**Министерство цифрового развития связи и массовых коммуникаций**

**Ордена Труда Красного Знамени**

**Федеральное государственное бюджетное образовательное учреждение высшего образования**

**«Московский технический университет связи и информатики»**

Кафедра информатики

Лабораторная работа №1

«Основы работы с системой контроля версий»

Выполнил студент:

Дзускаева Алина Эдуардовна

БВТ 2205

Москва 2023

**Цель работы**

Цель данной лабораторной работы — изучить базовые функции системы контроля версий на примере Git.

**Ход лабораторной работы**

Я создала репозиторий через сайт GitHub, а затем клонировала его на свой компьютер. (Рисунок 1-2)

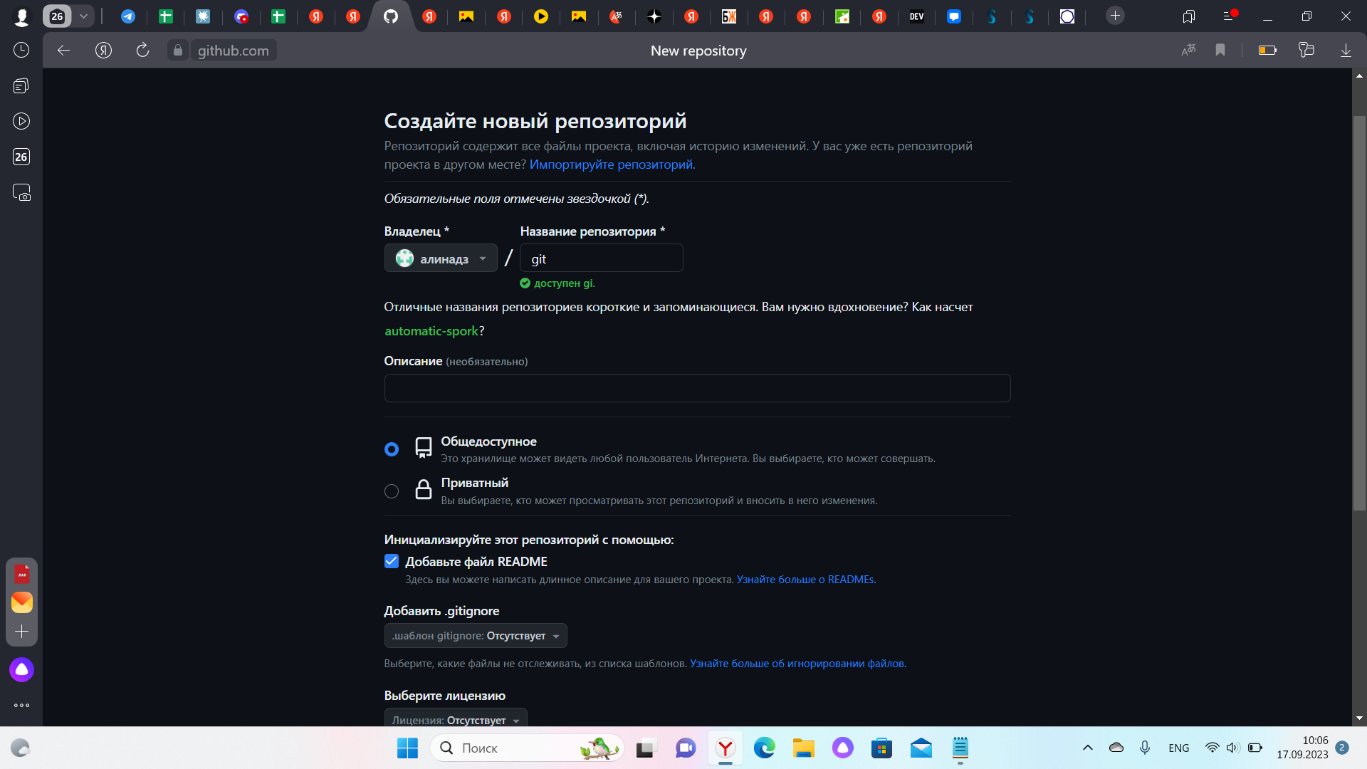


