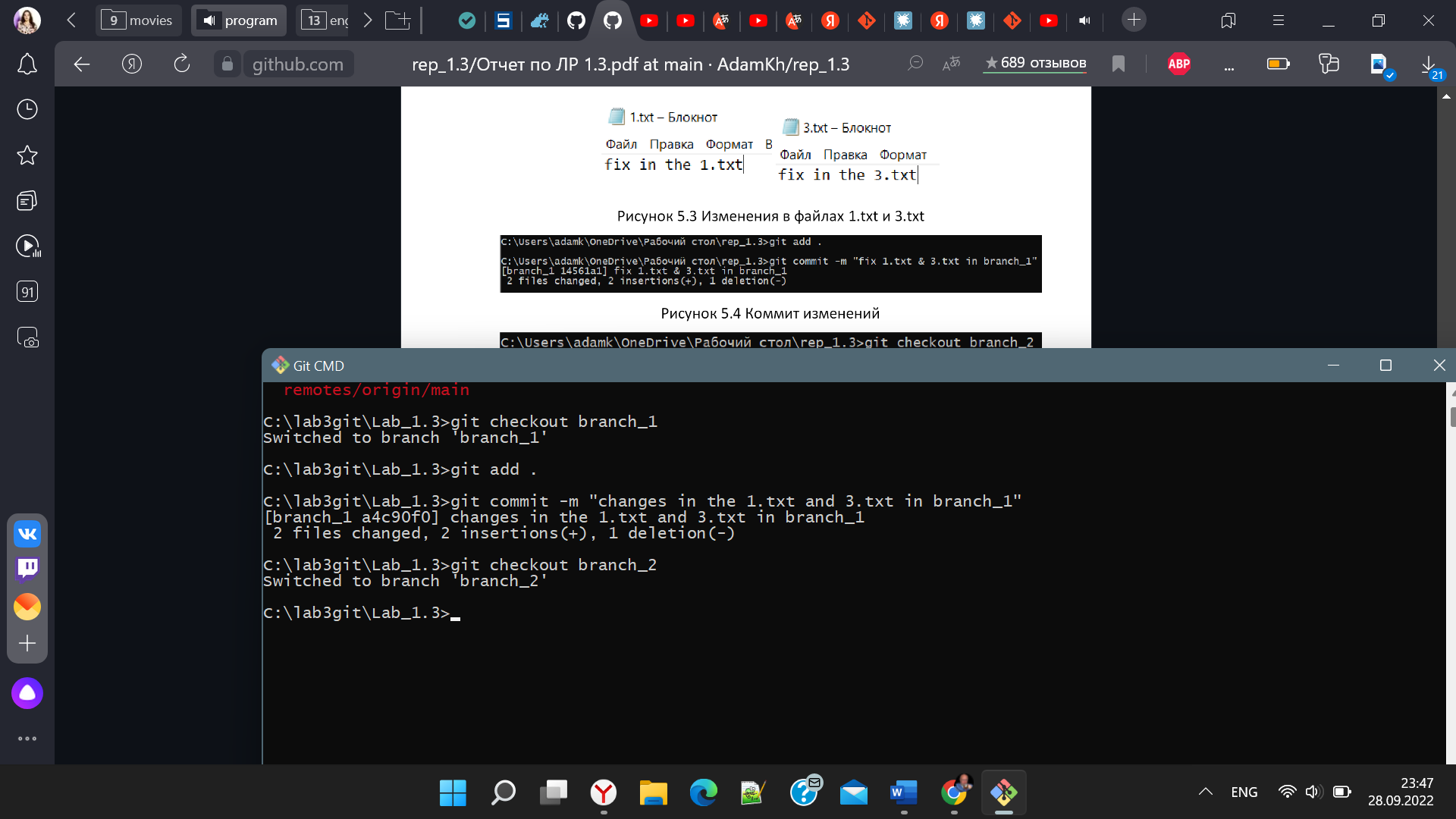


Рисунок 5.6 – Переход на ветку branch\_2

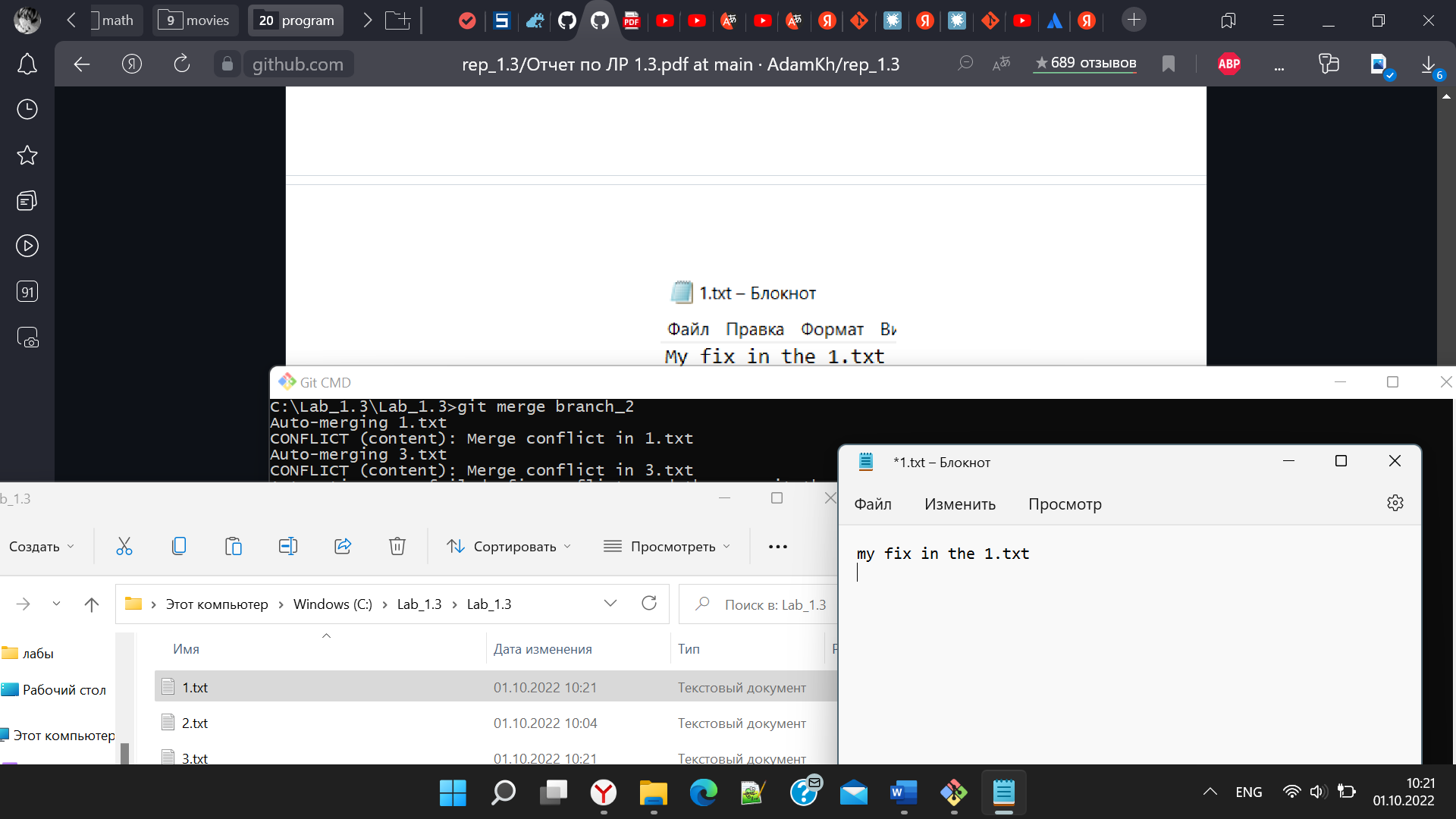


Рисунок 5.7 – Изменения в файле 1.txt

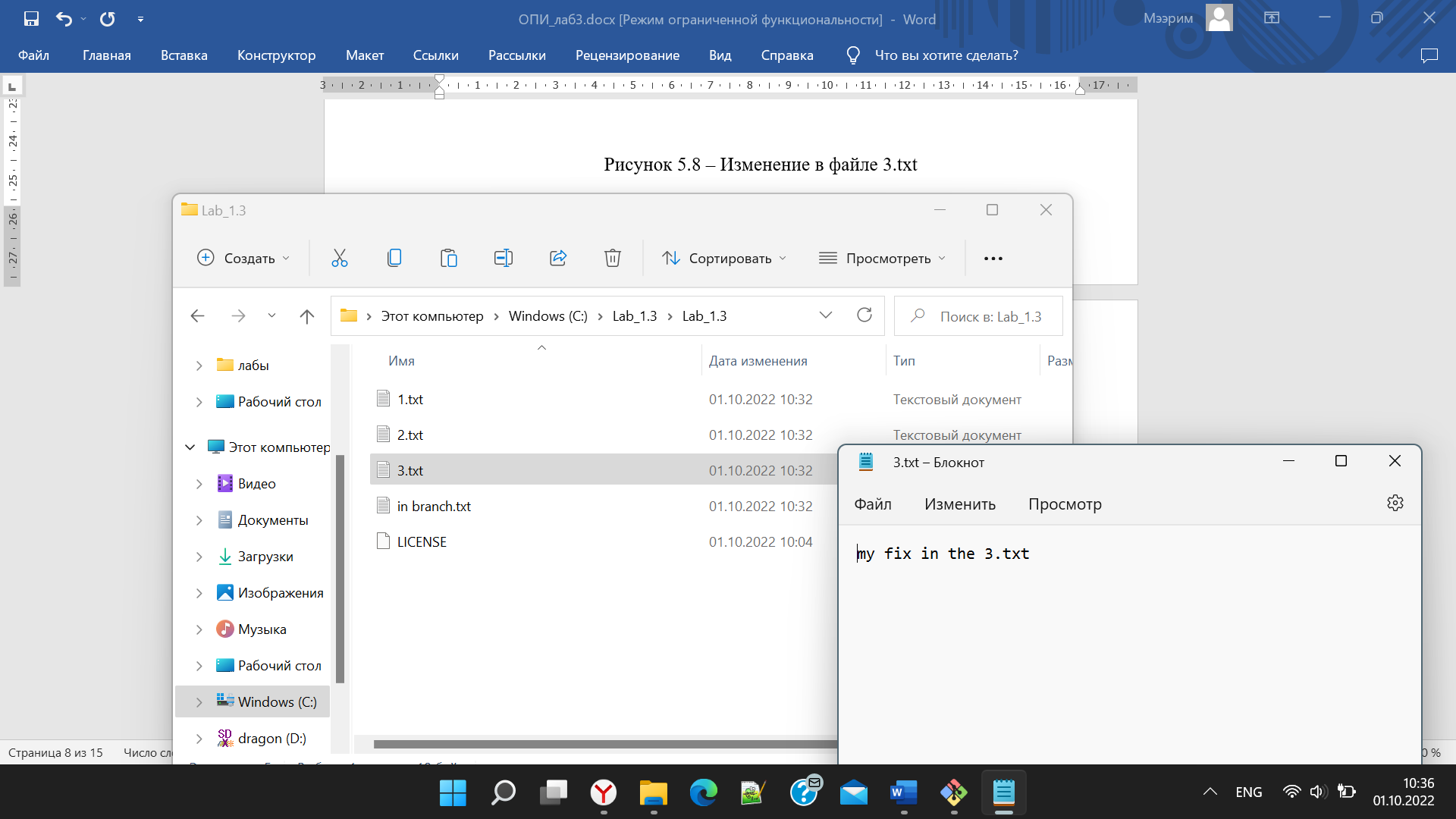


