

**МИНОБРНАУКИ РОССИИ**

**федеральное государственное бюджетное образовательное учреждение**

**высшего образования**

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

**(ФГБОУ ВО «МГТУ «СТАНКИН»)**

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

Дисциплина «Основы системного программного обеспечения»

**Отчет по лабораторной работе №1**

На тему: «Работа с системами контроля версий на примере Git Hub»

Направление: 15.03.04 «Автоматизация технологических процессов и производств»

|  |  |  |
| --- | --- | --- |
| **Выполнил**  **студент гр. АДБ-20-06:** | \_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_  *(дата) (подпись)* | **Гринкевич В.С.** |
| **Проверил**  **к.т.н., доцент** | \_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_  *(дата) (подпись)* | **Ковалев И.А.** |

**Москва 2023 г.**

**Оглавление**

[Цель работы: 3](#_Toc103968943)

[Основные задачи работы: 3](#_Toc103968944)

[Ход работы: 4](#_Toc103968945)

[Вывод: 12](#_Toc103968946)

# **Цель работы:**

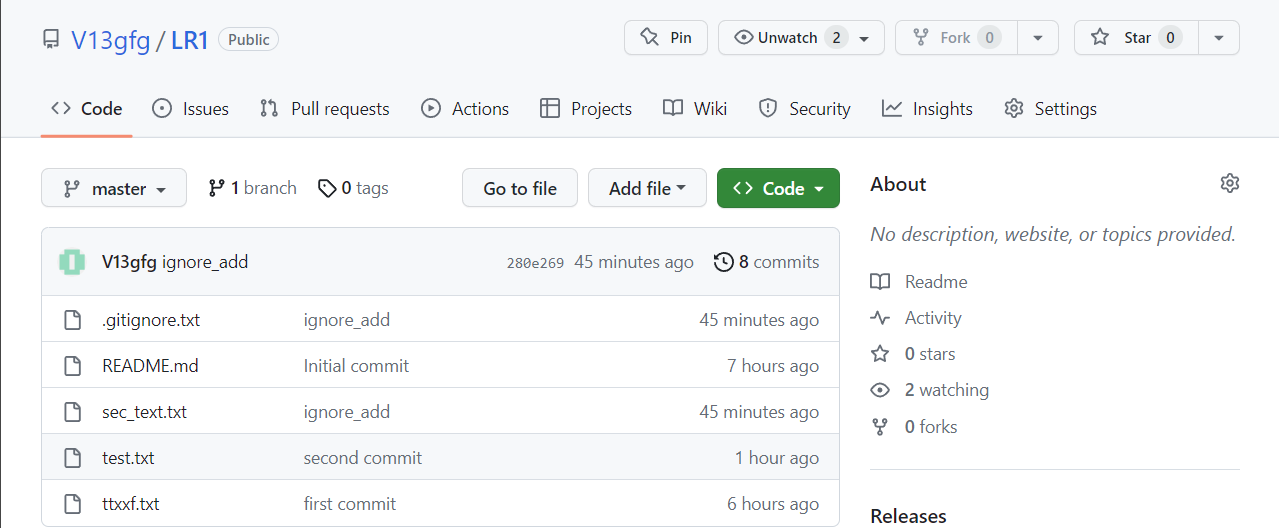
работа с системами контроля версий на примере Git Hub

## **Основные задачи работы:**

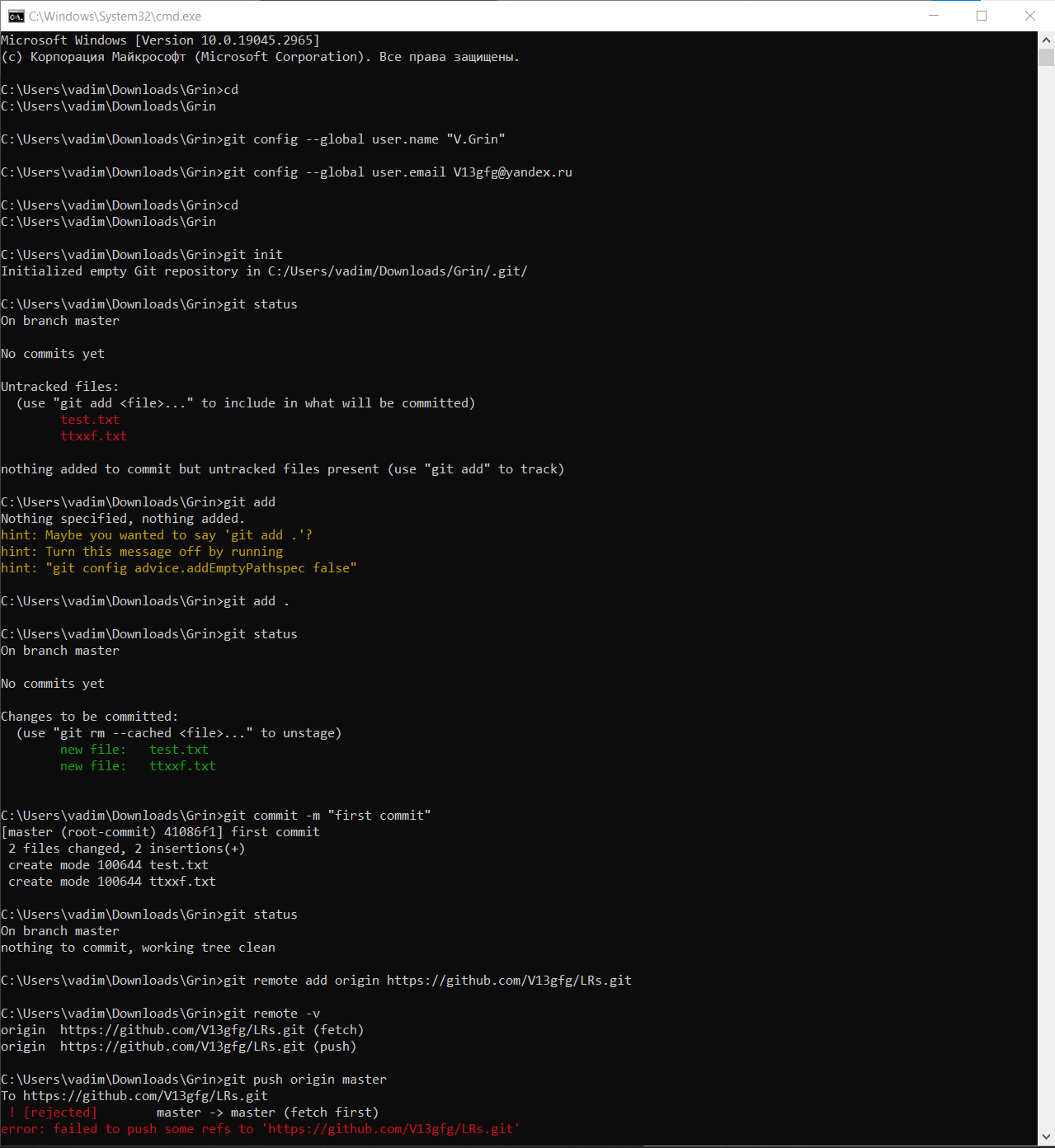
1. Создать учетную запись на github.com
2. Создать локальный репозиторий
3. Зафиксировать изменения в области заготовленных файлов
4. Переслать локальный коммит на сервер
5. Сделать слияние веток
6. Просмотреть изменения и разрешение конфликтов
7. Удалить ветку на сервере и вернуть к предыдущему состоянию
8. Исправить коммит
9. Совместная работа с git