Рисунок 1 — Создание репозитория

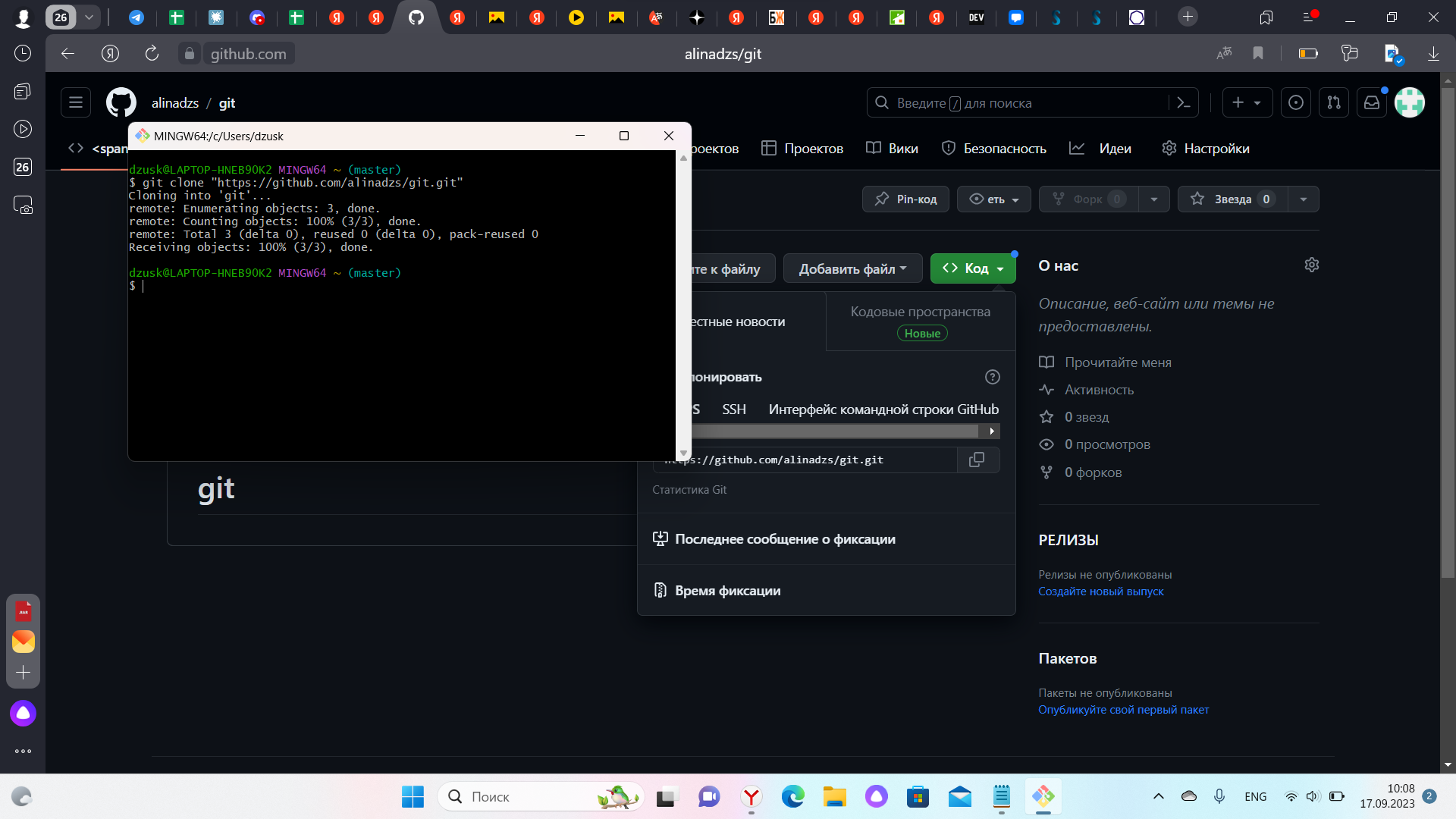


Рисунок 2 – Клонирование репозитория

На данный момент наша репрозиоорий пуст, поэтому добавим в нее текстовый файл «sss.txt». Содержимое текстового файла представлено на рисунке 3.

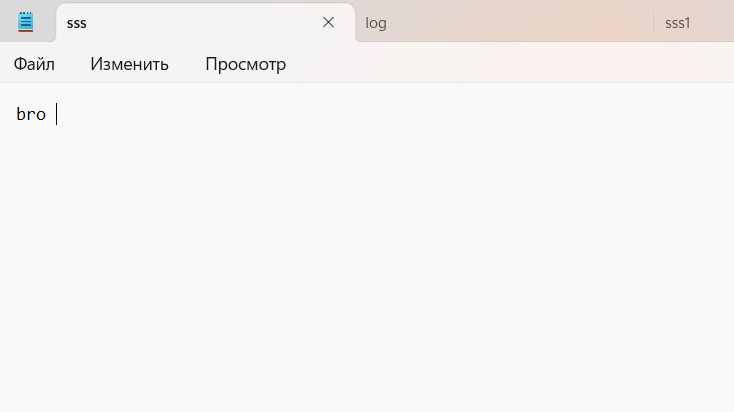


Рисунок 3 — Содержимое текстового файла

Проверить состояние проекта можно при помощи команды «git status». Чтобы добавить созданный текстовый файл в локальный репозиторий для начала необходимо его добавить в staging area при помощи команды «git add sss.txt». После этого производим коммит в локальный репозиторий при помощи команды «git commit -m "Добавлен файл"». На рисунке 4 представлена консоль и введенные в нее команды.