Рисунок 5.8 – Изменение в файле 3.txt

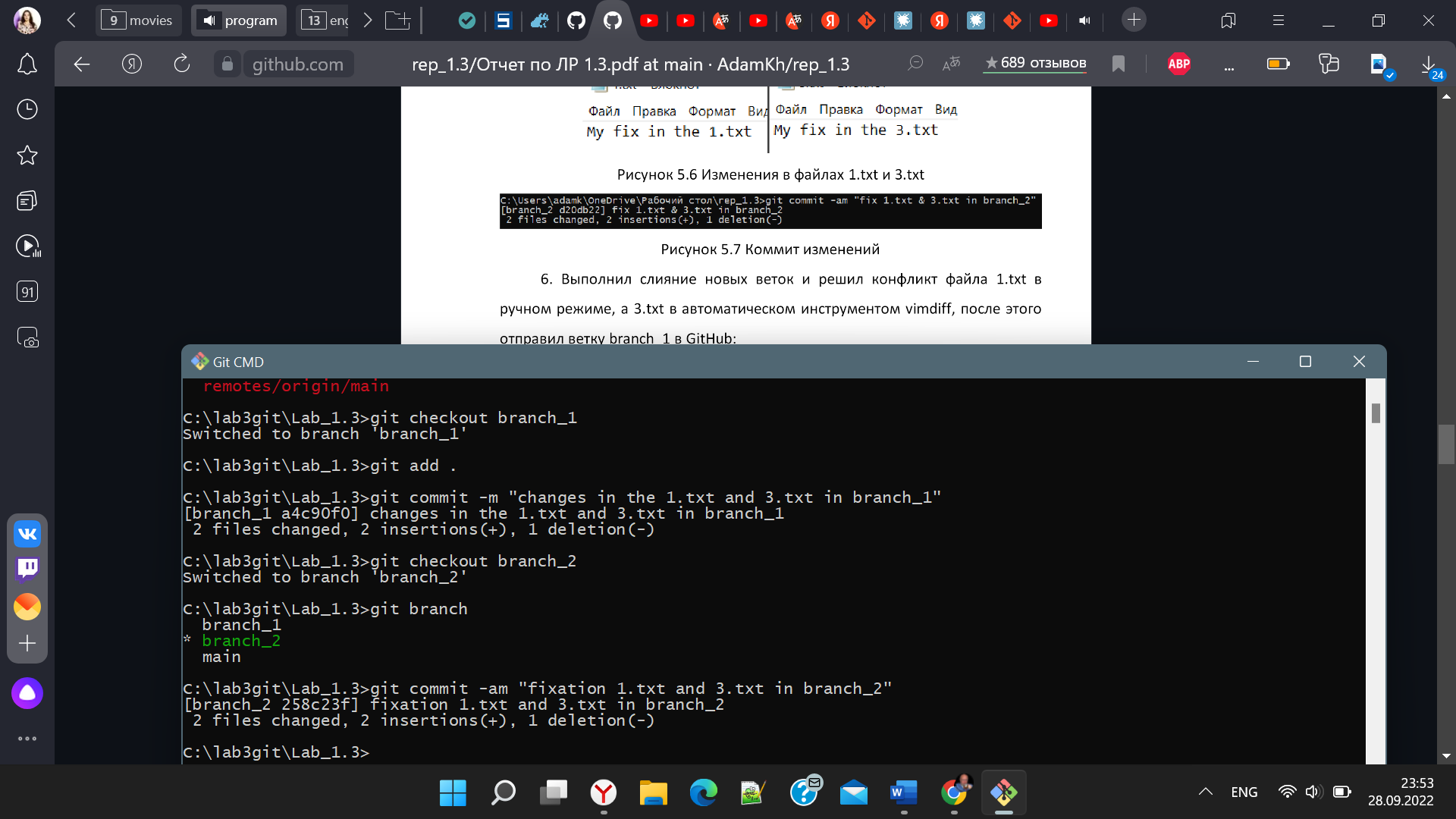


Рисунок 5.9 – Коммит изменений

1. Выполнение слияния веток и устранение конфликтов

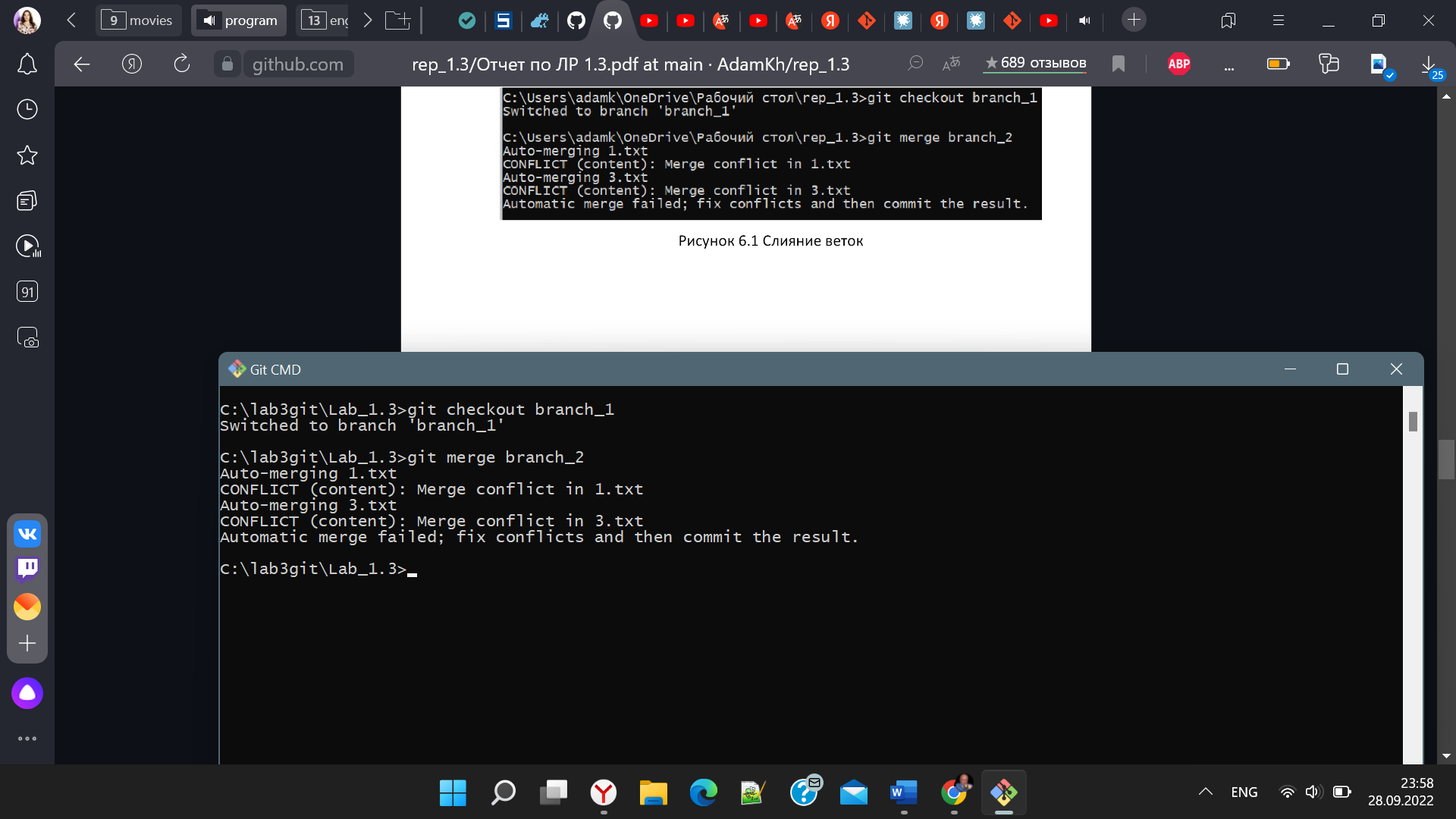


Рисунок 6.1 – Слияние веток