## **Ход работы:**

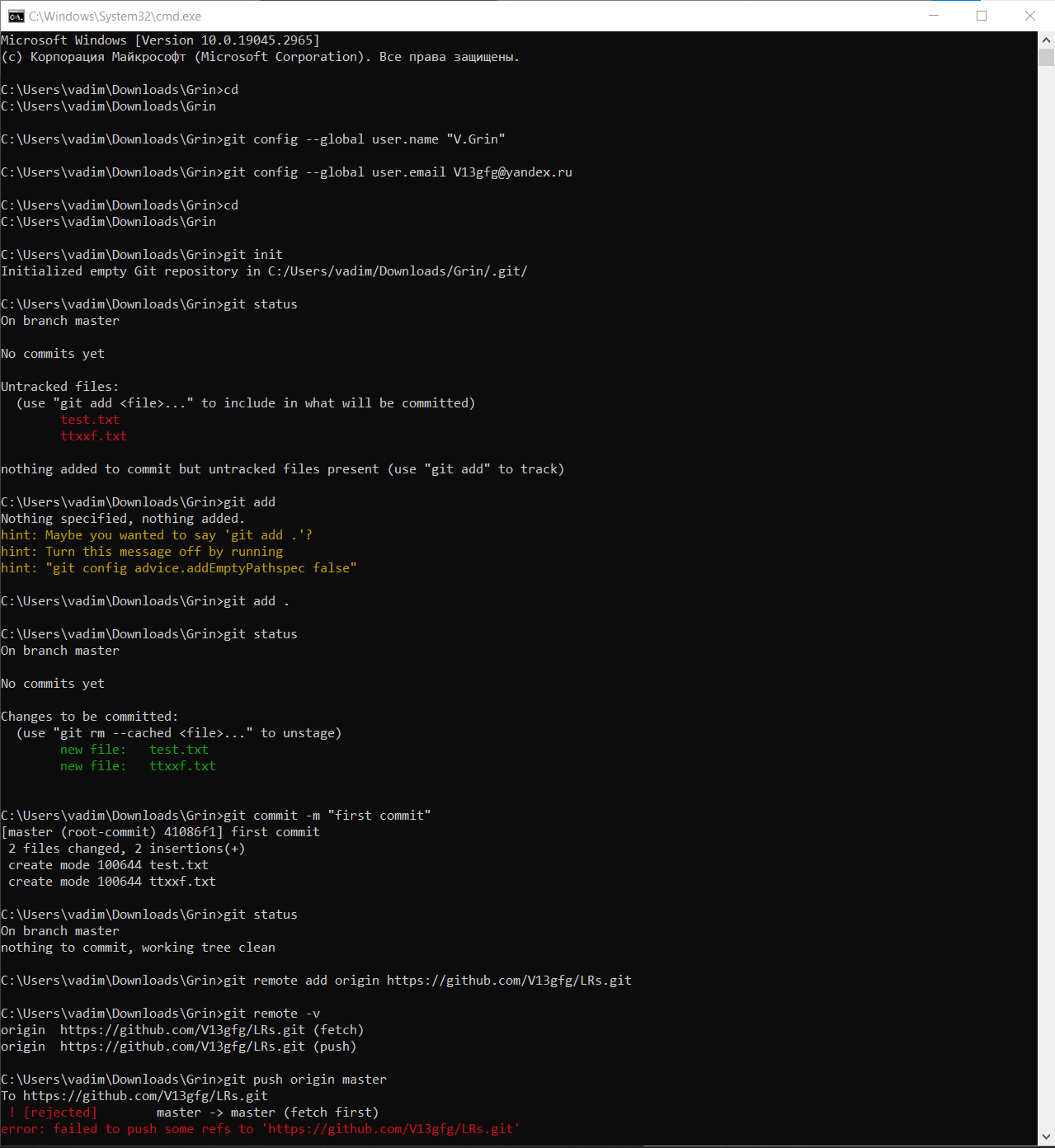
1. Создаем учетную запись на github.com или заходим в уже имеющийся личный кабинет, как в нашем случае и создаем новый репозиторий с именем V13gfg

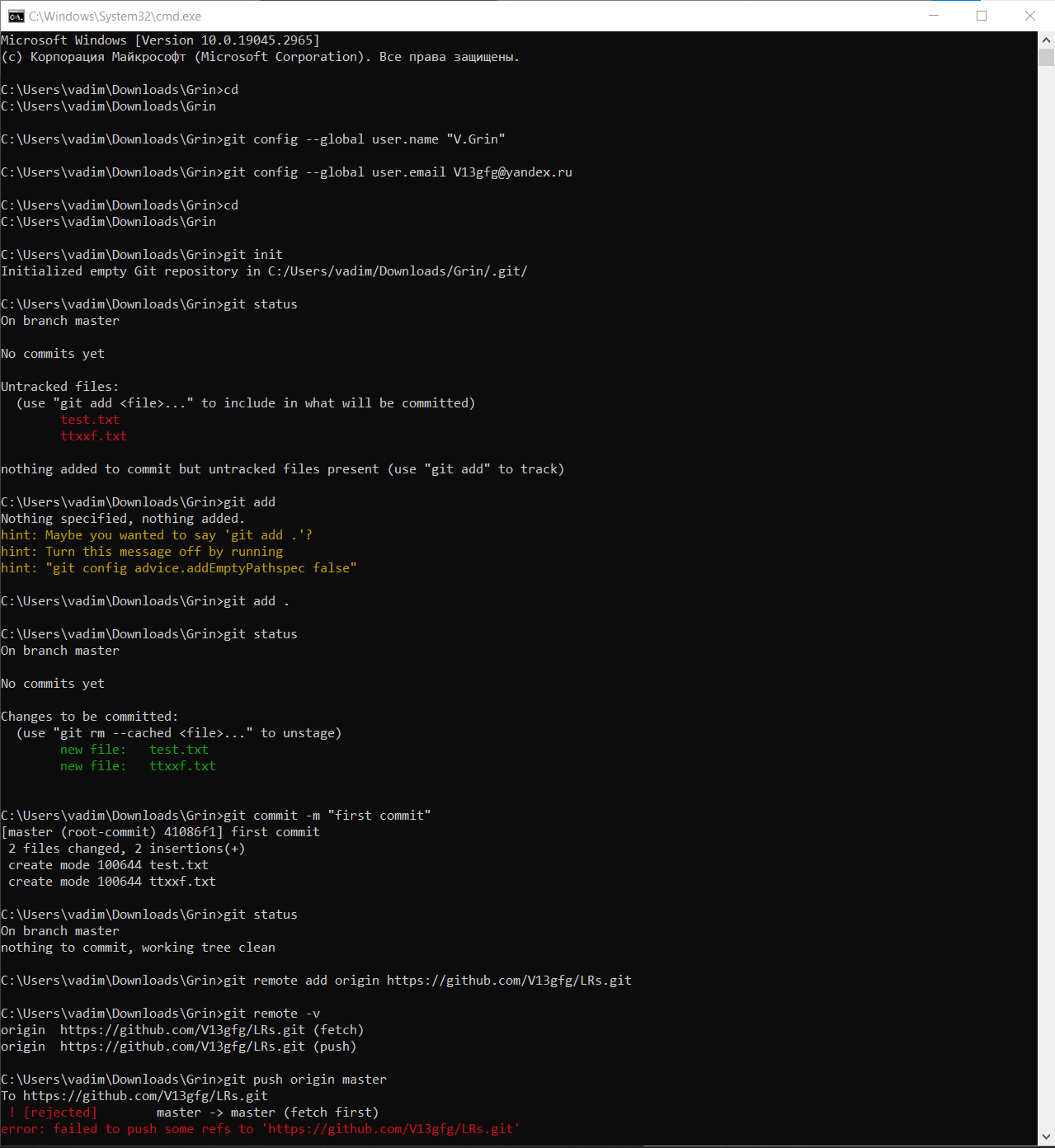


1. Создали файлик test.txt и написал наши фамилии. Открыли командную строку «cmd» и перешли с помощью команды «cd» в созданный нами каталог. Настроим самые важные опции и параметры: наше пользователя и адрес электронной почты.