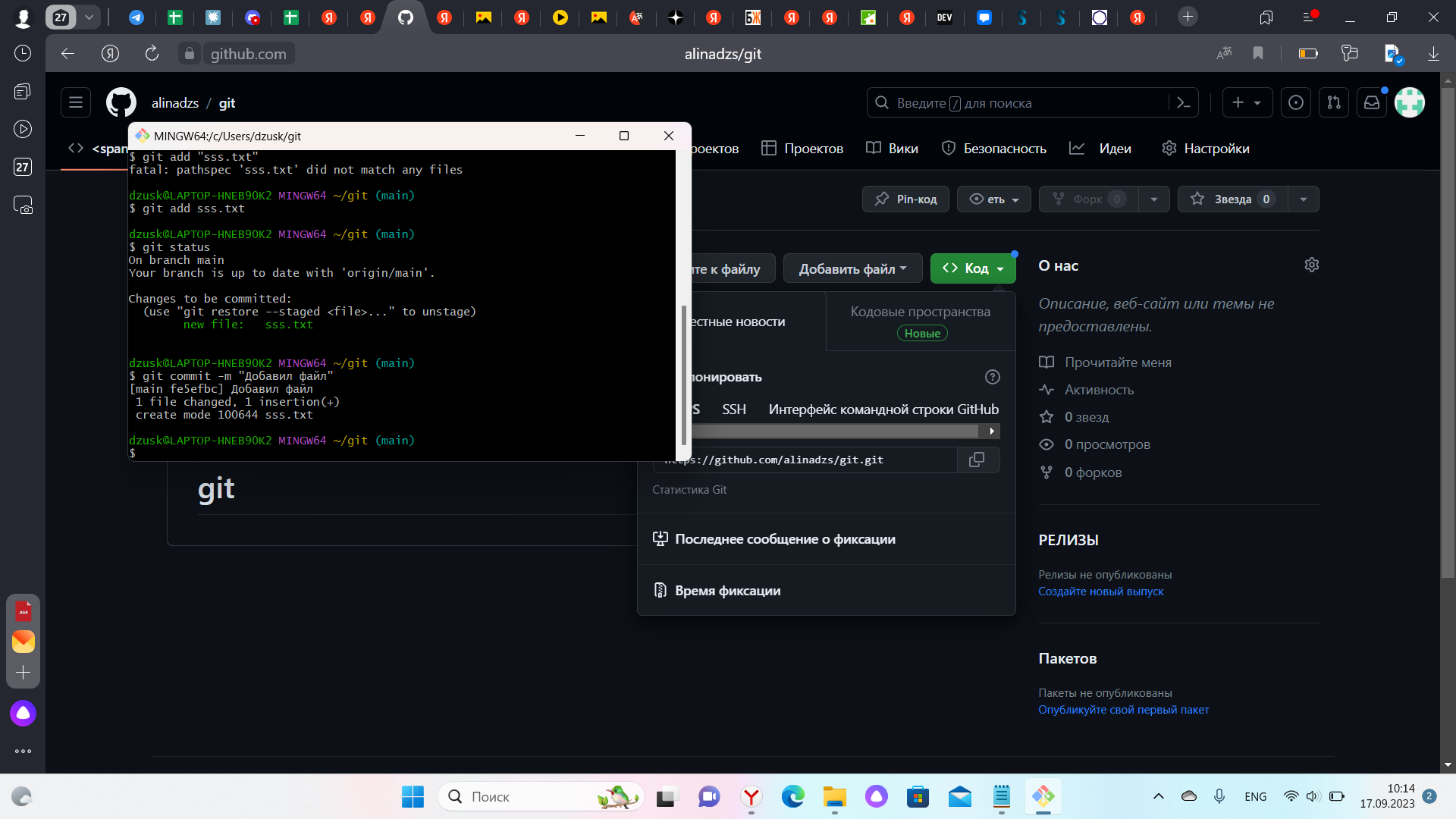


Рисунок 4 — Добавление текстового файла в репозиторий через консоль

Теперь добавим в ещё один текстовый файл под названием «sss1.txt». Его содержимое представлено на рисунок 5.

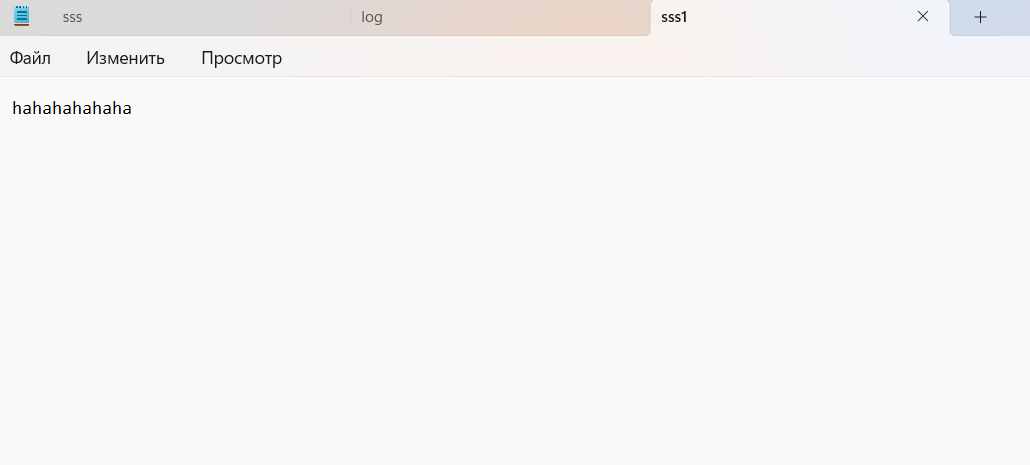


Рисунок 5 — Содержимое нового текстового файла

Вновь используя команды «git add sss1.txt» и «git commit -m "Добавлен второй файл"», добавляем второй текстовый файл в локальный репозиторий. Просмотреть историю коммитов можно при помощи команды «git log». На рисунке 6 представлено добавление второго текстового файла в репозиторий.