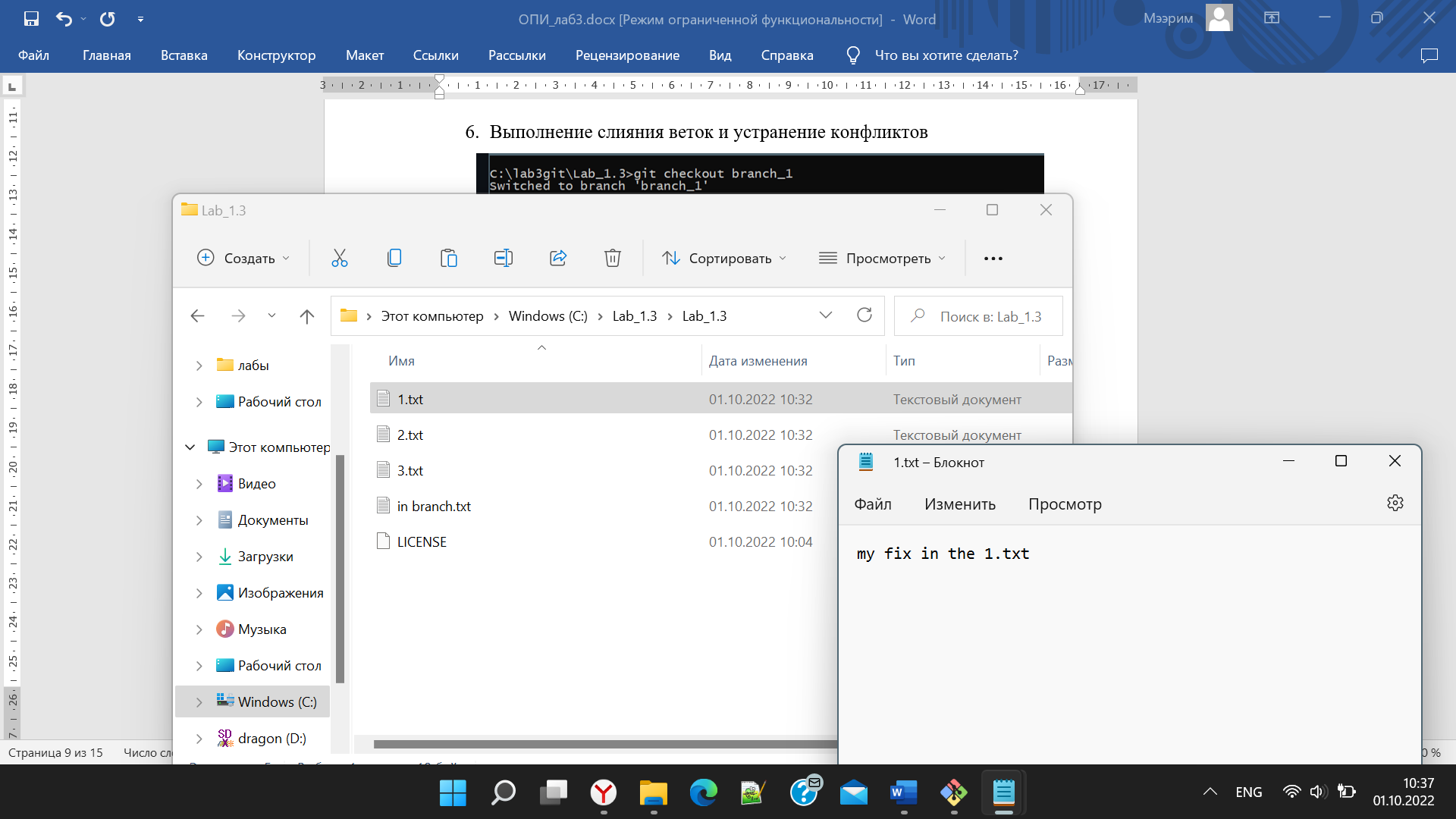


Рисунок 6.2 – Решение конфликта вручную

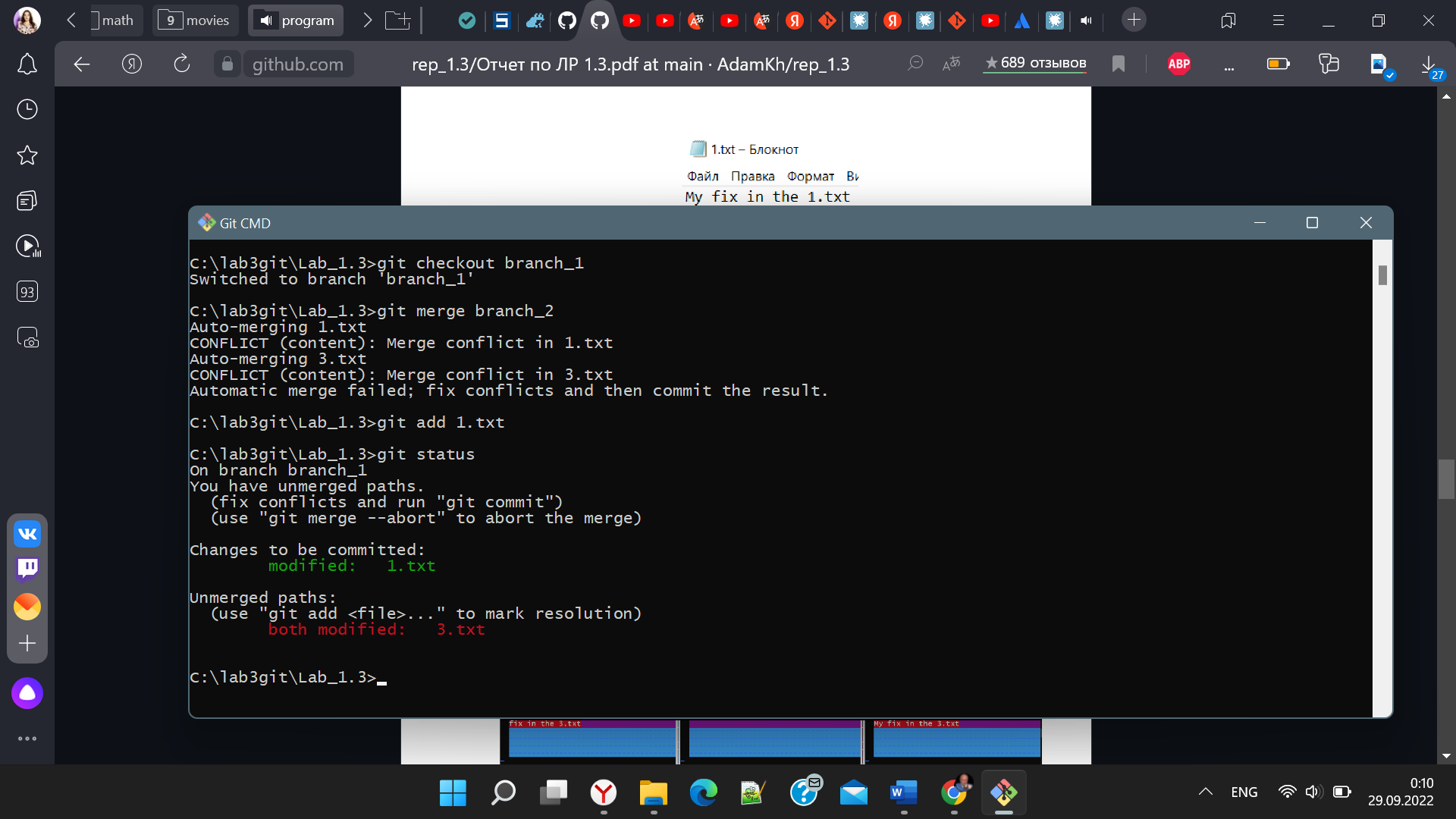


Рисунок 6.3 – Добавление индекса 1.txt для решения конфликта

