

**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**

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

|  |  |  |
| --- | --- | --- |
| Выполнил  студент гр. АДБ-21-07: | \_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_  *(дата) (подпись)* | Голубев Д.А. |
| Проверил  к.т.н., доцент | \_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_  *(дата) (подпись)* | Ковалев И.А**.** |

**Москва 2024 г.**

Оглавление:

[Цель лабораторной работы: 3](#_Toc130663739)

[Порядок выполнения работы: 3](#_Toc130663740)-27

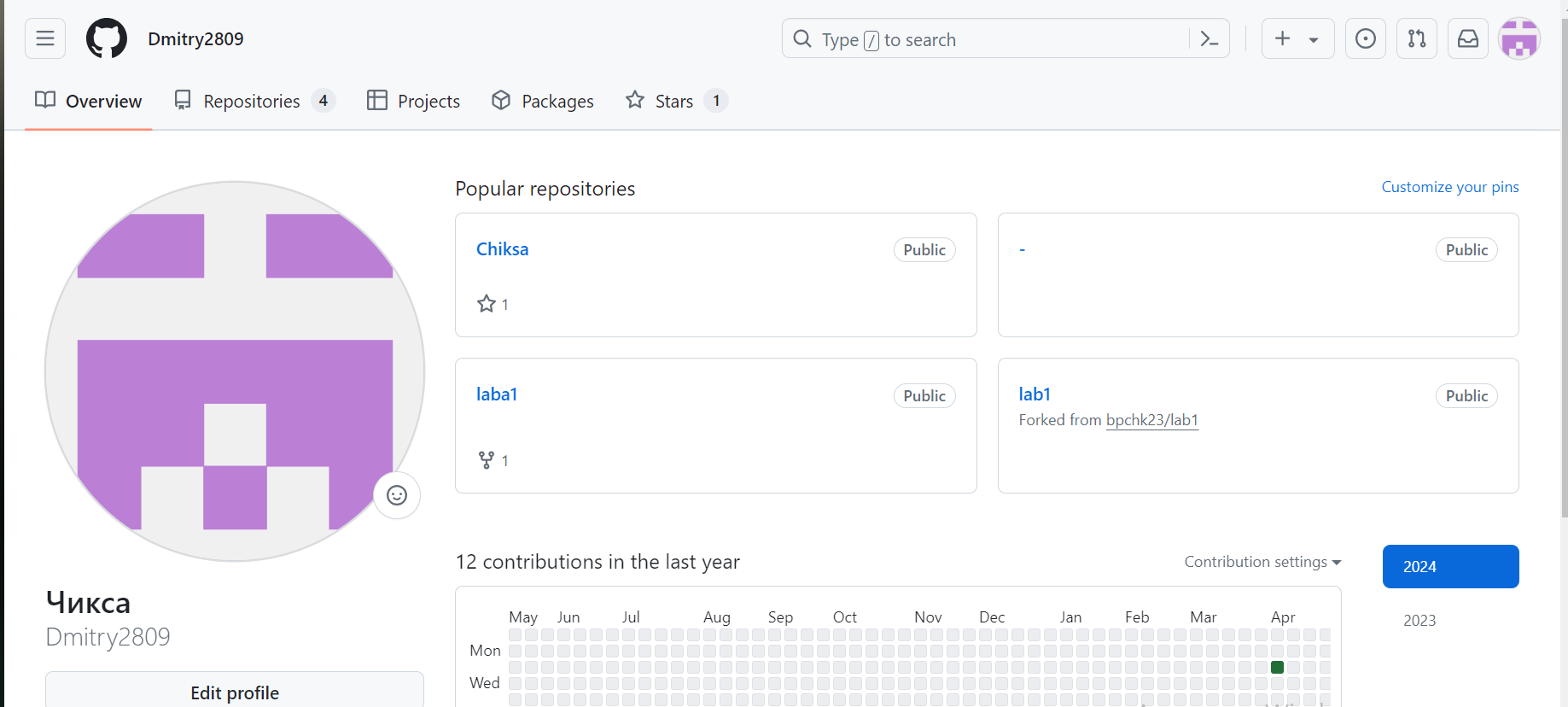
[Вывод: 27](#_Toc130663741)