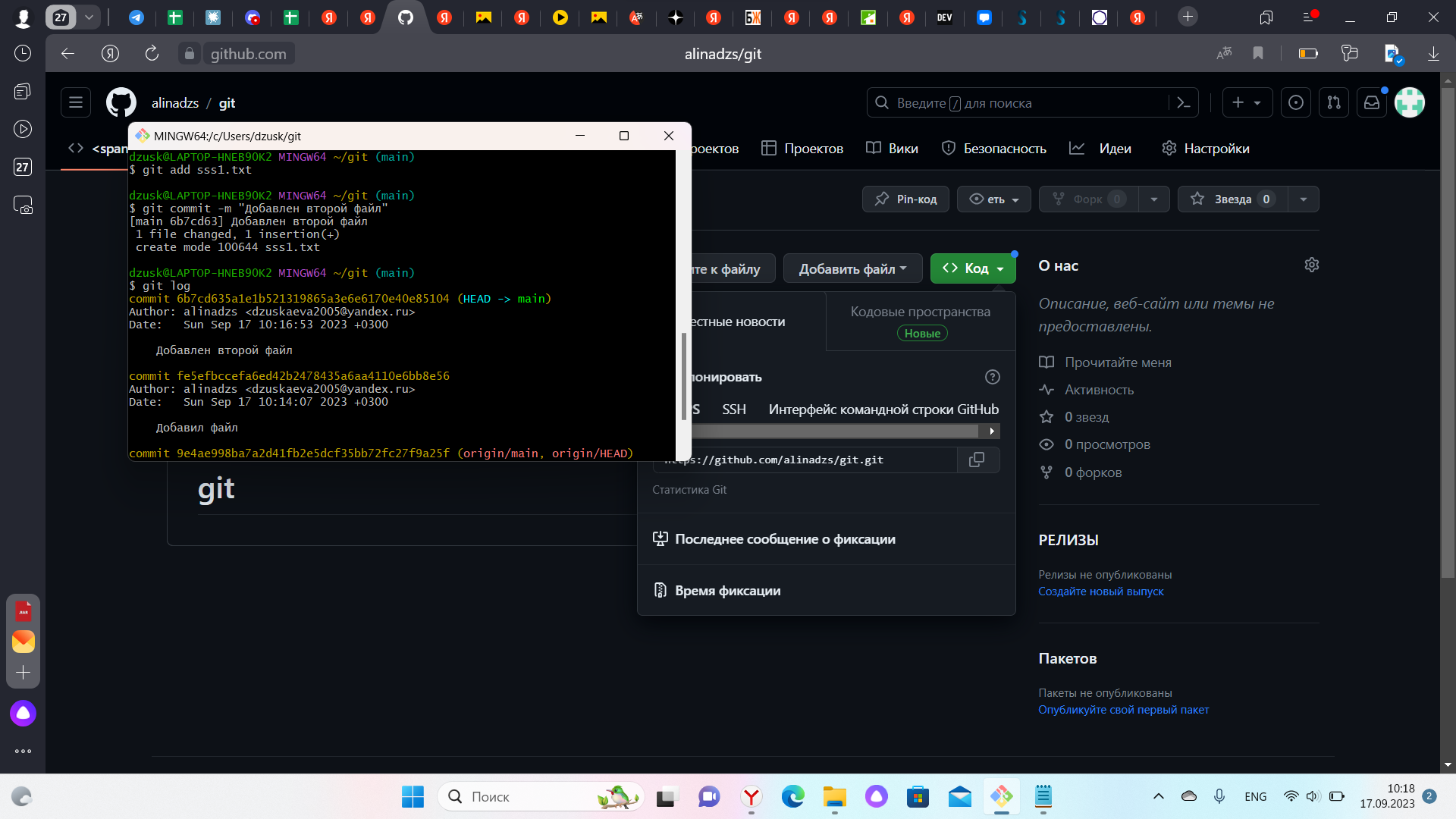


Рисунок 6 — Добавление второго текстового файла в репозиторий через консоль

Теперь создадим новую ветку под названием «Kanon» и сразу перейдем на нее при помощи команды «git checkout -b Kanon». Находясь в новой ветке, изменим содержимое текстового файла «sss.txt» и произведем коммит. После этого сделаем слияние ветки «Kanon» с веткой main. На рисунке 7 представлено создание новой ветки и взаимодействие с ней. На рисунке 8 представлено слияние ветки «Kanon» с веткой main.