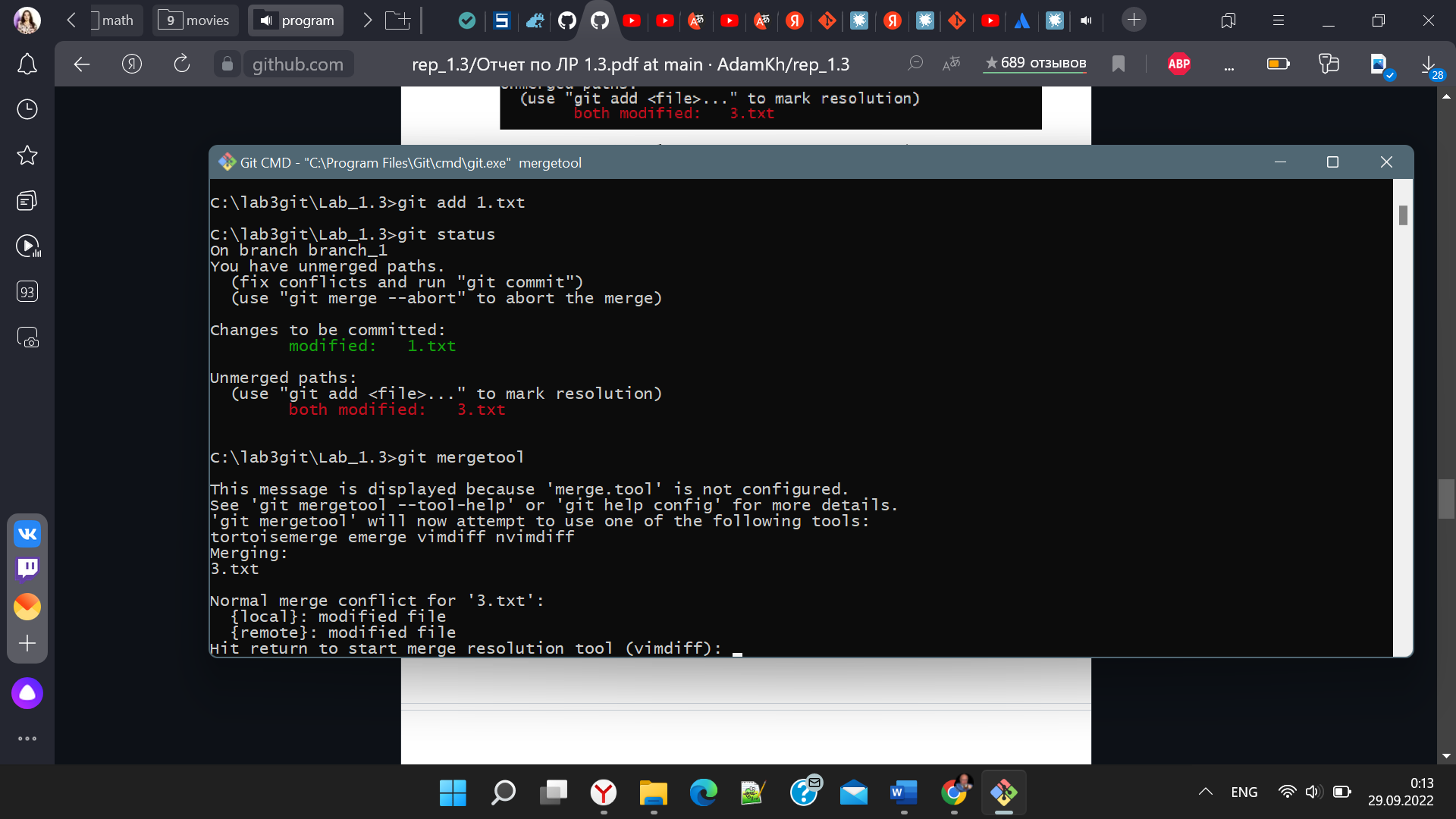


Рисунок 6.4 – Команда mergetool

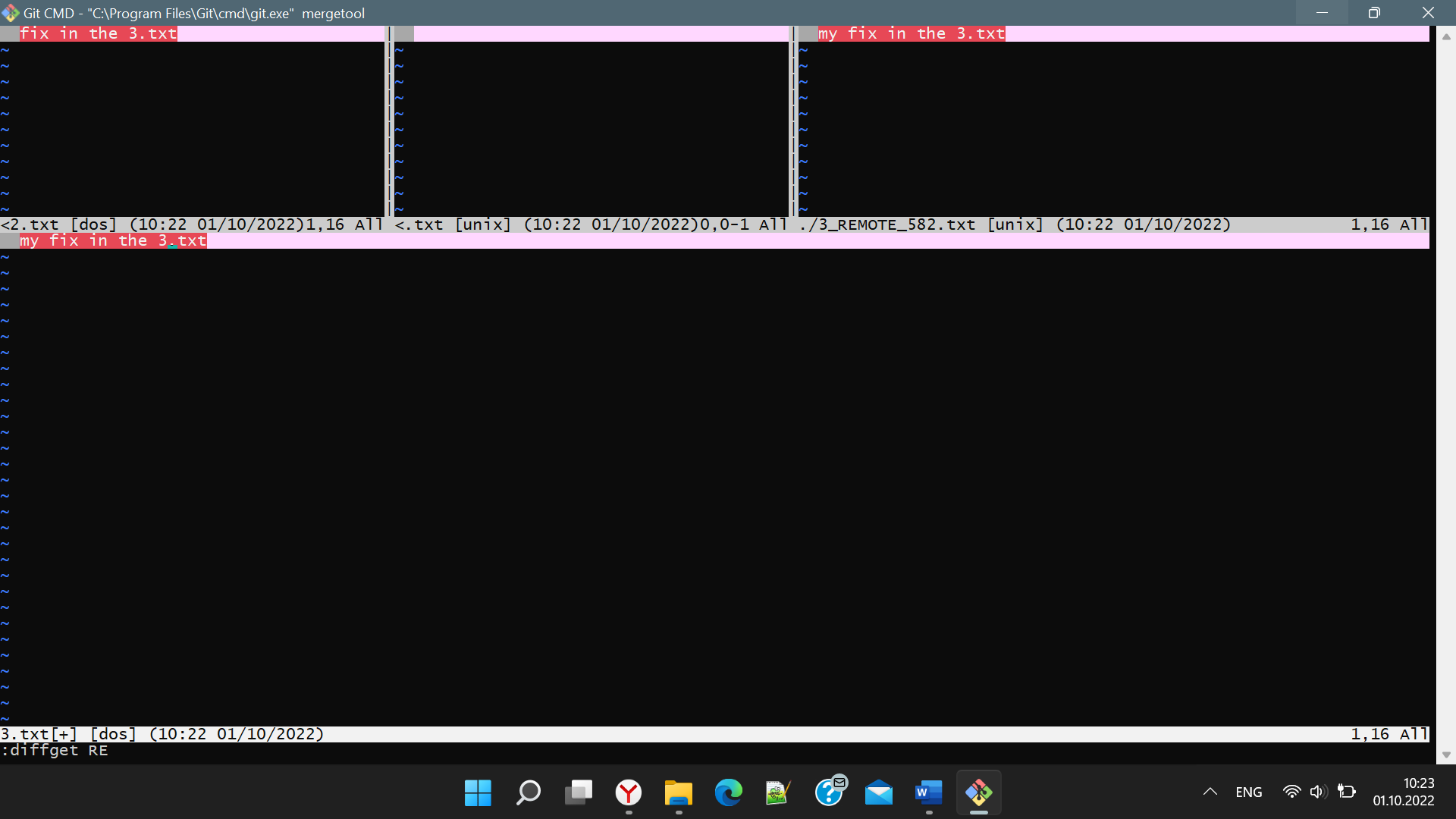


Рисунок 6.5 – Команда diffg RE

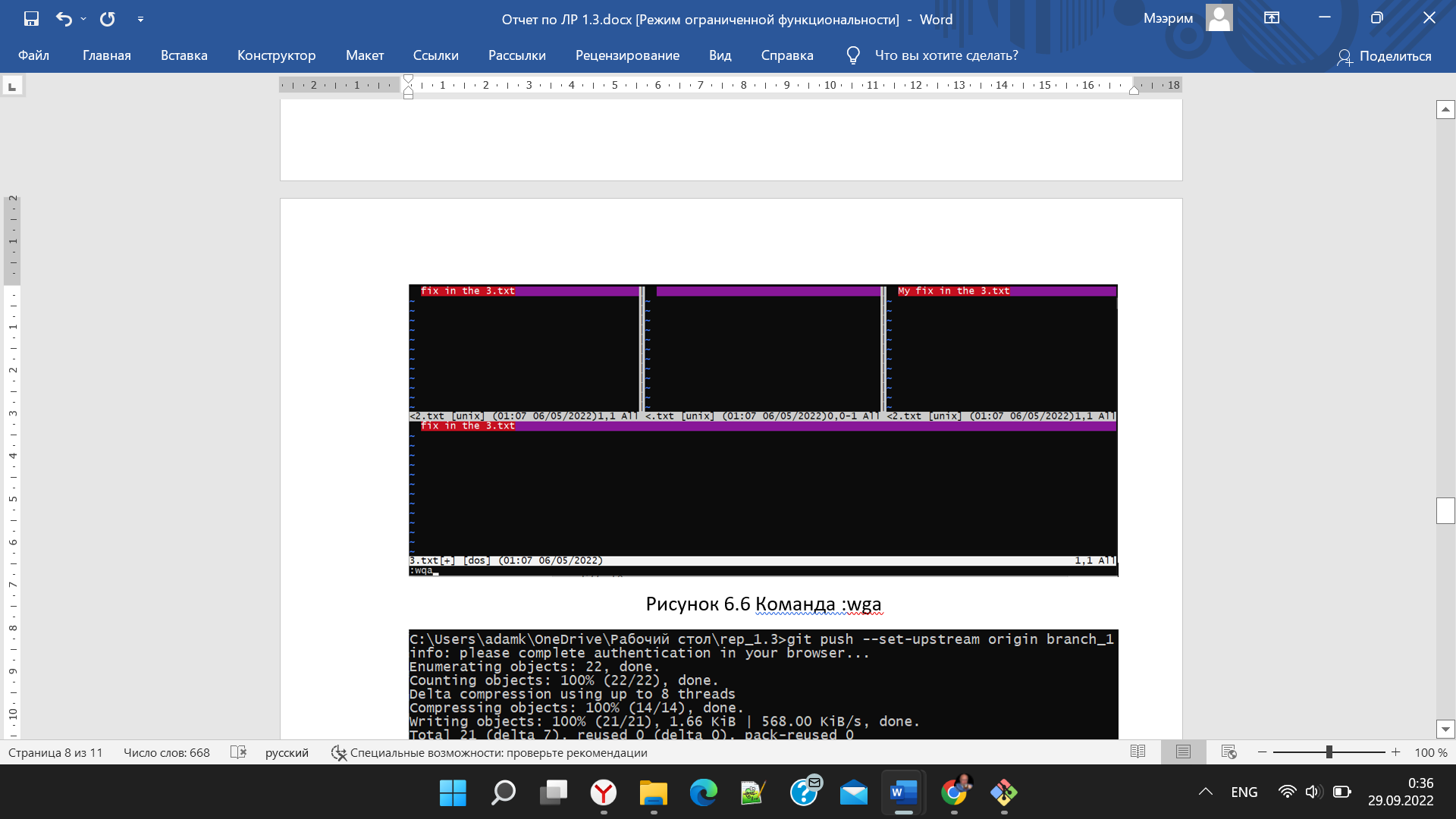


Рисунок 6.6 – Команда wga