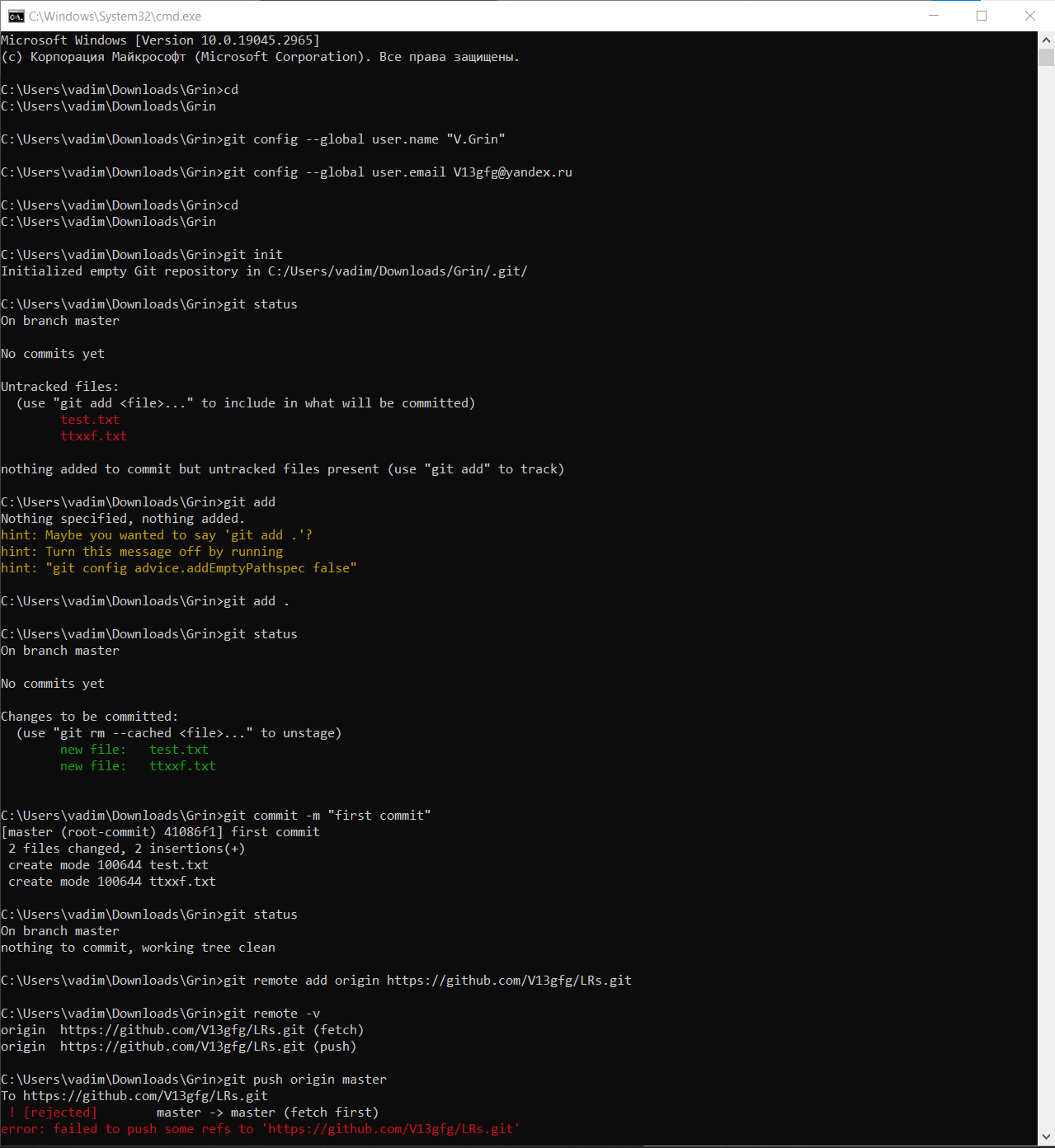


1. Переходим в созданную нами папку, используя команду cd. Теперь необходимо проинициализировать эту папку как git репозиторий. Вводим команду: git init. Командная строка возвращает нам сообщение, что проинициализирована пустая git директория. Теперь используя проводник windows создаем в своей папке текстовый файл и пишем в него наши имена. Вводим команду: git stаtus и нам отображается сообщение, что есть новый файл, но он не отслеживается. Чтобы исправить ситуацию используем команды add . и commit.



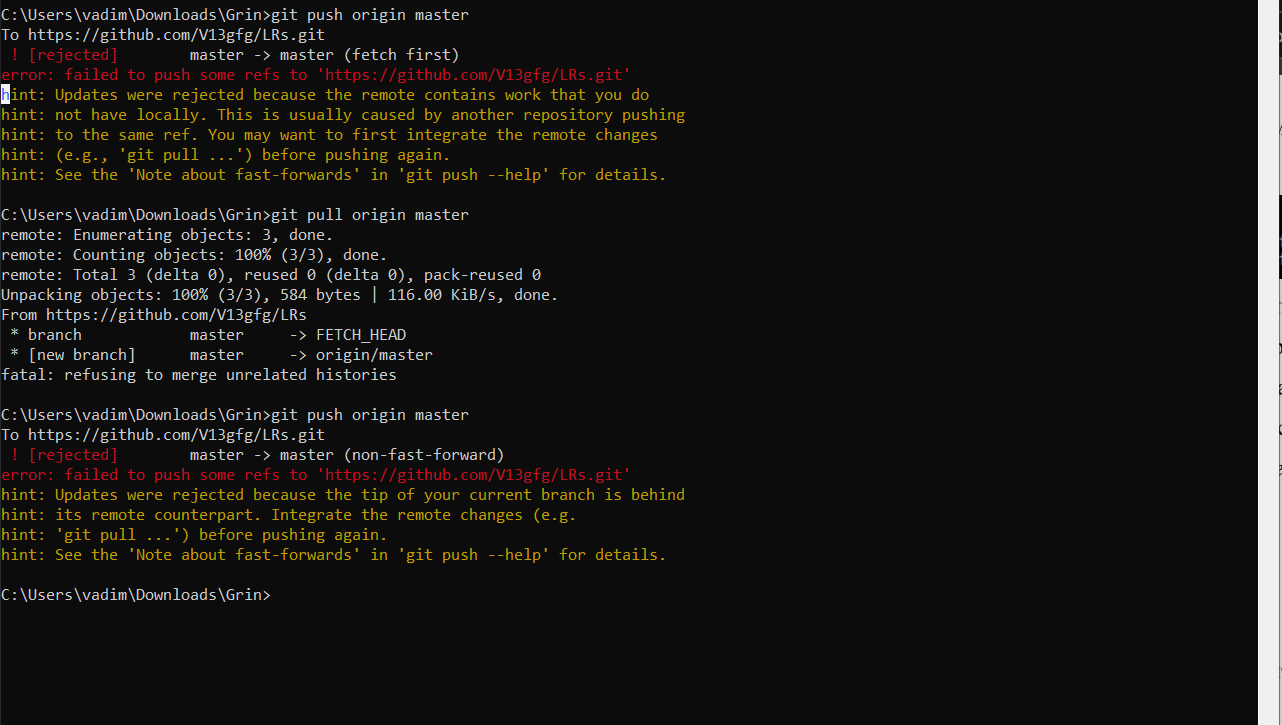


1. Чтобы связать наш локальный репозиторий с репозиторием на GitHub, выполним следующую команду в терминале.