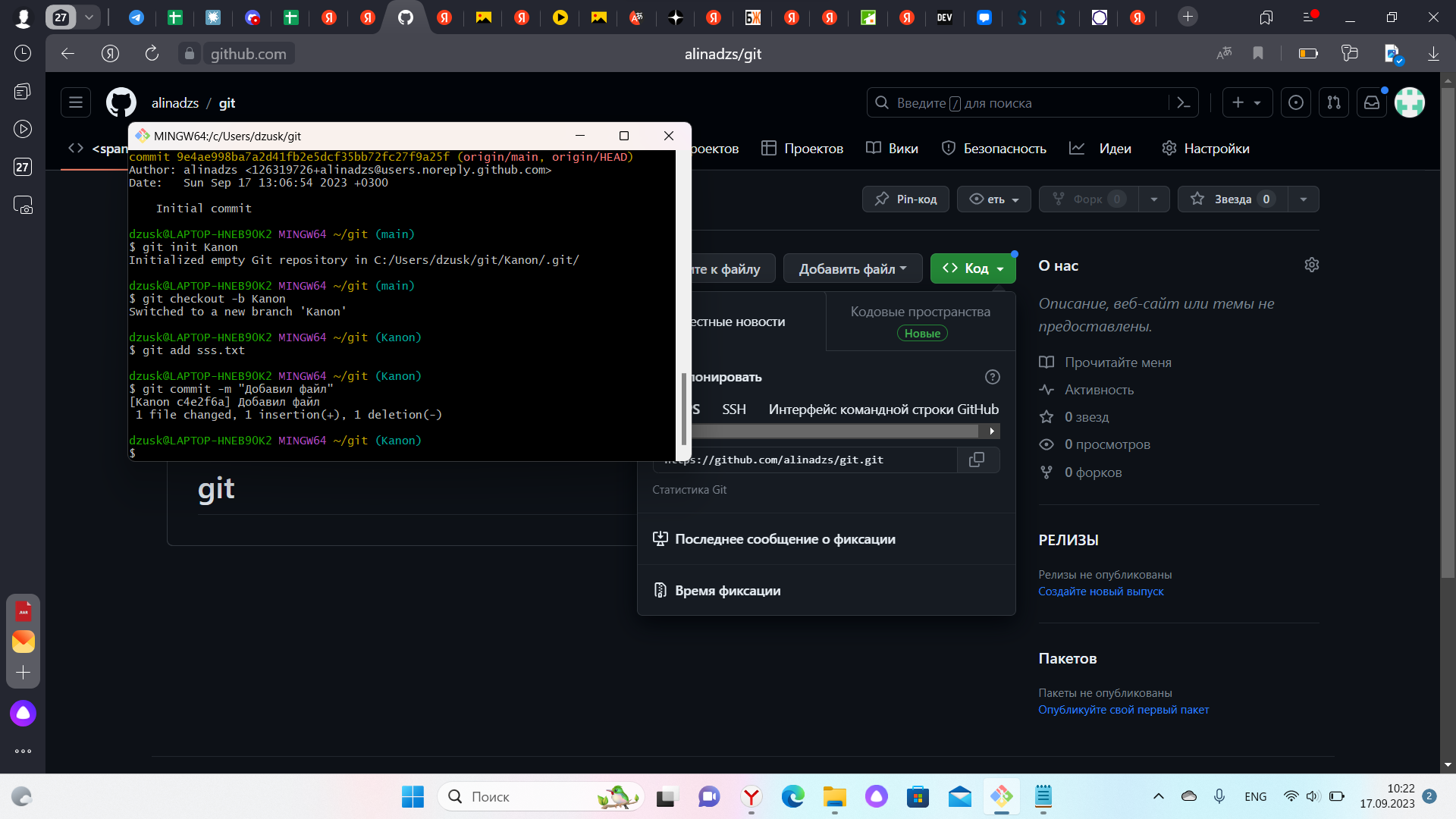


Рисунок 7 — Создание новой ветки

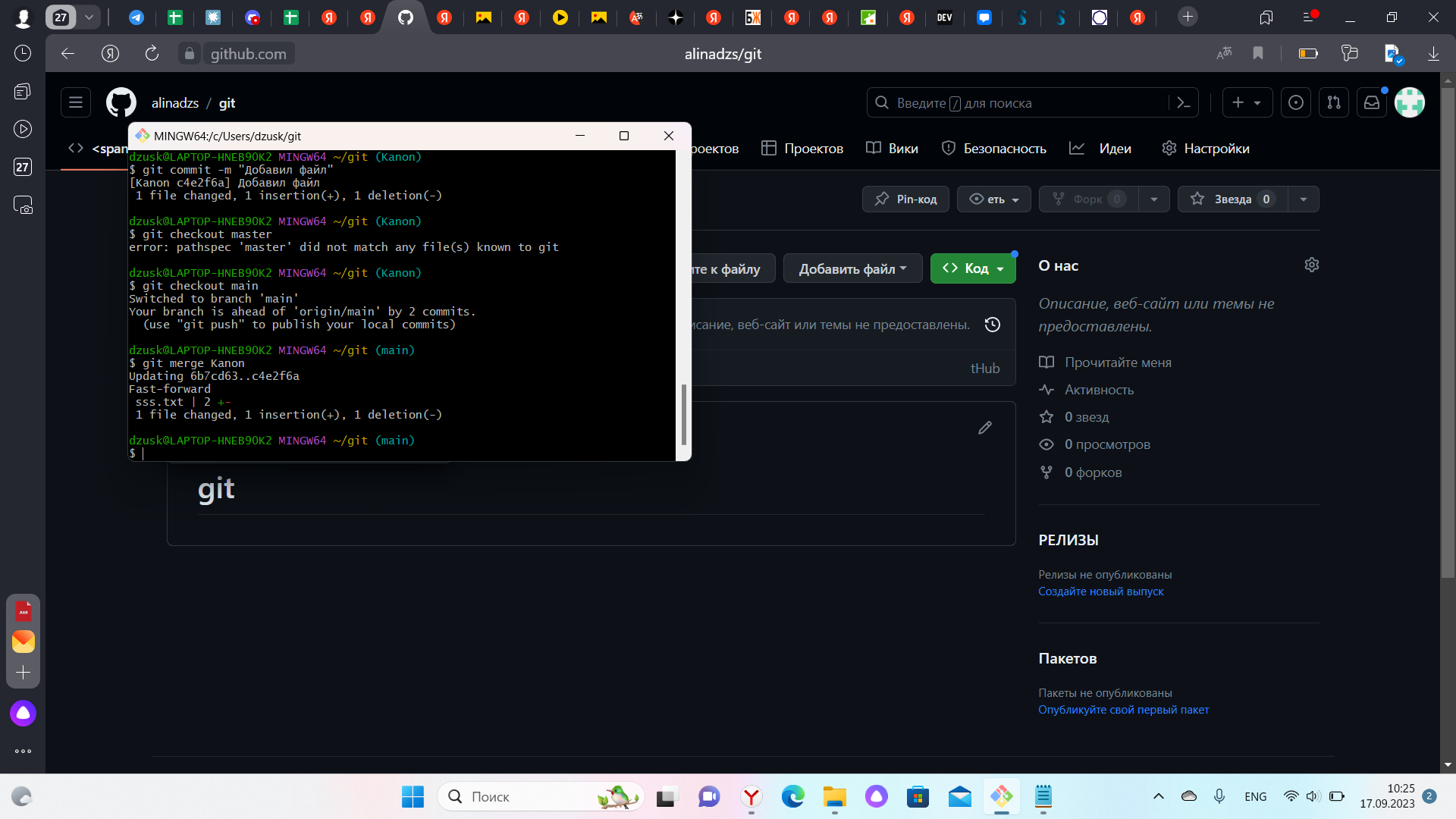


Рисунок 8 — Слияние веток

Предположим, что у нас возникла необходимость сделать откат до одного из предыдущих коммитов. Пускай это будет коммит, когда мы добавили первый текстовый файл. Сделать это можно при помощи команды «git reset —hard». Удостоверится в откате можно при вводе команды «git log». На рисунке 9 представлен откат до предыдущего коммита.