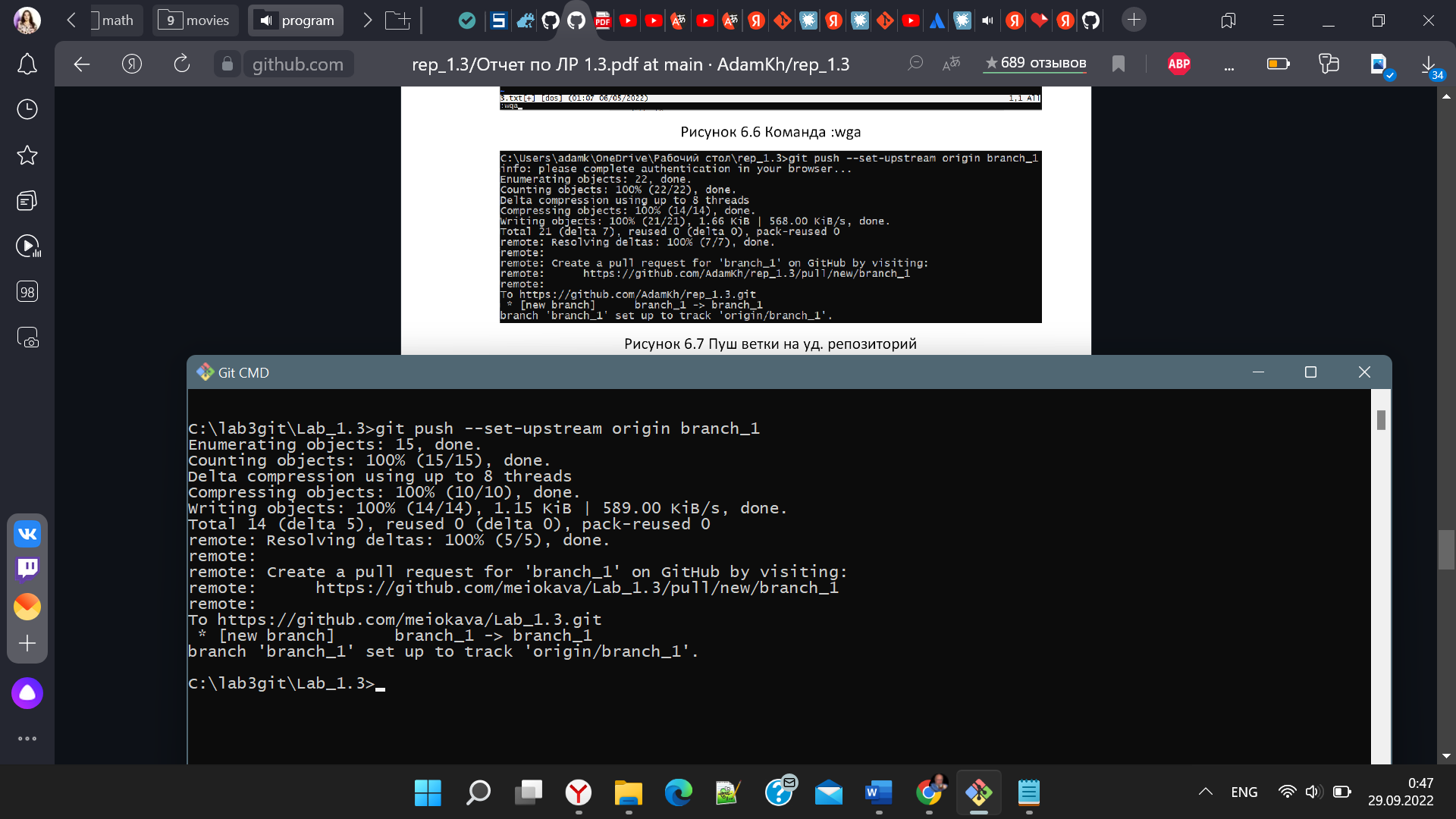


Рисунок 6.7 – Осуществление пуша ветки на удаленный сервер

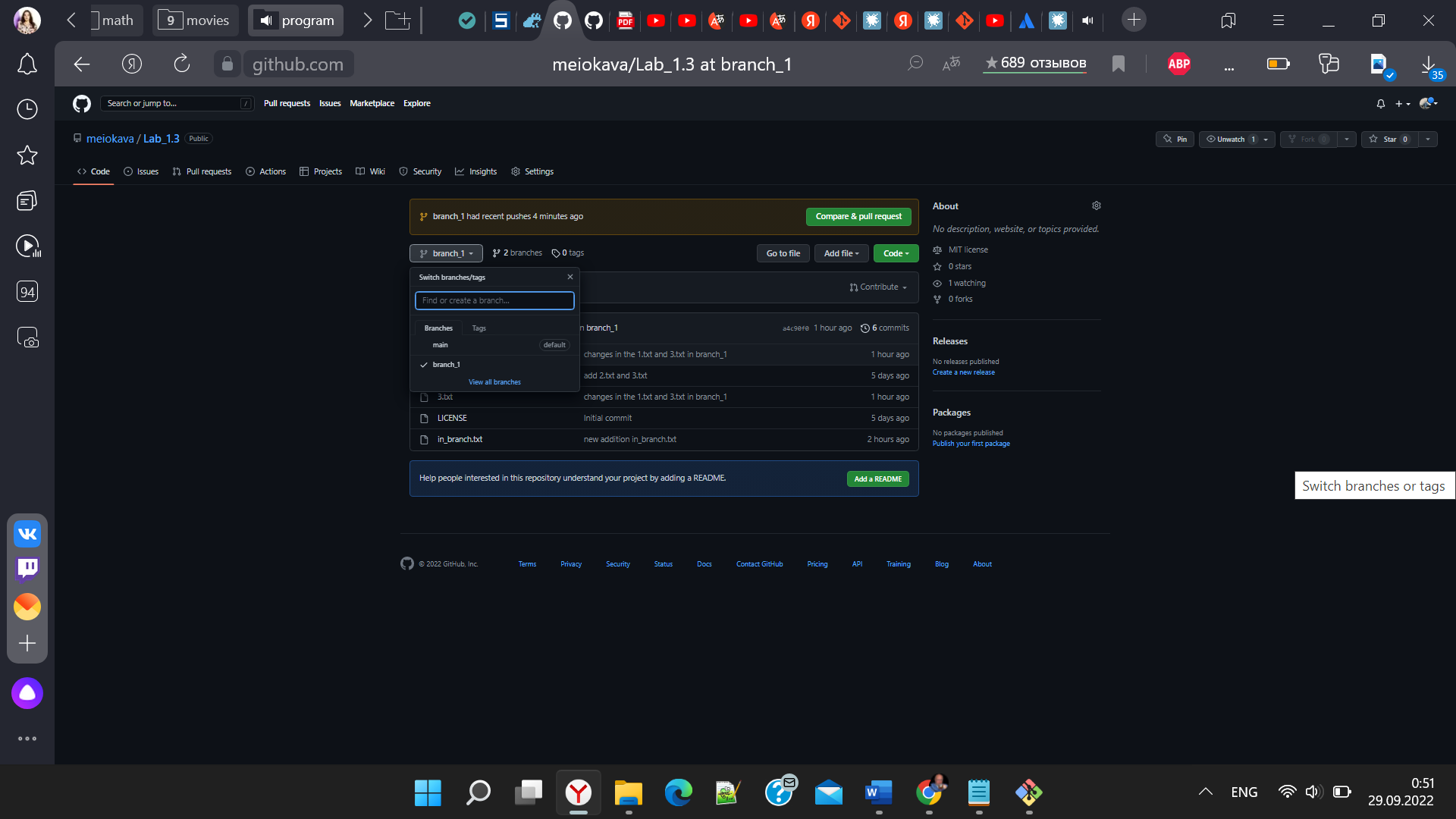


Рисунок 6.8 – Изменения на удаленном сервере

1. Было создано средствами GitHub удаленная ветка branch\_3, также была создана ветка в локальном репозитории для отслеживания удаленной ветки branch\_3. Был осуществлен переход на ветку branch\_3 и была добавлена строка t «he final fantasy in the 4.txt file»