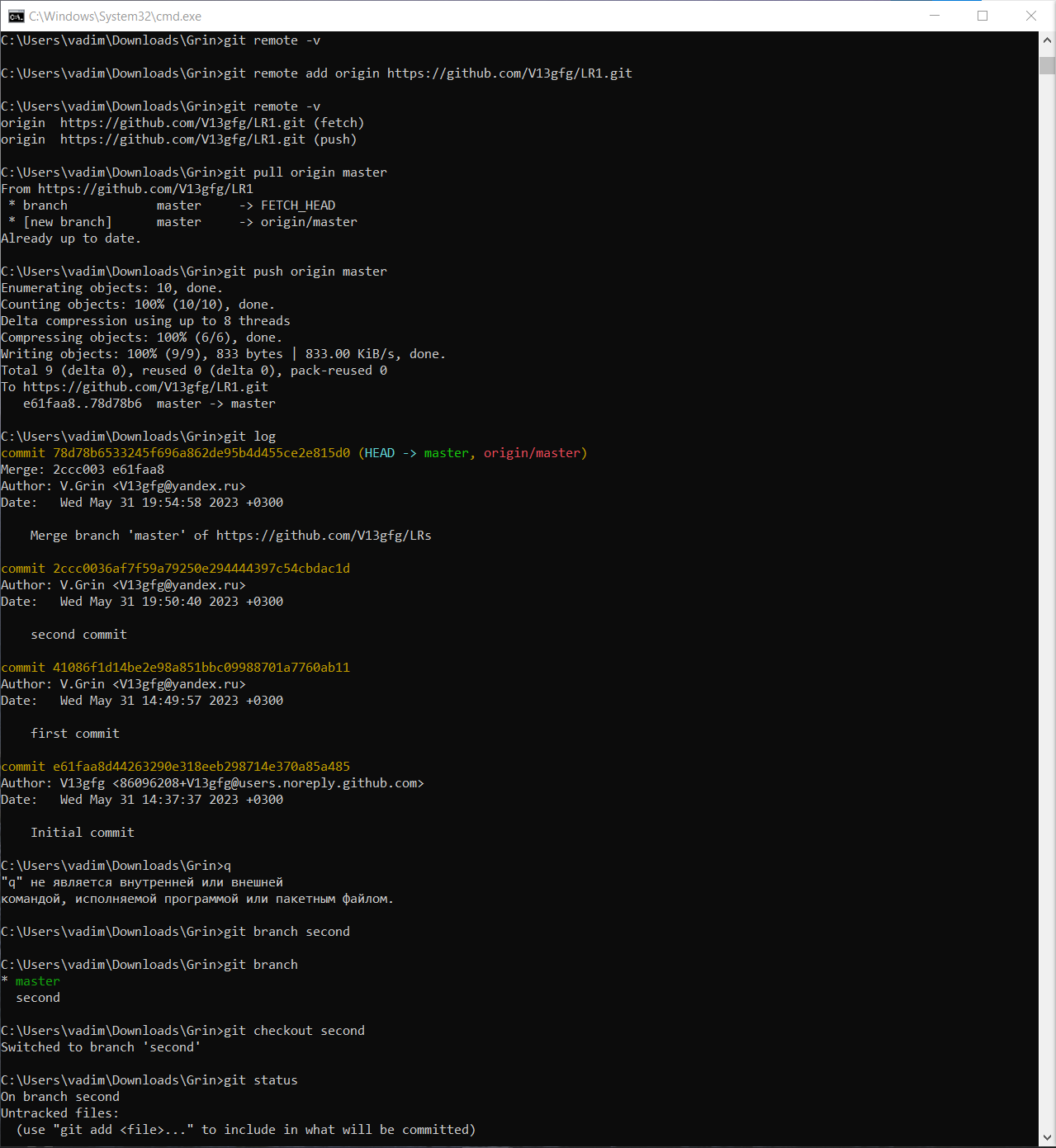


1. Запросим изменения с сервера

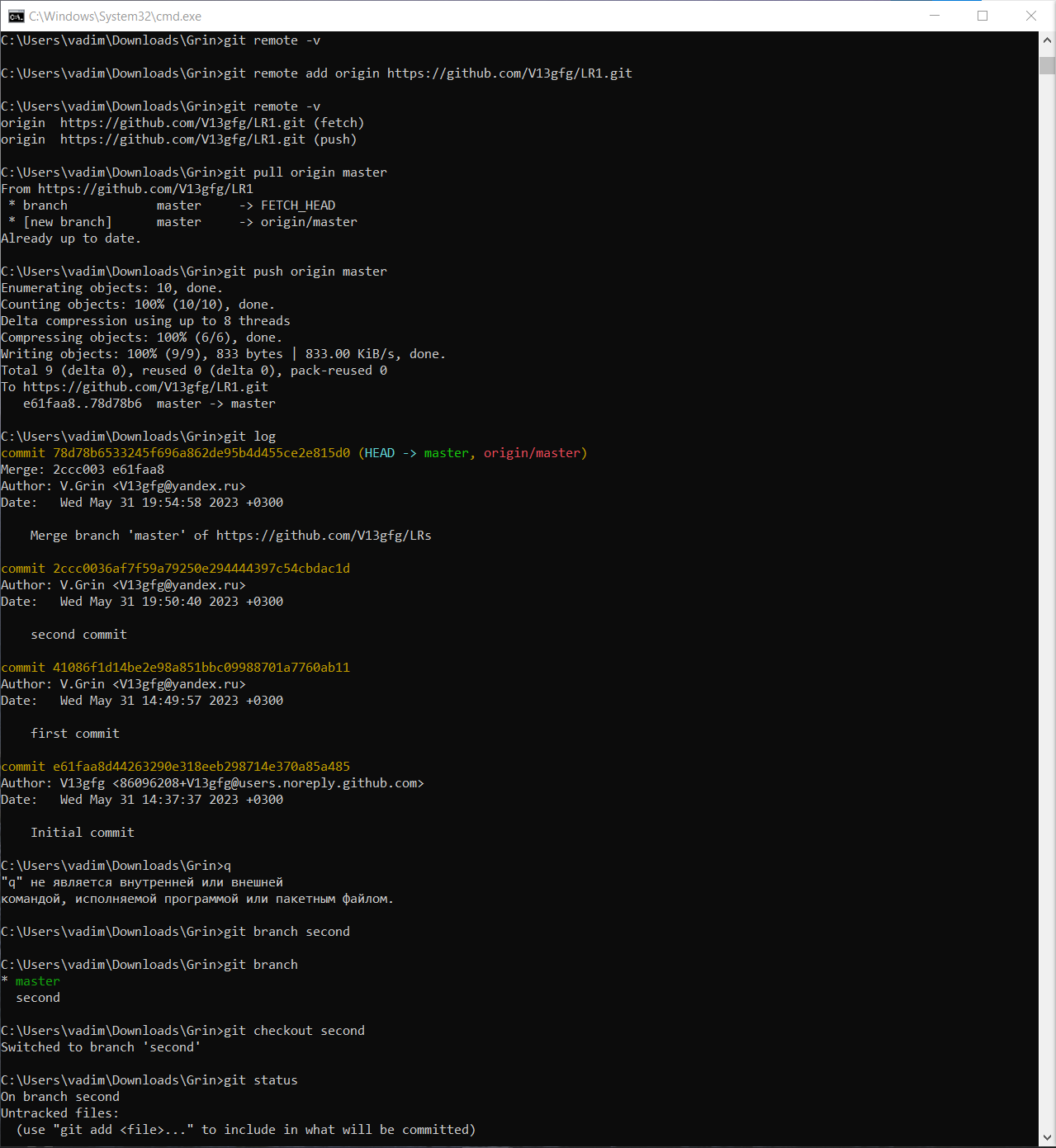
Мы создали изменения в нашем репозитории, создав файл readme и другие пользователи могут скачать изменения при помощи команды pull.