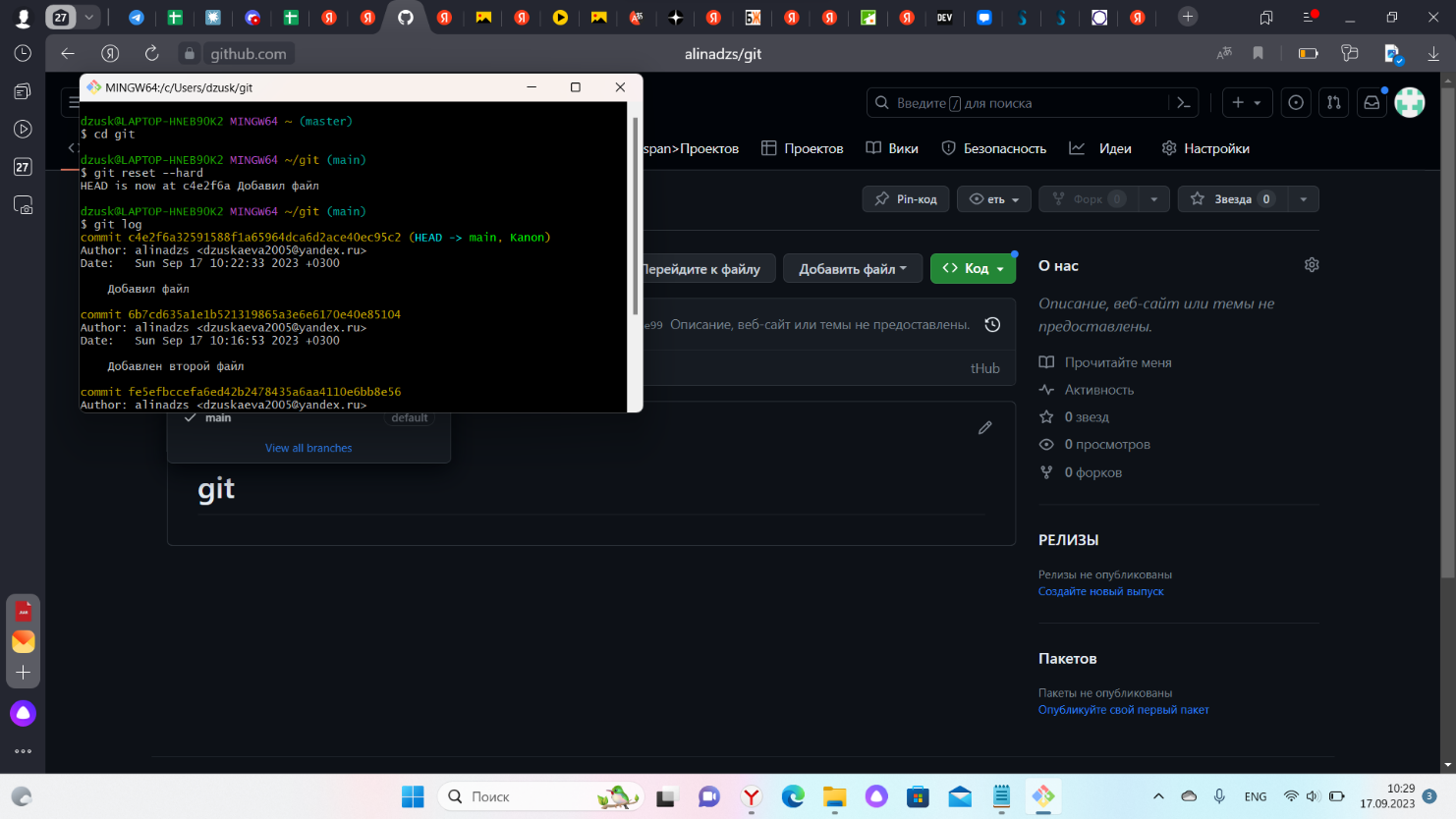


Рисунок 9 — Откат до предыдущего коммита

Создаем в новом репозитории игнорирование Git. Чтобы настроить игнорирование Git'ом файлов определенного расширения для начала необходимо создать файл «.gitignore». Процесс создания файла и добавление его в репозиторий представлен на рисунок 10.

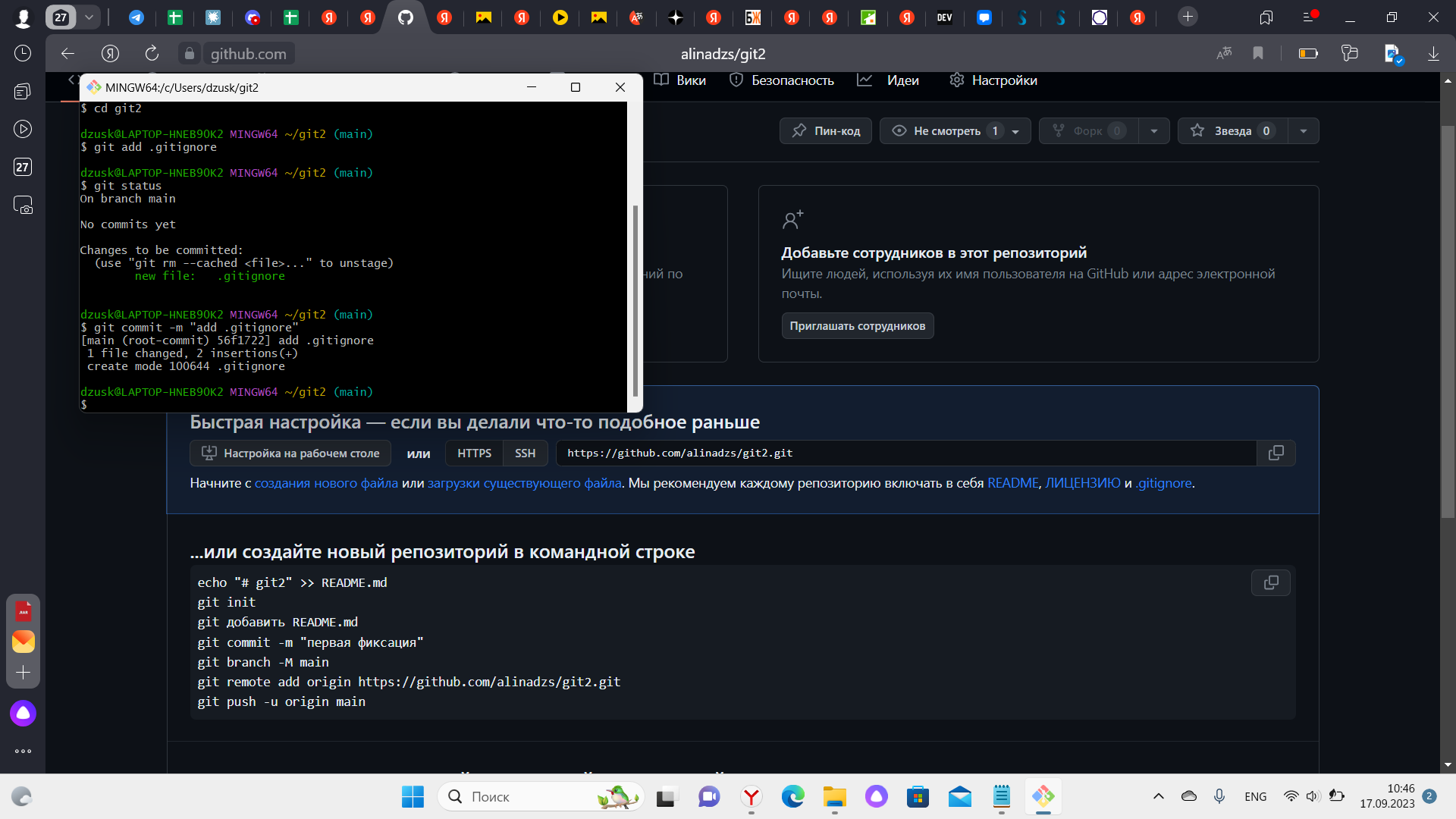


Рисунок 10 — Создание файла «.gitignore»

Внутри этого файла мы теперь можем указать, какой формат файлов будет игнорироваться Git'ом. Для примера возьмем файлы формата «.png» и файл с названием «IGT.txt». Теперь добавление или изменение файлов формата «.png» и изменение файла «IGT.txt» будут игнорироваться Git'ом. Содержимое файла «.gitignore» представлено на рисунке 11.

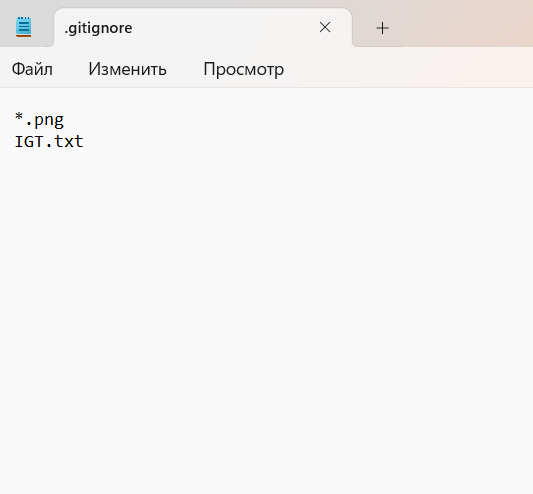


Рисунок 11 — Содержимое файла «.gitignore»

Вывести лог коммитов можно в текстовый файл при помощи команды «git log > log.txt». Содержимое файла «log.txt» представлено на рисунки 12-13.