**Цель лабораторной работы:**

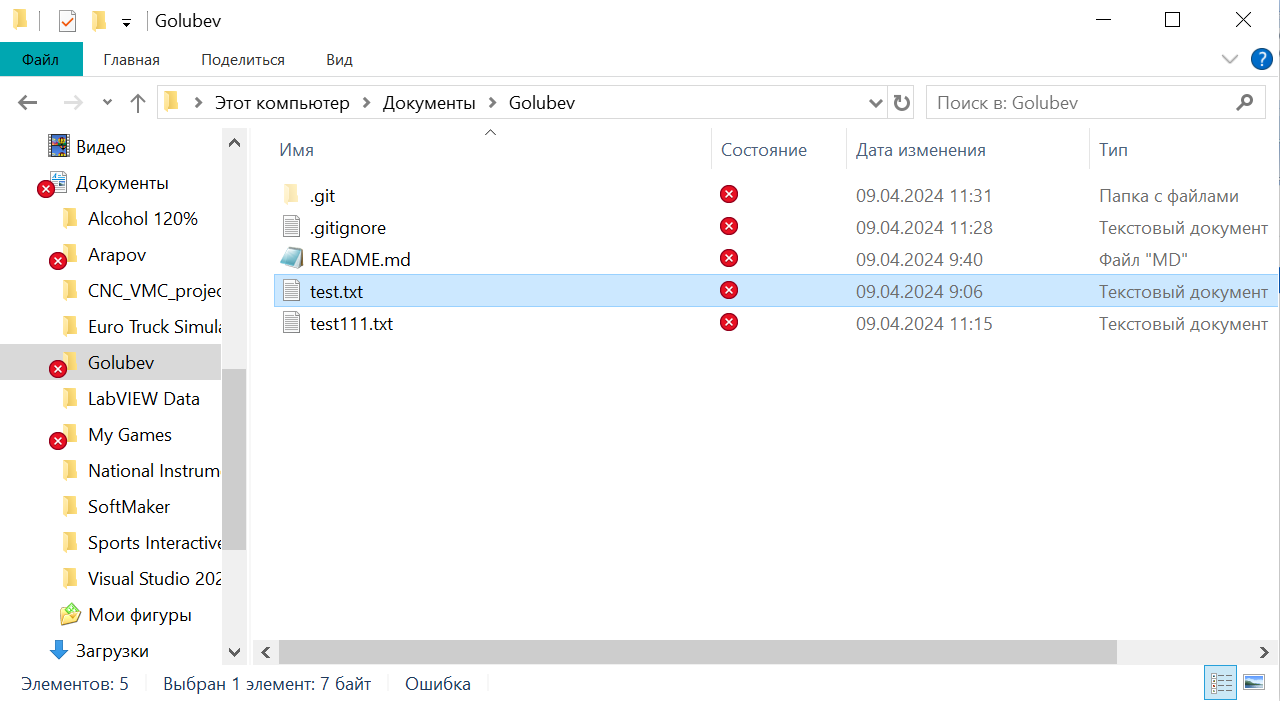
Научиться работать с системами контроля версий GitHub. Узнать краткую историю GitHub.

**Порядок выполнения работы:**

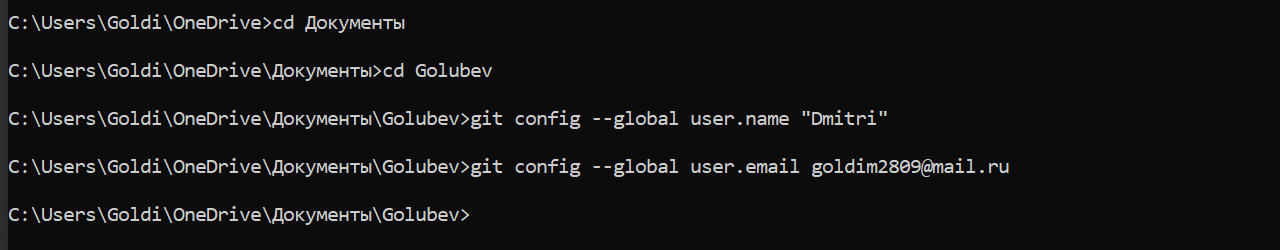
1) Создаём или входим в свою учётную запись на github.com и создаём новый репозиторий.



2) Создаем на диске локальную папку со своей фамилией.

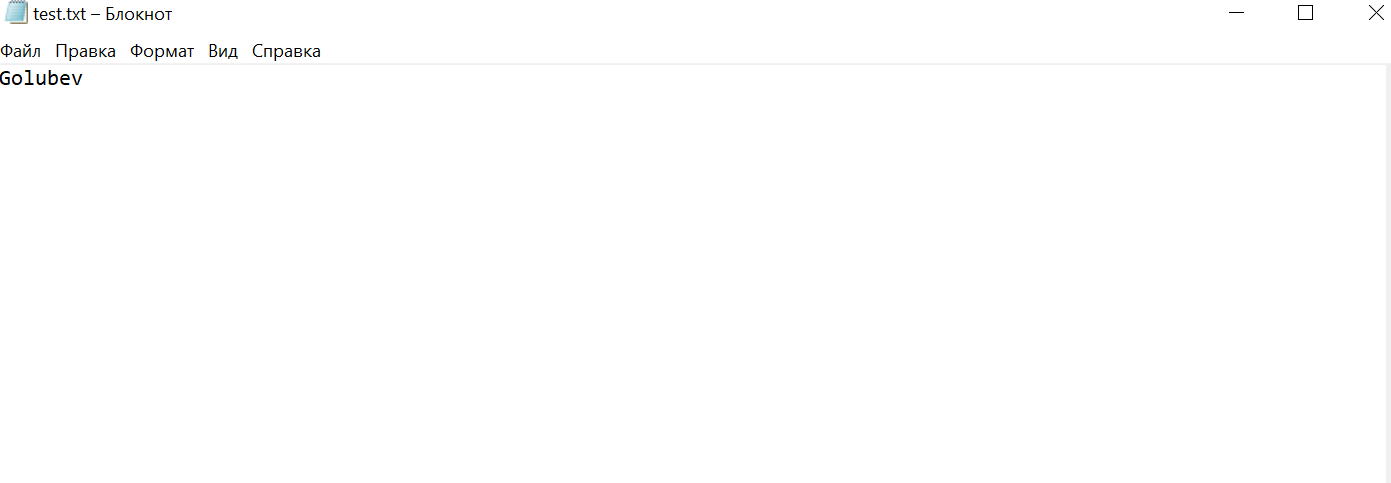


3) Переходим в созданную нами папку в командной строке используя команду cd.