1. Теперь перешлем локальный коммит на сервер. Изменения отправились на сервер:

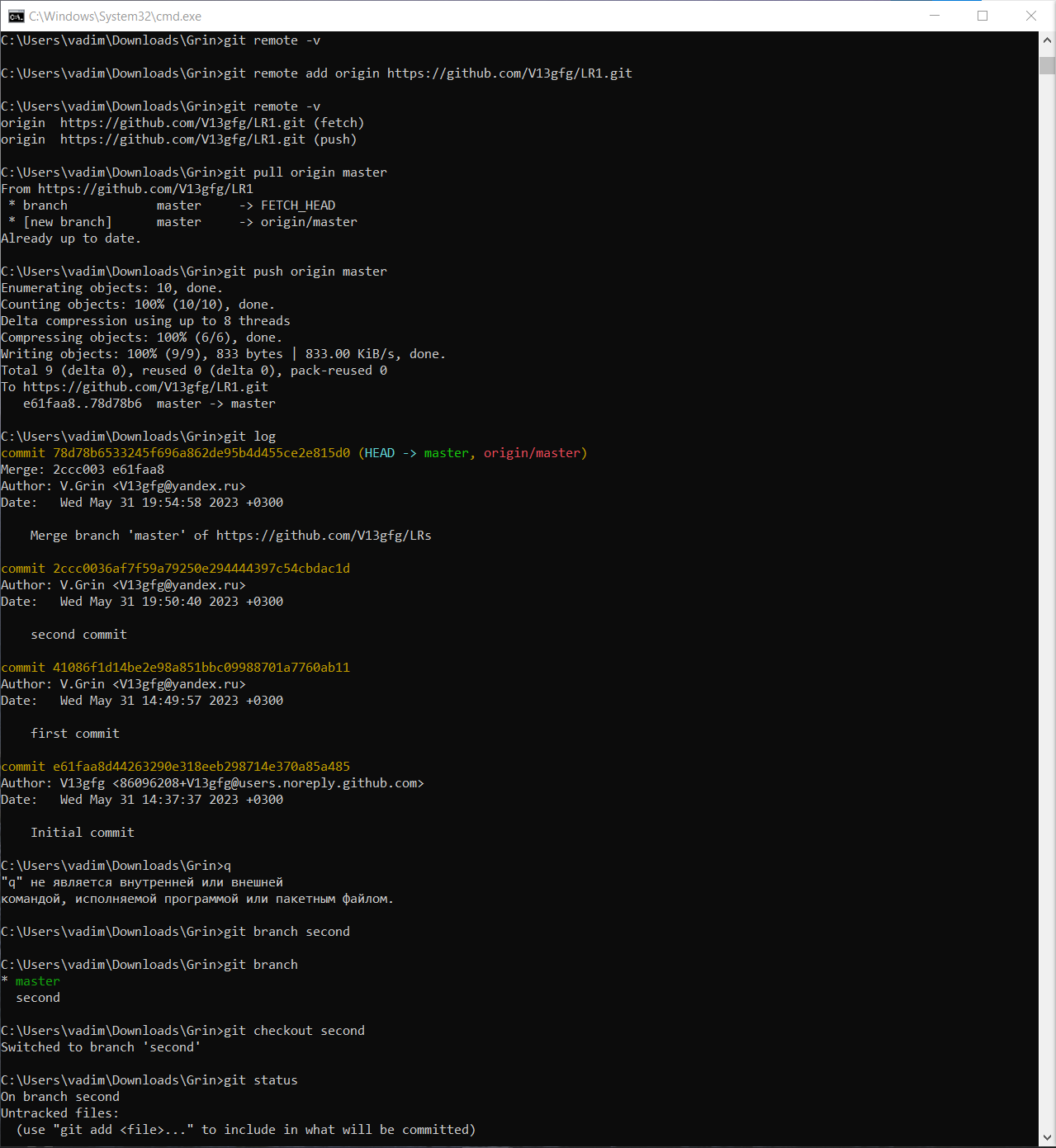


1. Посмотрим наши изменения

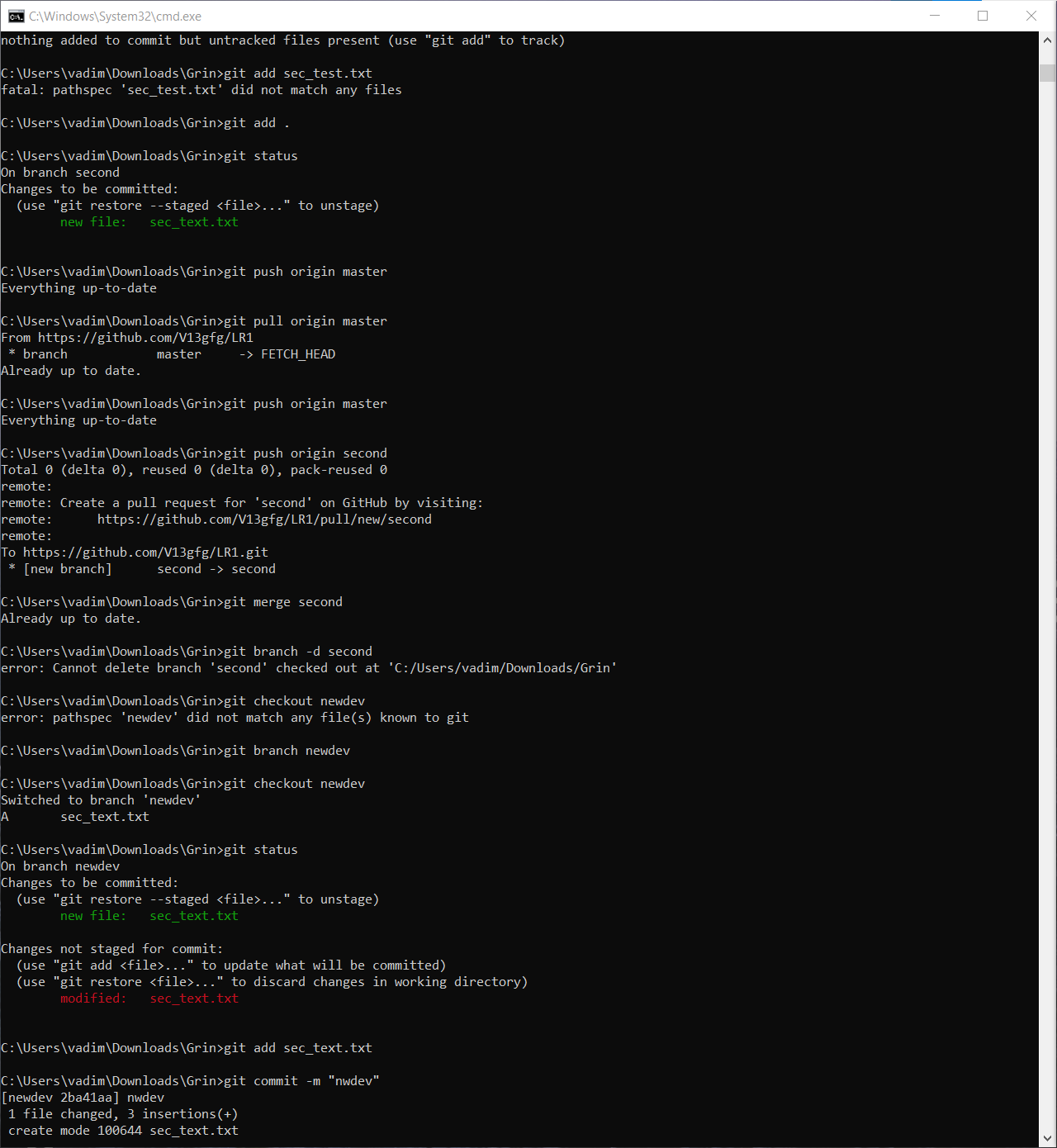


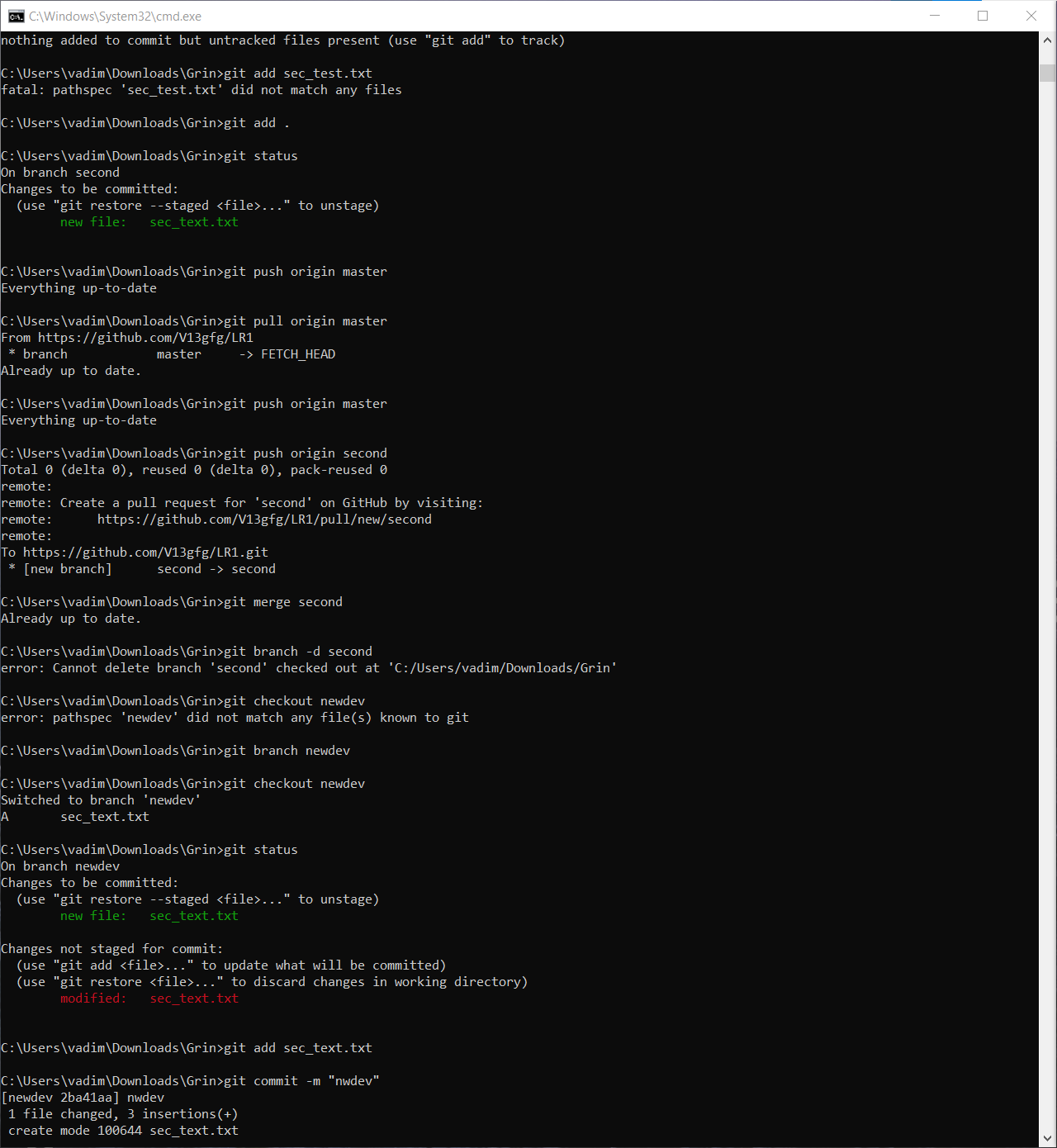
1. Создание новой ветки