Рисунок 7.1 – Создание удаленной ветки branch\_3

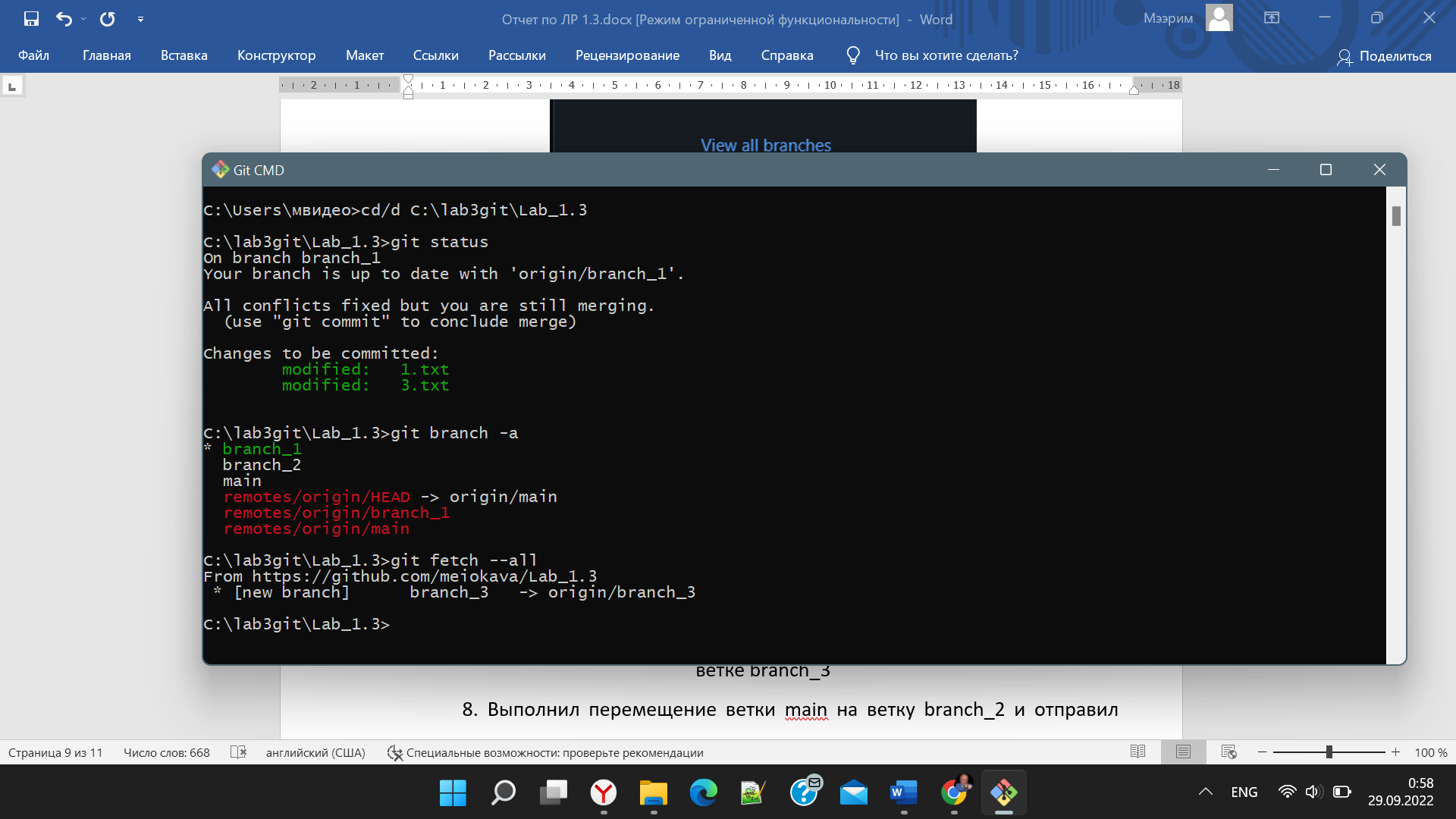


Рисунок 7.2 – Загрузка содержимого из удаленного репозитория перед отслеживанием

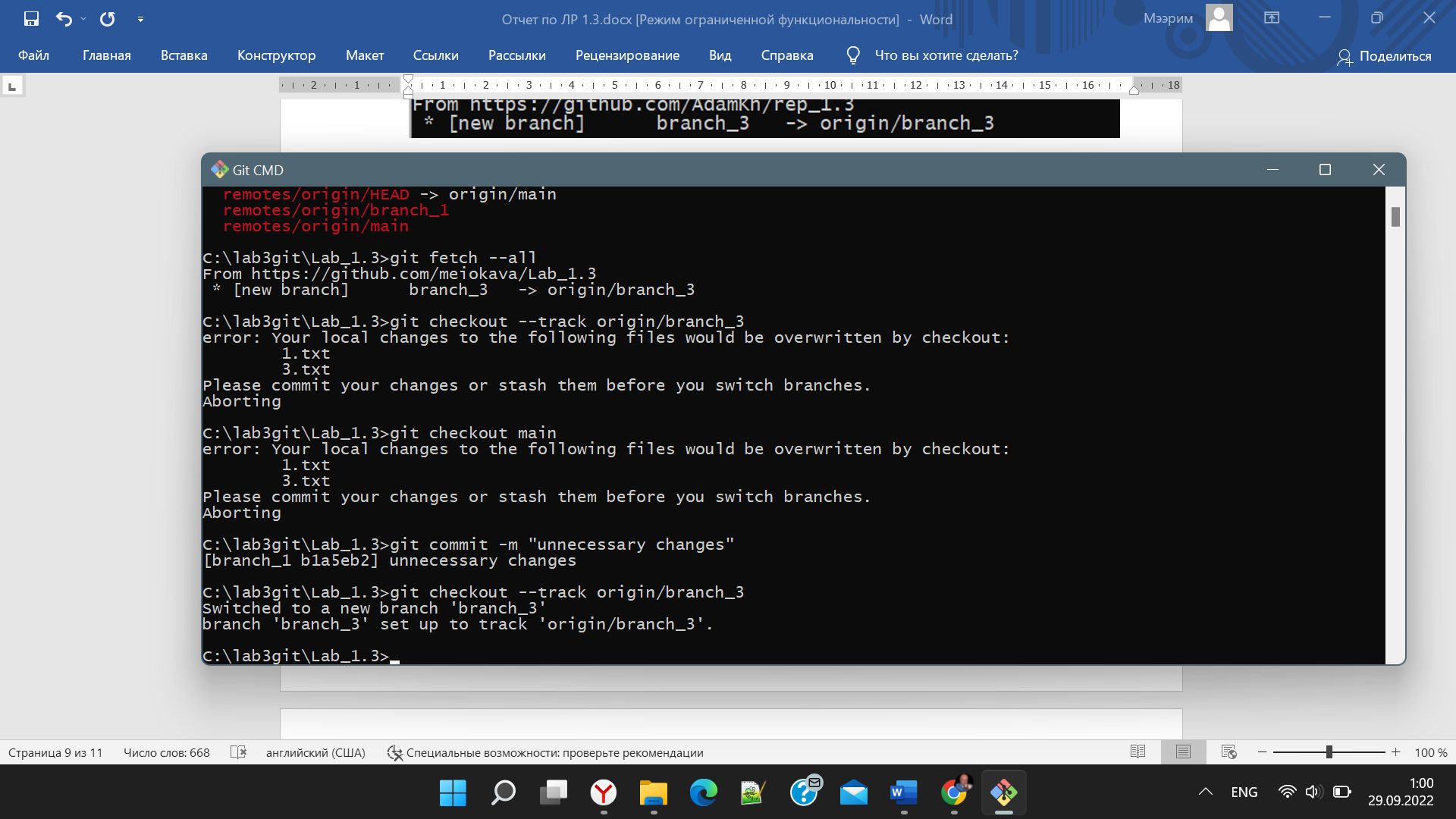


Рисунок 7.3 – Создание ветки для отслеживания branch\_3

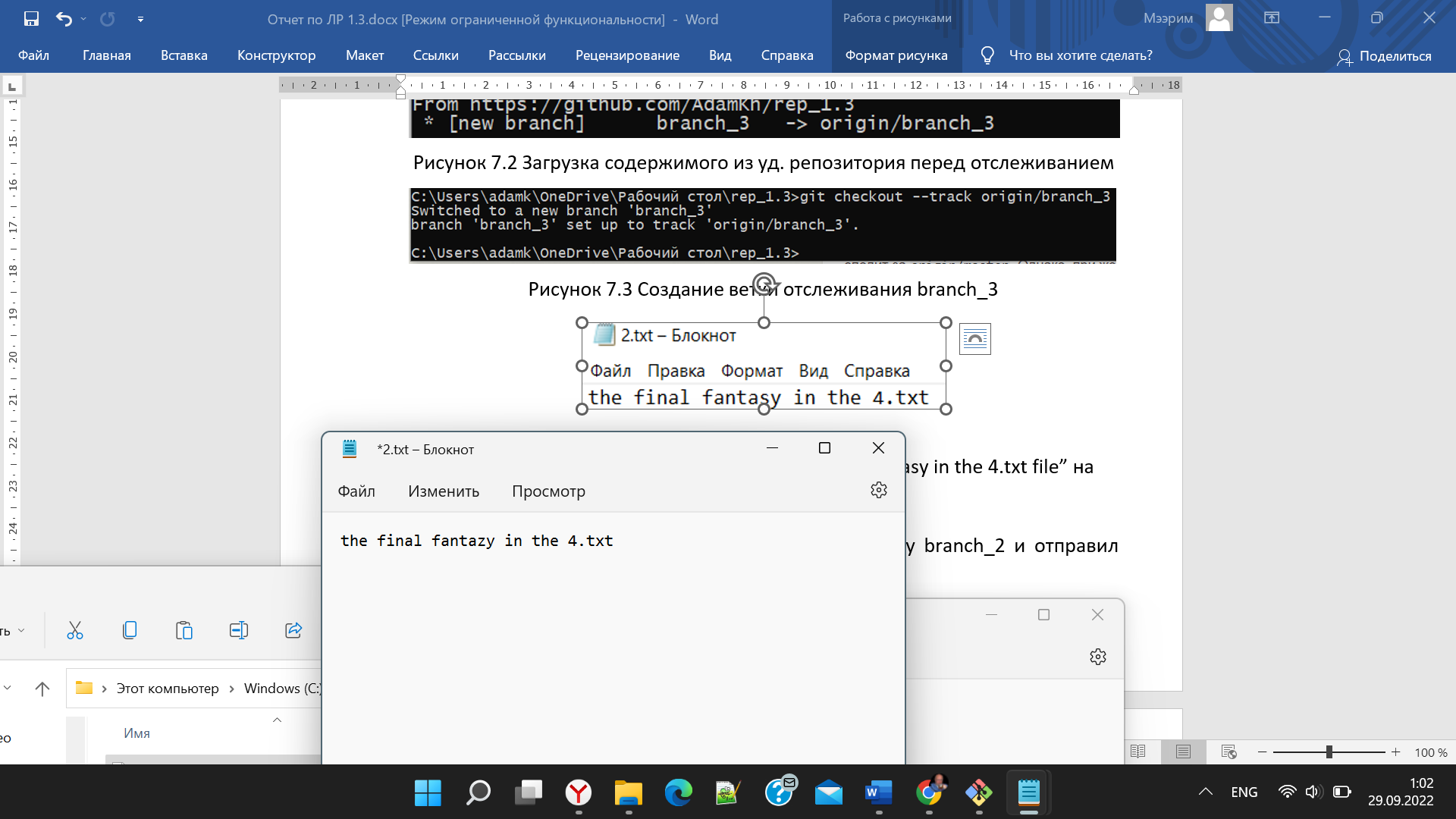


Рисунок 7.4 – Добавление новой строки в файл 2.txt

1. Было выполнено перемещение ветки main на ветку branch\_2, также изменения были отправлены изменения веток main и branch на GitHub