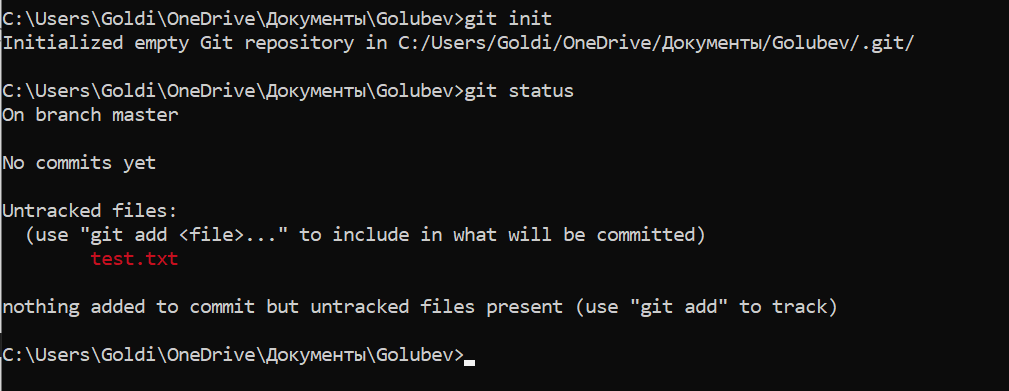


4) Теперь необходимо проинициализировать эту папку как git репозиторий Введем в командной строке: git init.

Командная строка вернет нам сообщение, что проинициализирована пустая git директория. Теперь используя проводник windows создаем в своей папке любой текстовый файл и прописываем в него своё имя.

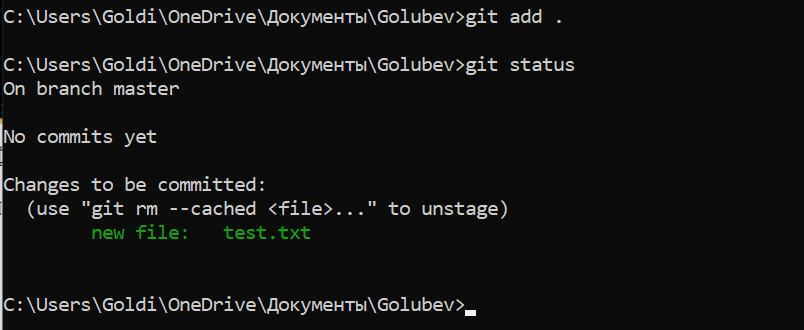


5) Вводим команду: git stаtus и нам отобразиться сообщение, что есть новый файл, но он не отслеживается.



6) Фиксация изменений в области заготовленных файлов

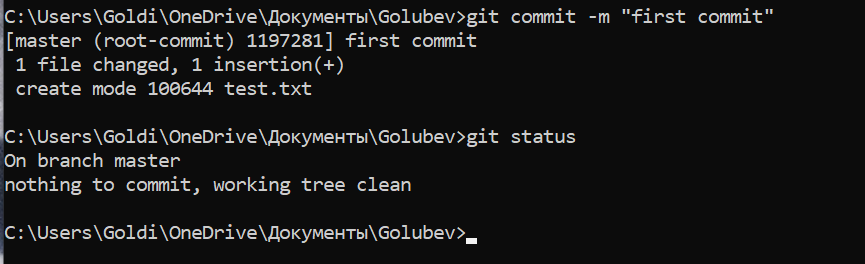
Если измененных файлов много, то можно сделать так: git add.



7) Теперь закомитим наши файлы. Коммит представляет собой состояние репозитория в определенный момент времени как некий опечаток во времени.

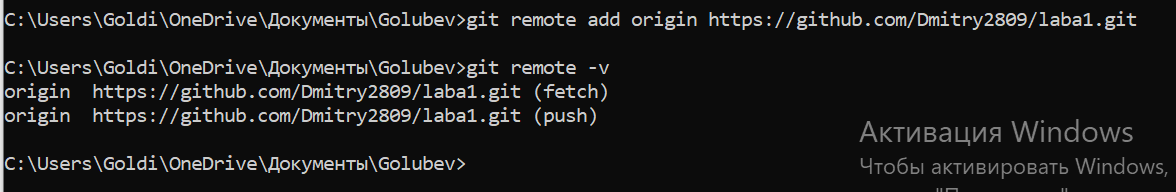
Коммит отмечается hash суммой и к которому мы можем в любой момент времени вернуться.