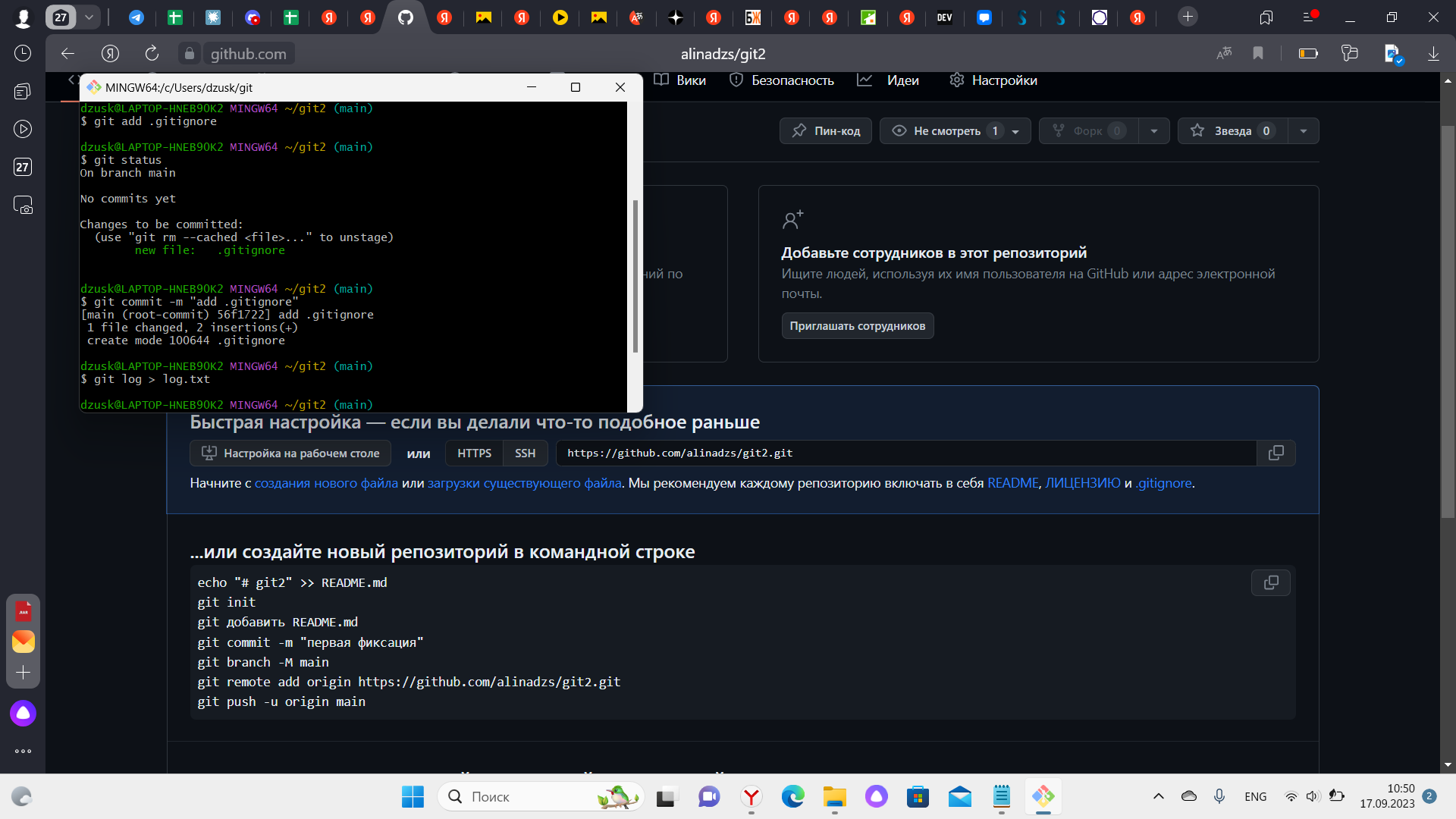


Рисунок 12 – Ввод лог коммита

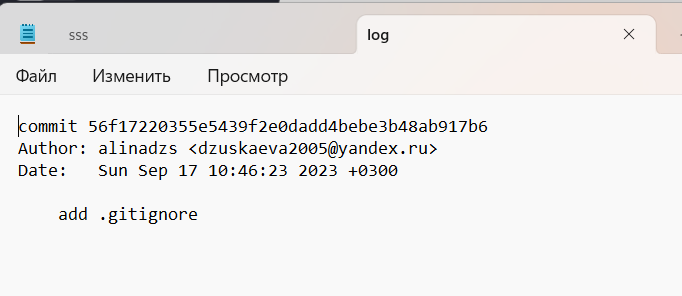


Рисунок 13 — Содержимое файла «log.txt»

Теперь попробуем переименовать созданную нами в прошлой лабораторной работе ветку «Kanon» в «TestKanon». Сделать это можно при помощи команды «git branch -m Kanon TestKanon». Теперь, перейдя в ветку «TestKanon», изменим содержимое файла «sss.txt», произведем коммит и сольем ветку с веткой main. Удалим слитую ветку при помощи команды «git branch -d TestKanon».

После этого создадим удаленный репозиторий под названием «stud» на сайте «[https://github.com](https://github.com/)». Мы можем связать локальный и удаленный репозитории при помощи команды «git remote add origin https://github.com/alinadzs/stud.git»,

после чего отправить содержимое локального репозитория на удаленный при помощи команды «git push -u origin main». Содержимое удаленного репозитория после произведенных операций представлено на рисунок 14.