Создадим новую ветку second, посмотрим в какой находимся и перейдем на другую:



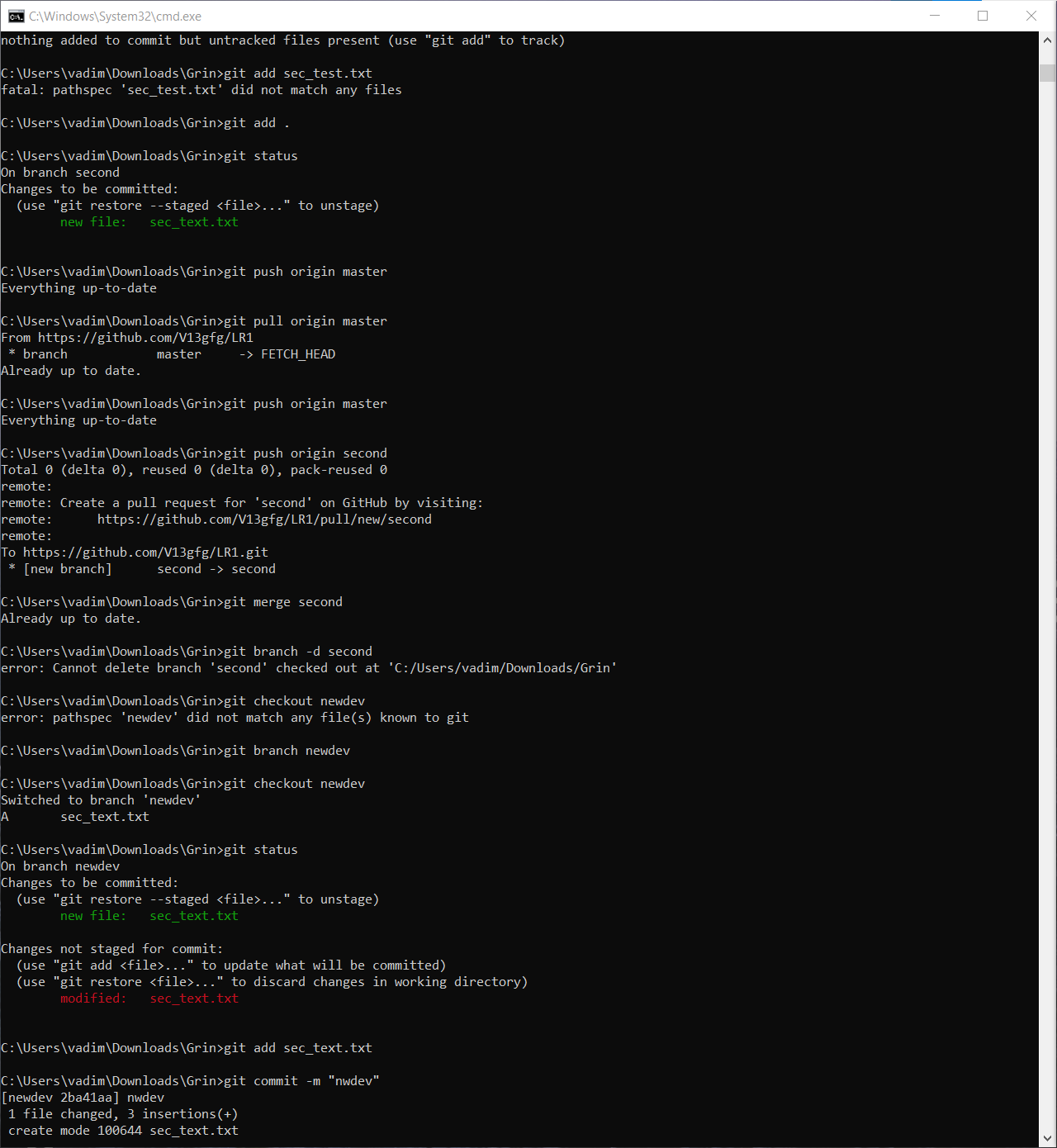
1. Создаем новый файл в нашем локальном репозитории и пишем в нем свои фамилии, добавляем в область подготовленных файлов, коммитим и отправляем на сервер:





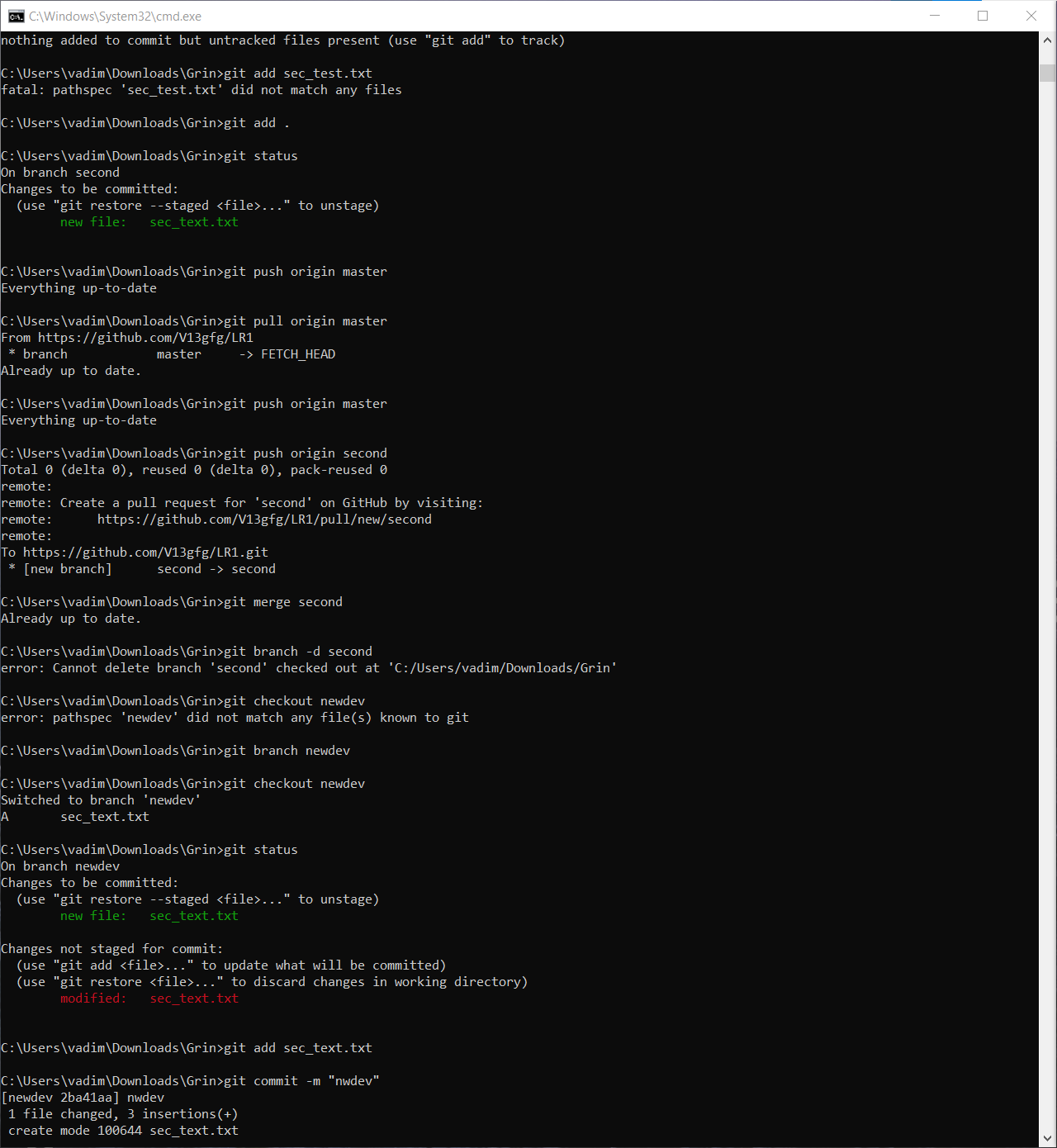
1. **Слияние веток**

Снова переключаемся на ветку master, объединяем ветки с помощью команды git merge second и удаляем ветку second:



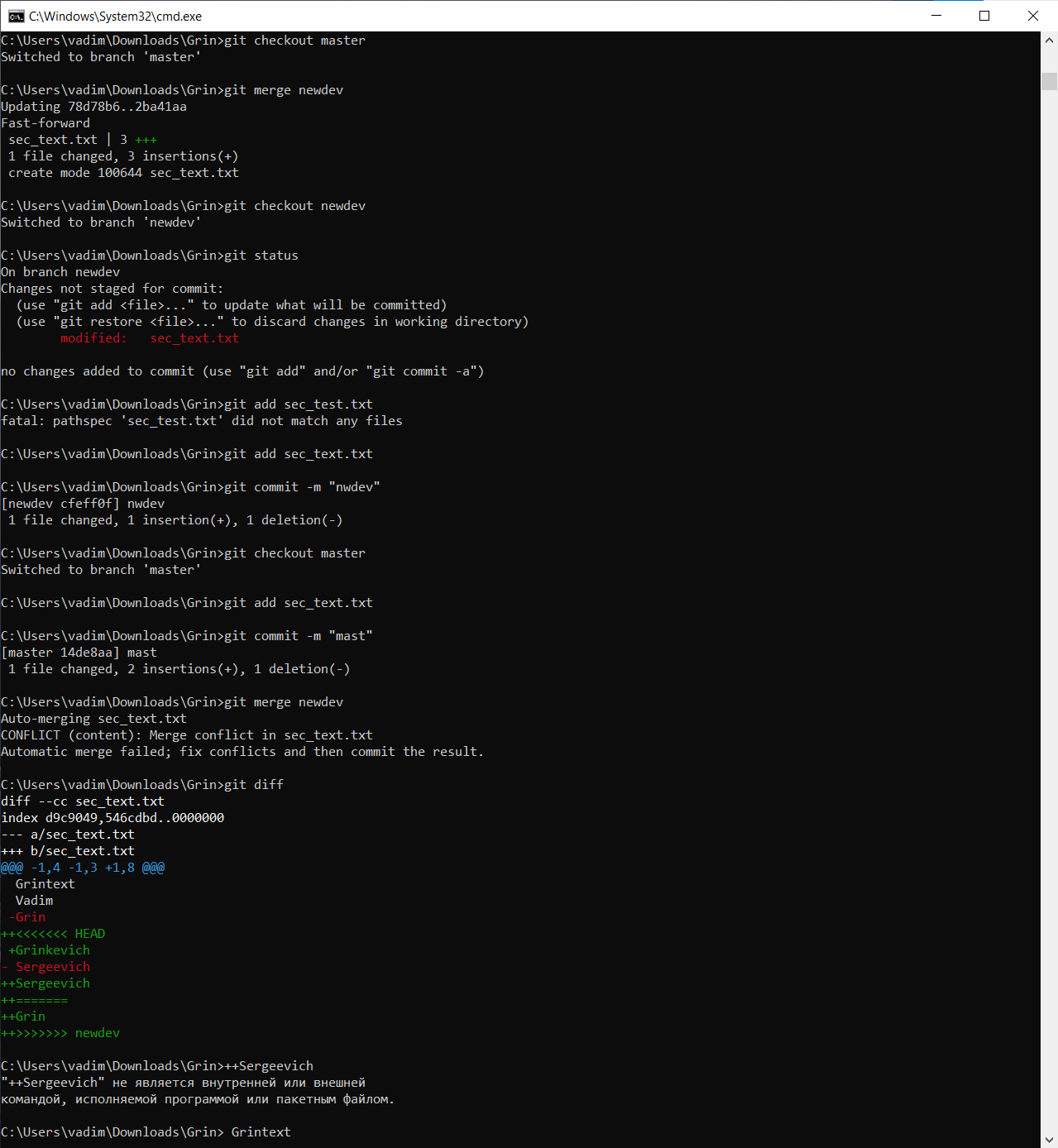
1. Теперь представив, что задание усложняется, предполагаем, что в двух ветках могут быть одинаковые файлы и над ними работают разные разработчики.

Создаем ветку с названием newdev. Переключаемся на нее. Добавляем в файл с именами и отчествами. Фиксируем и коммитим изменения.