Чтобы зафиксировать изменения, нам нужно хотя бы одно изменение в области подготовки (мы как раз создали новый файл): git commit -m "first commit".

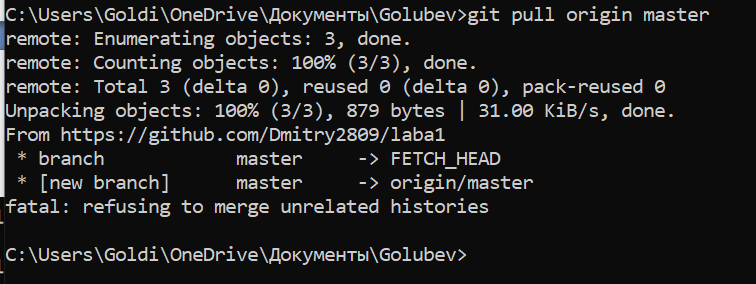


8) Чтобы связать наш локальный репозиторий с репозиторием на GitHub, выполним следующую команду в терминале.

Можно вызвать команду, для просмотра, к какому проекту мы подключены git remote -v



9) Если вы сделали изменения в вашем репозитории (а мы их сделали, создав файл readme), другие пользователи могут скачать изменения при помощи команды pull. git pull origin master.

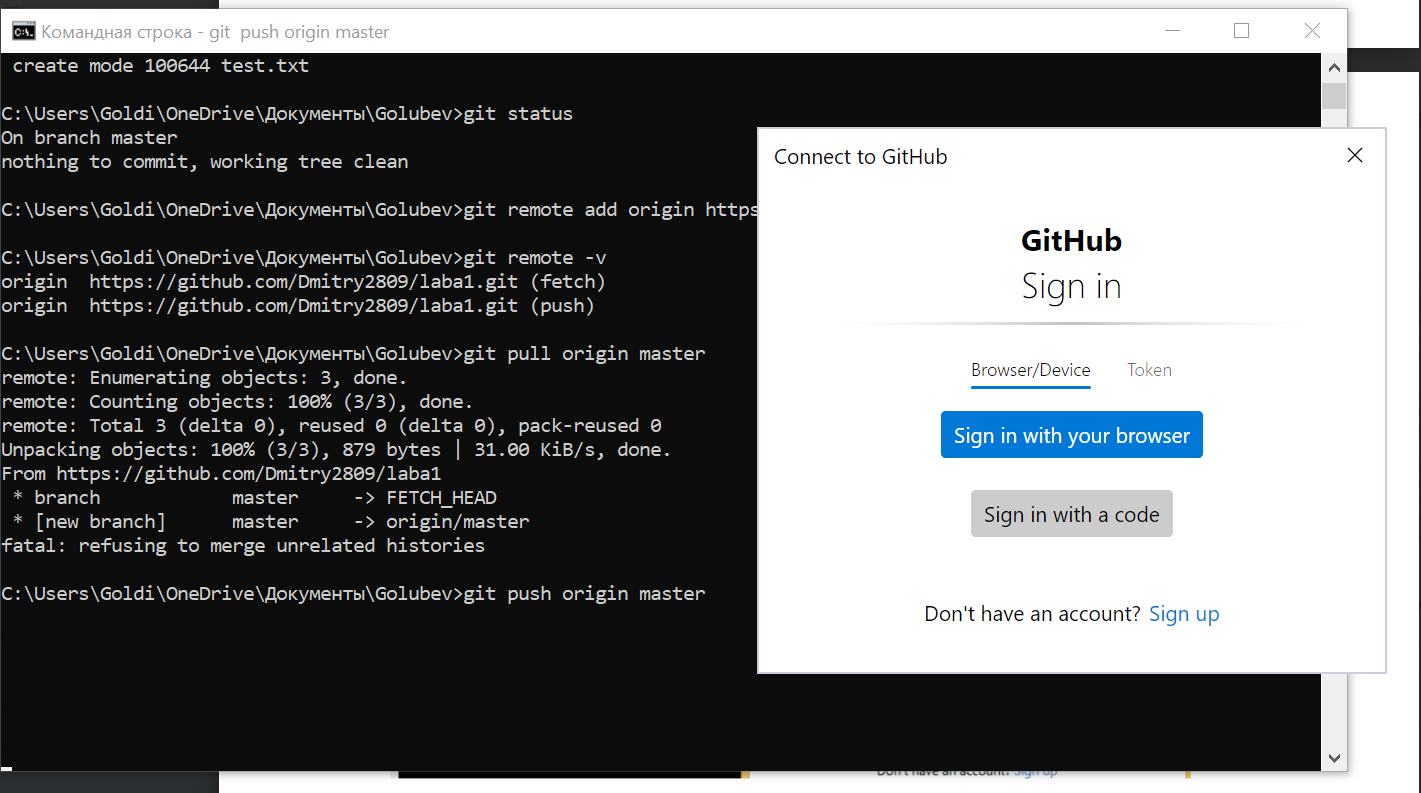


10) Теперь отравим коммит на сервер, команда, предназначенная для этого — push.