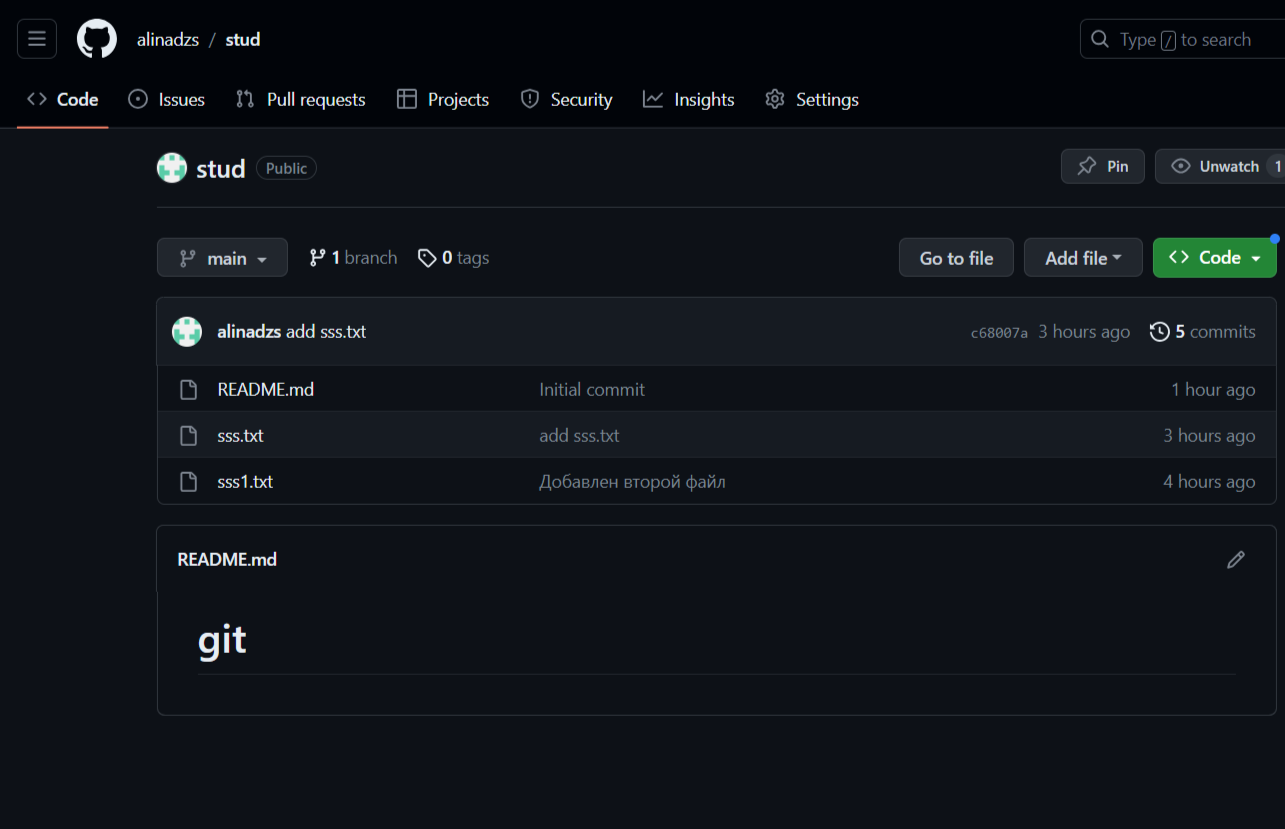


Рисунок 14 — Содержимое удаленного репозитория

Запрашиваем запрос на предоставления права к репозиторию друга. Далее заходим в GitBash и клонируем репозиторий. Создаем отдельную ветку в репозитории, куда мы будем пушить файлы с изменениями. Далее изменяем наши файлы, коммитим и пушим. Представлен на рисунке 15.

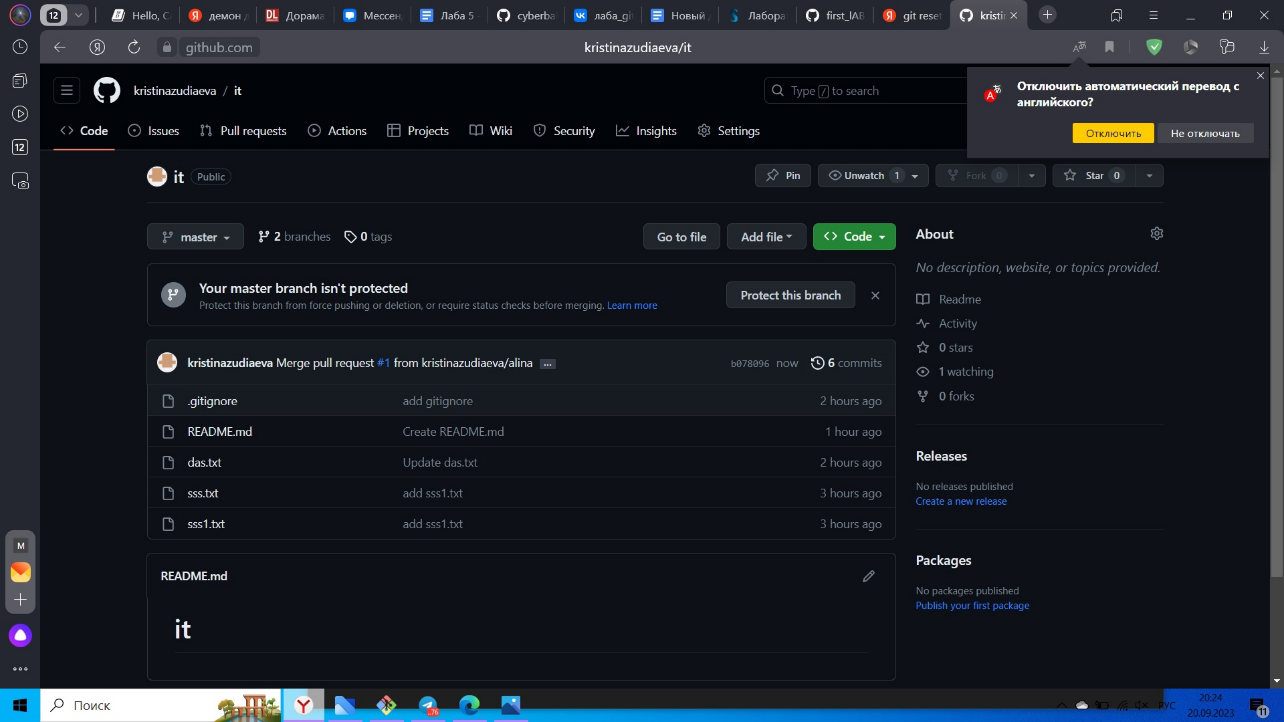


Рисунок 15 – Работа с чужим репозиторием

**Вывод**

Таким образом, на примере Git я научилась пользоваться базовыми функциями системы контроля версий и изучила подходы к командной разработке в системе контроля версий Git.