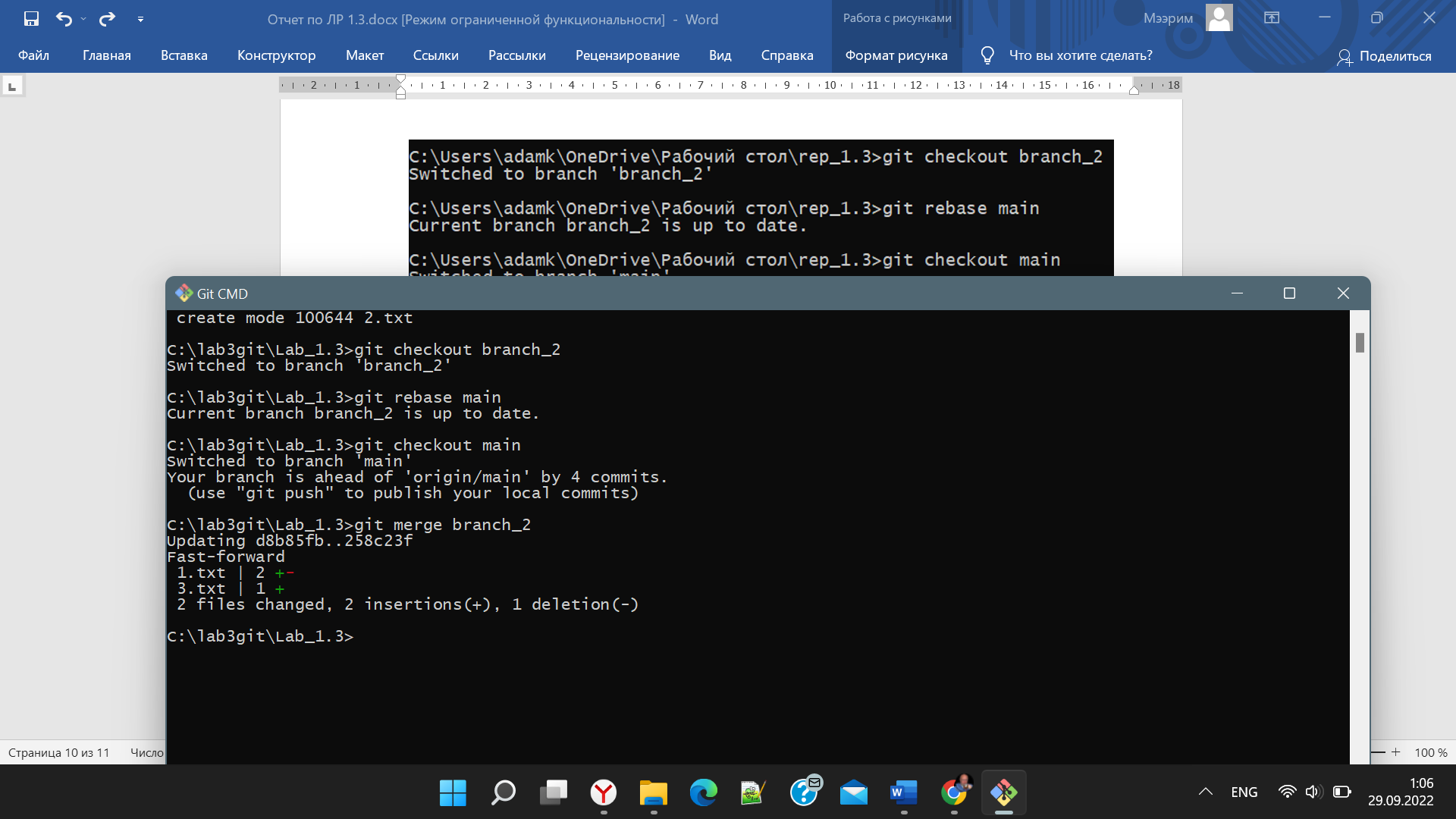


Рисунок 8.1 – Результат после выполнения команд

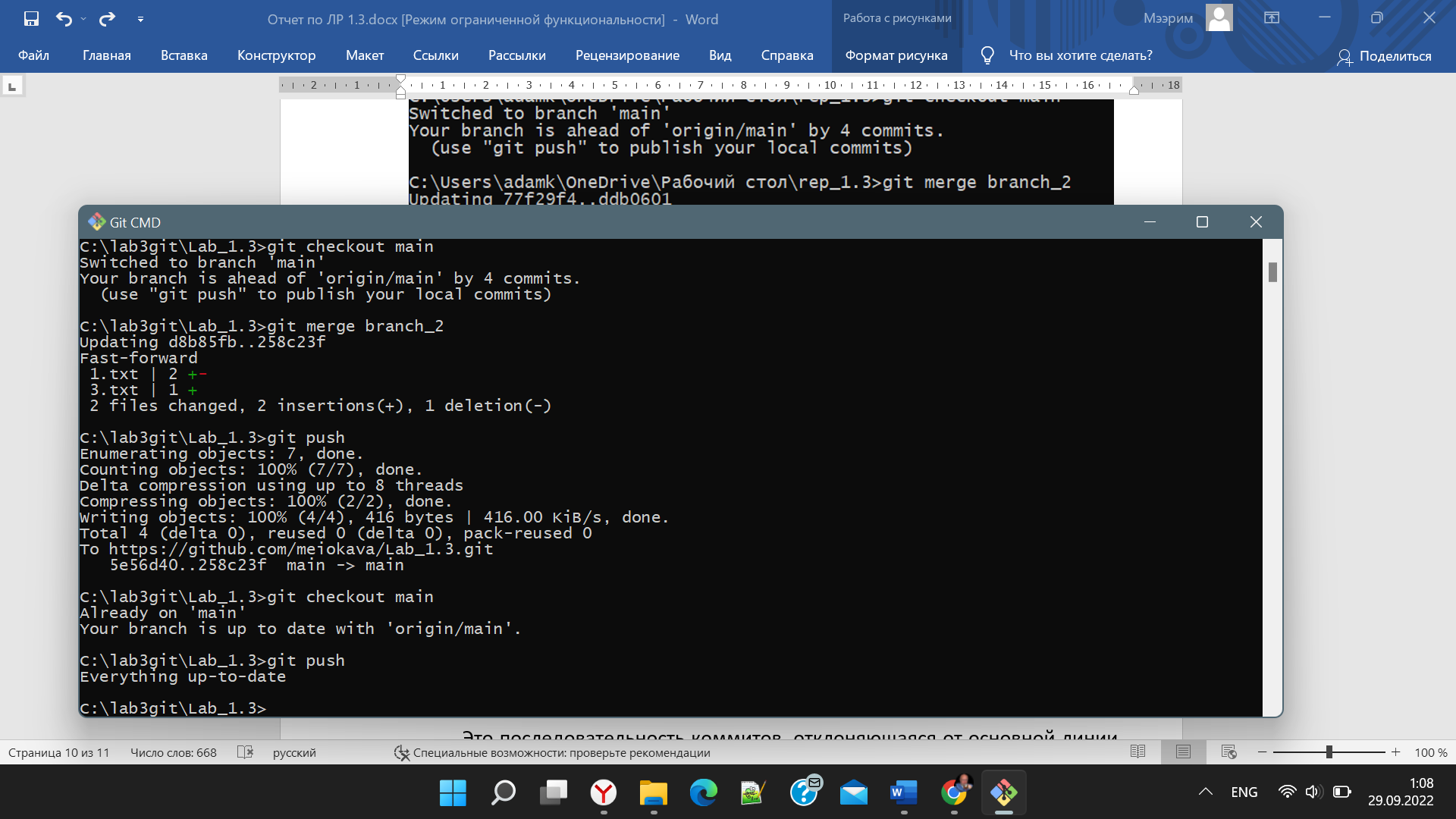


Рисунок 8.2 – Пуш веток

**Вывод:** были исследованы базовые возможности по работе с локальными и удаленными ветками GIT.

**Контрольные вопросы**

1. **Что такое ветка?**

В Git ветки — это элемент повседневного процесса разработки. По сути ветки в Git представляют собой указатель на снимок изменений.

1. **Что такое Head?**

HEAD — указатель на текущий коммит или на текущую ветку (то есть, в любом случае, на коммит). Указывает на родителя коммита, который будет создан следующим

1. **Способы создание веток**

Новую ветку можно создать командой git branch <название\_ветки> или на удаленном репозитории git hub.

1. **Как узнать текущую ветку?**

При помощи команды **git branch -a**

1. **Как переключаться между ветками?**

При помощи команды git checkout <Название ветки>

1. **Что такое удаленная ветка?**

Удалённые ветки — это ссылки на состояние веток в ваших удалённых репозиториях. Это локальные ветки, которые нельзя перемещать. Они двигаются автоматически всякий раз, когда вы осуществляете связь по сети. Удалённые ветки действуют как закладки для напоминания о том, где ветки в удалённых репозиториях находились во время последнего подключения к ним.

1. **Что такое ветка отслеживания?**

Проверка локальной ветви из удаленной ветви автоматически создает так называемую ветвь отслеживания. Ветви отслеживания-это локальные ветви, имеющие прямое отношение к удаленной ветви. Если вы находитесь в ветке отслеживания и набираете git push, Git автоматически знает, на какой сервер и ветку нажать.

1. **Как создать ветку отслеживания?**

Командой git checkout --track origin/<название\_ветки>.

1. **Как отправить изменения из локальной ветки в удаленную ветку?**

Командой git push origin <название\_ветки>.

1. **В чем отличие команд git fetch и git pull?**

Git pull – это сочетание команд git fetch (получение изменений с удаленного репозитория) и git merge (объединение веток).

1. **Как удалить локальную и удаленную ветки?**

Используя команду git branch -d <название\_ветки>. Для удаление удаленной ветки существует команда git push origin -d <название\_ветки>.

1. **Какие основные типы веток присуствуют в модели git-flow? Как организована работа с ветками в модели git-flow? В чем недостатки git-flow?**

В целом GitFlow состоит из двух постоянных веток и нескольких типов временных веток (В контексте GitHub Flow, master ветка — постоянная, а другие — временные).

В соответствии с этой моделью разработчики создают функциональную ветку и откладывают ее слияние с главной магистральной веткой до завершения работы над функцией. Такие долгосрочные функциональные ветки требуют тесного взаимодействия разработчиков при слиянии и создают повышенный риск отклонения от магистральной ветки.

Первая проблема: авторам приходится использовать ветку develop вместо master, поскольку master зарезервирован для кода, который отправляется в продакшен.

Вторая проблема процесса git flow – сложности, возникающие из-за веток для патчей и для релиза. Подобная структура может подойти некоторым организациям, но для абсолютного большинства она просто убийственно излишяя.