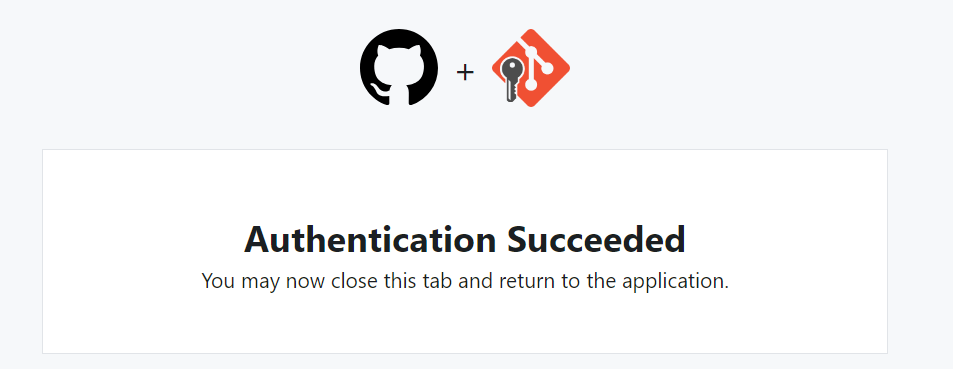
Она принимает два параметра: имя удаленного репозитория (мы назвали наш origin) и ветку, в которую необходимо внести изменения (master — это ветка по умолчанию для всех репозиториев).

git push origin master

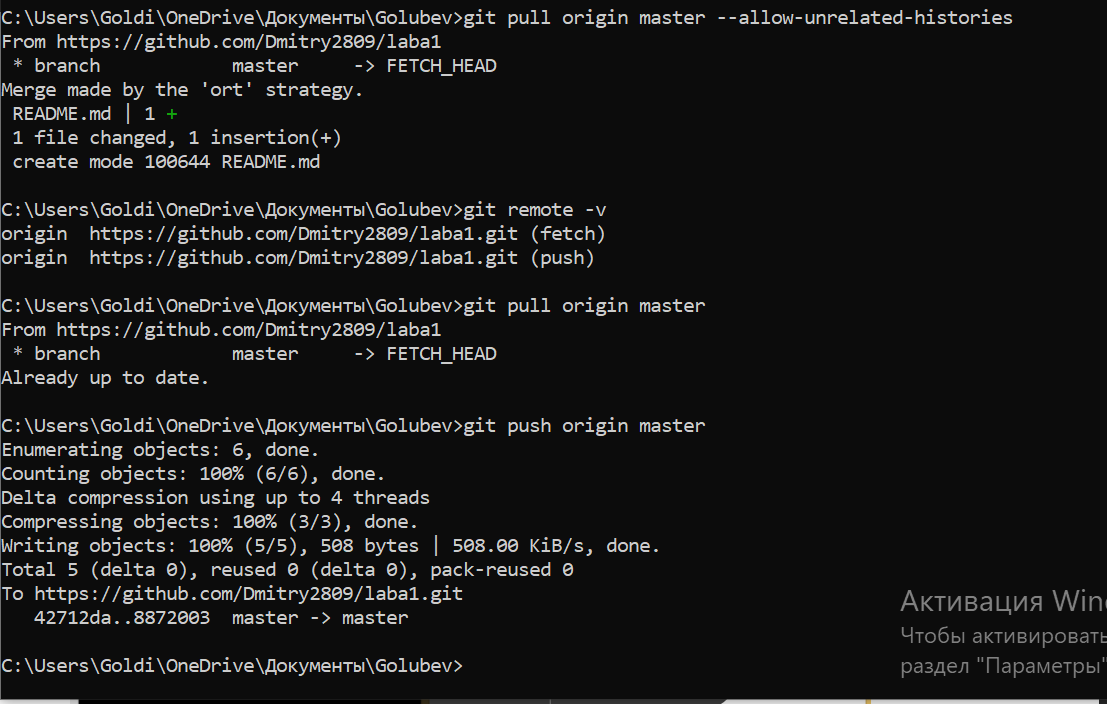
У вас должно появиться окно, где нужно ввести авторизоваться на github.com



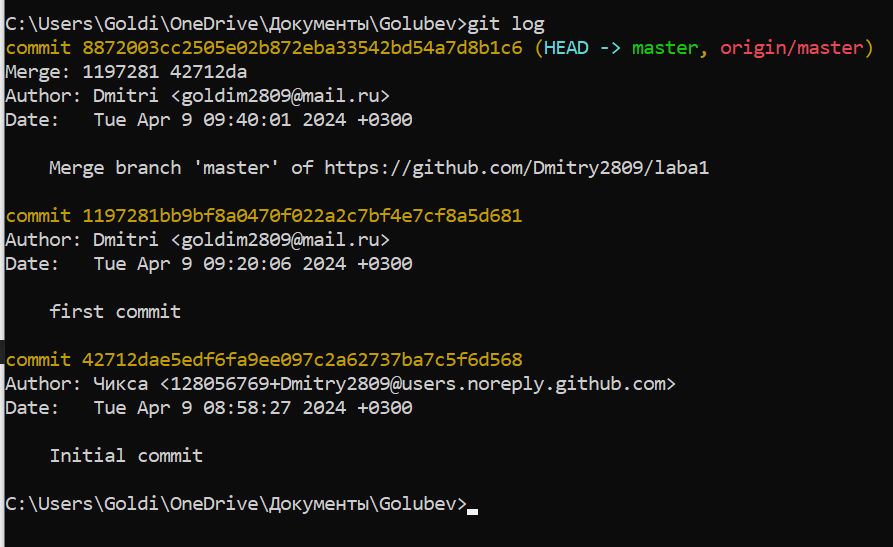
11) На странице нажимаем «Authorize GitCredentialManager».