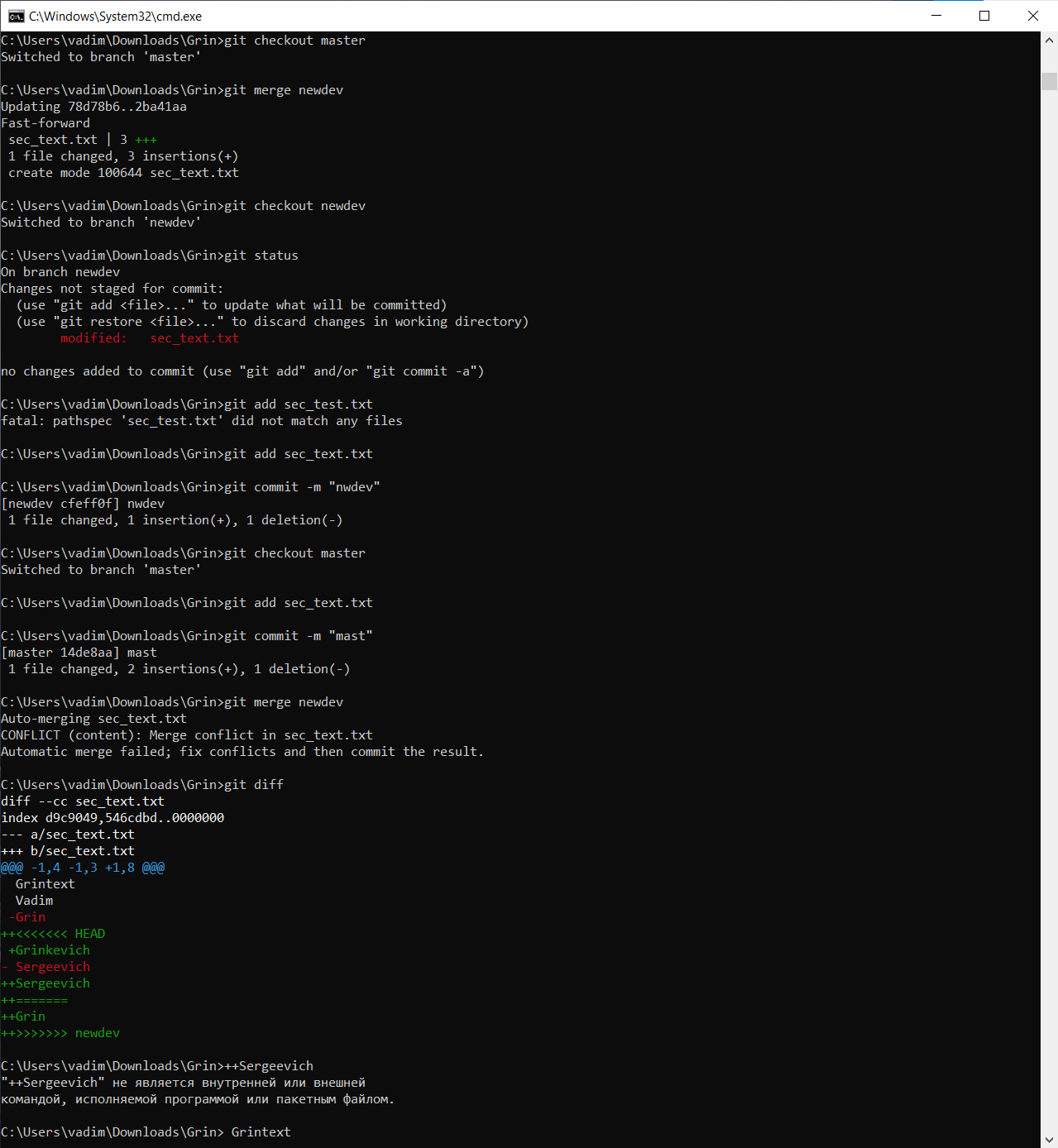


1. **Просмотр изменений и разрешение конфликтов**

Объединим ветки git merge newdev. Теперь ничего не получится, т.к. есть изменения в обеих ветках. Нужно разрешить конфликты:



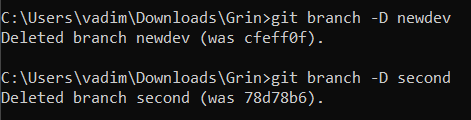
Наберем команду для просмотра изменений git dif:



Приложение отметило строки, содержащие конфликт:

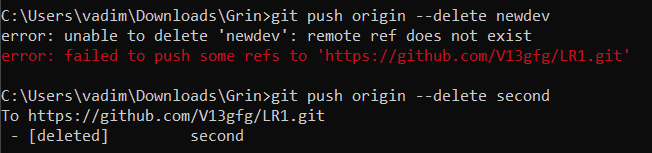
Над разделителем ======= мы видим последний (HEAD) коммит, а под ним — конфликтующий. Перепишем все, удалив разделители (HEAD, ===, <<>>), и дадим git понять, что закончили.

Удаляем ветки newdev и second:



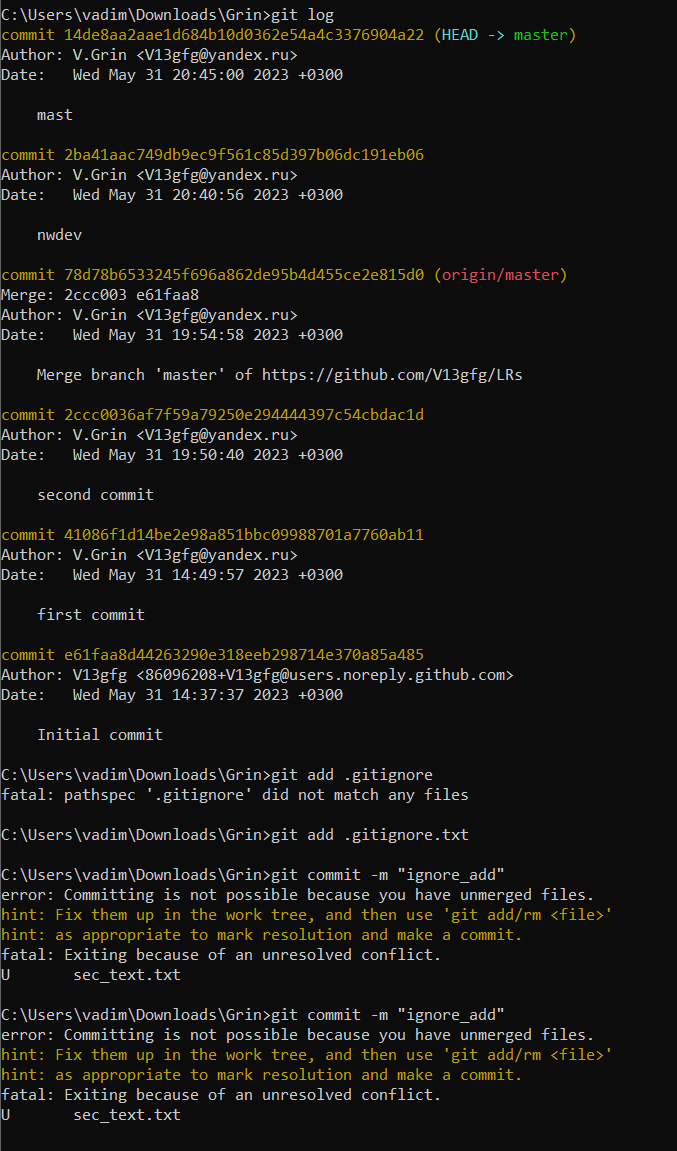
1. **Удаление веток на сервере**

Удаляем ветки с github:



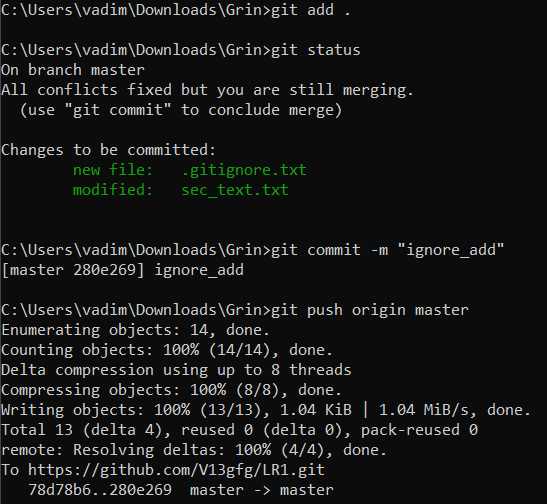
1. **Возврат к предыдущему состоянию**

Чтобы посмотреть все коммиты, можно использовать команду git log:



1. Отправка только нужных файлов на сервер

Создаем вручную файл под названием «.gitignore» и сохраняем его в директорию проекта. Внутри файла перечисляем названия файлов/папок, которые нужно игнорировать, каждый с новой строки. Файл «.gitignore» добавляем, коммитим и отправляем на сервер.



**Вывод:** мы провели работу с системами контроля версий на примере Git Hub и научились создавать локальный репозиторий, пересылать локальный коммит на сервер иделать слияние веток**.**