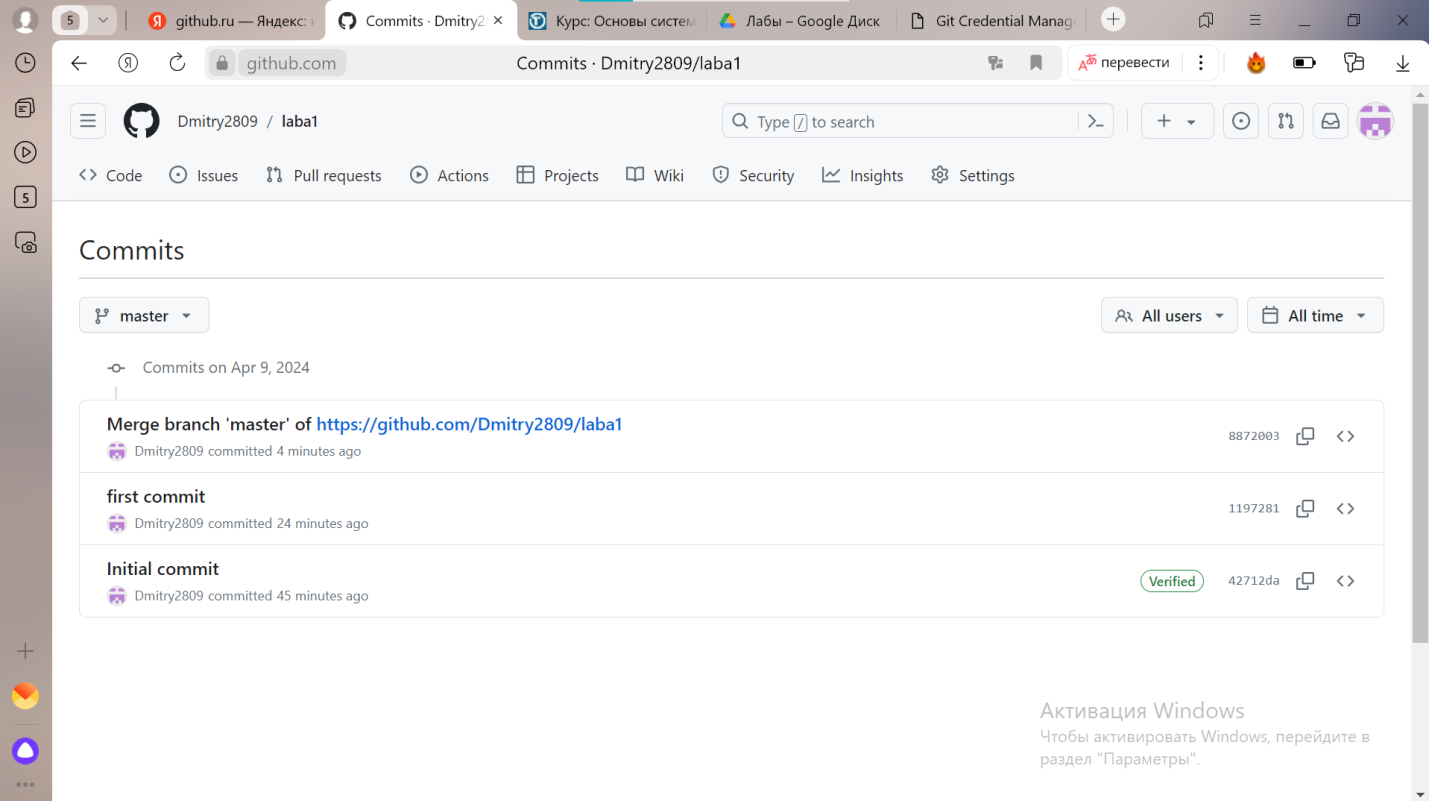
12) Если все правильно, то изменения отправятся на сервер.

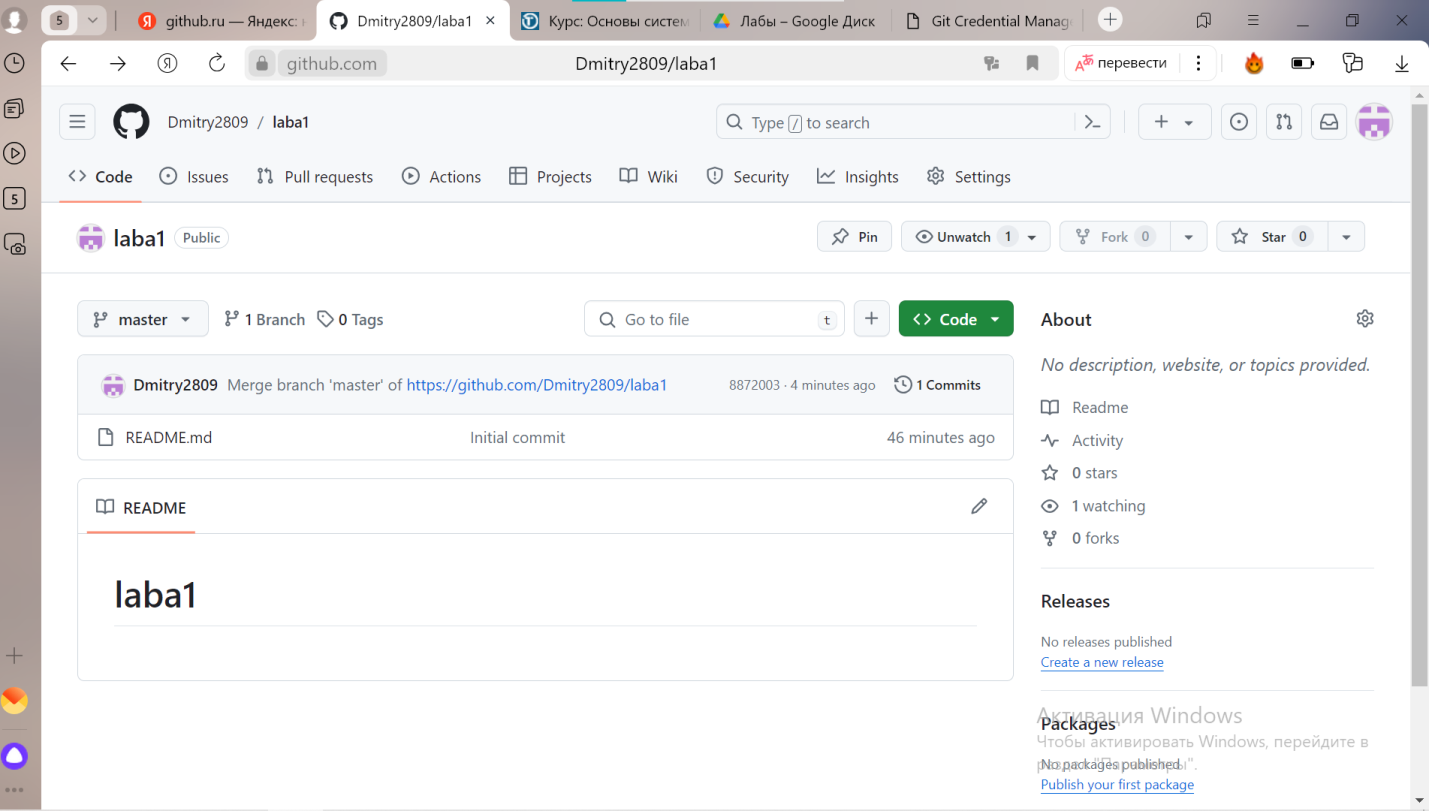


13) Посмотрим наши изменения с использованием команды git log.



14) Посмотрим изменения на github.com

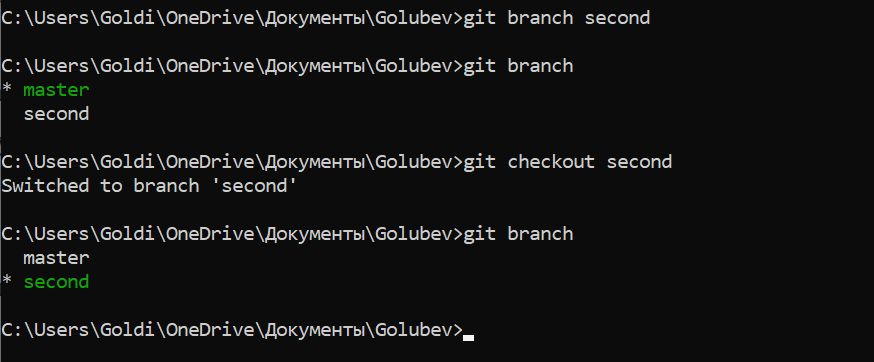




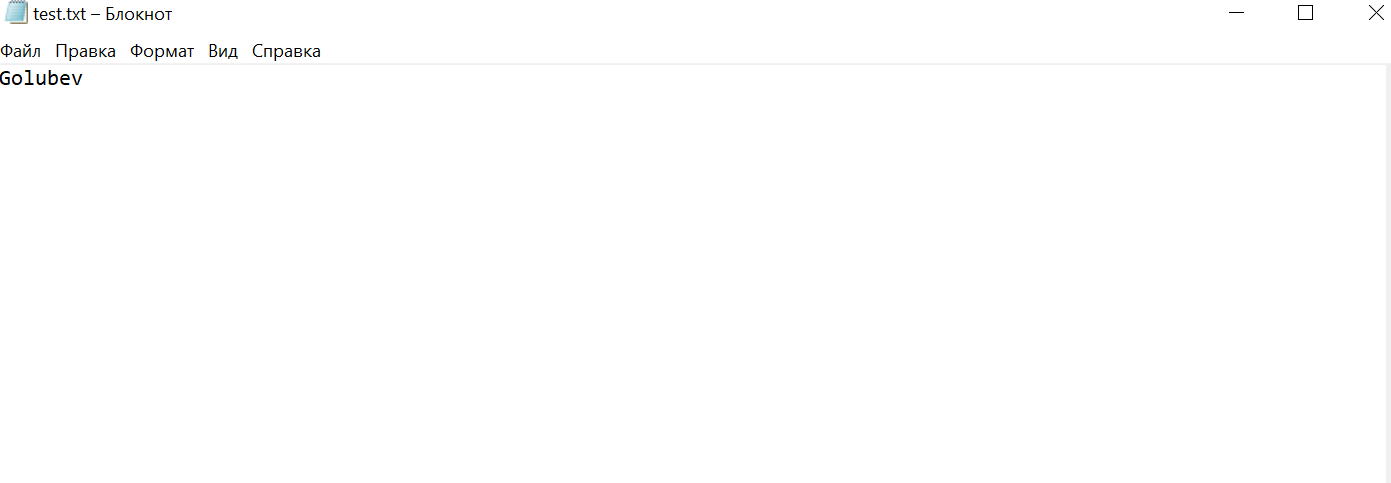
15) Создадим новую ветку second: git branch second.

Можно посмотреть в какой ветки находимся, используя команду git branch.

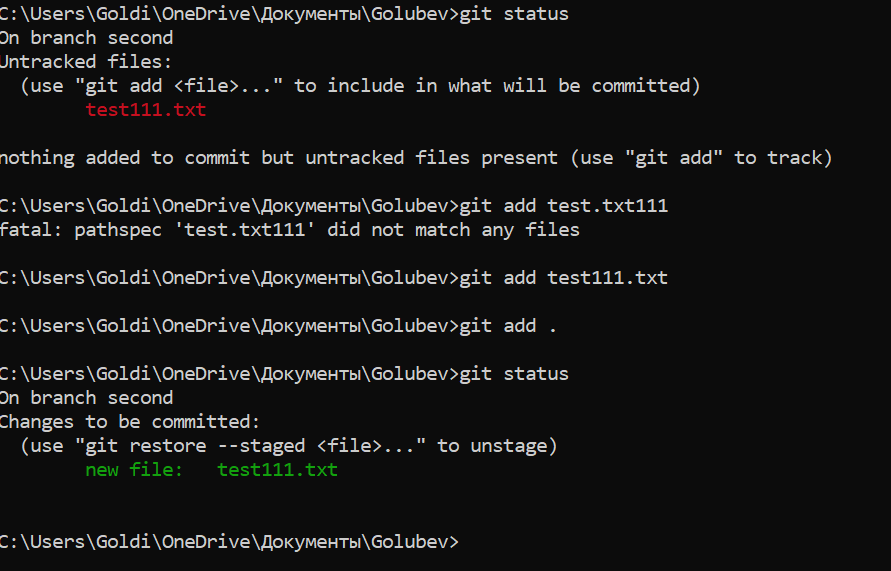
Переключимся на другую ветку, используя команду git checkout second.

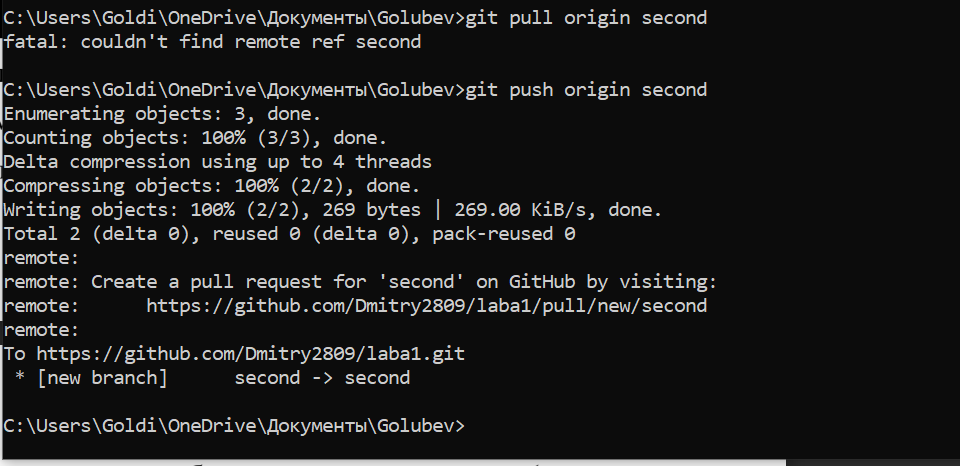


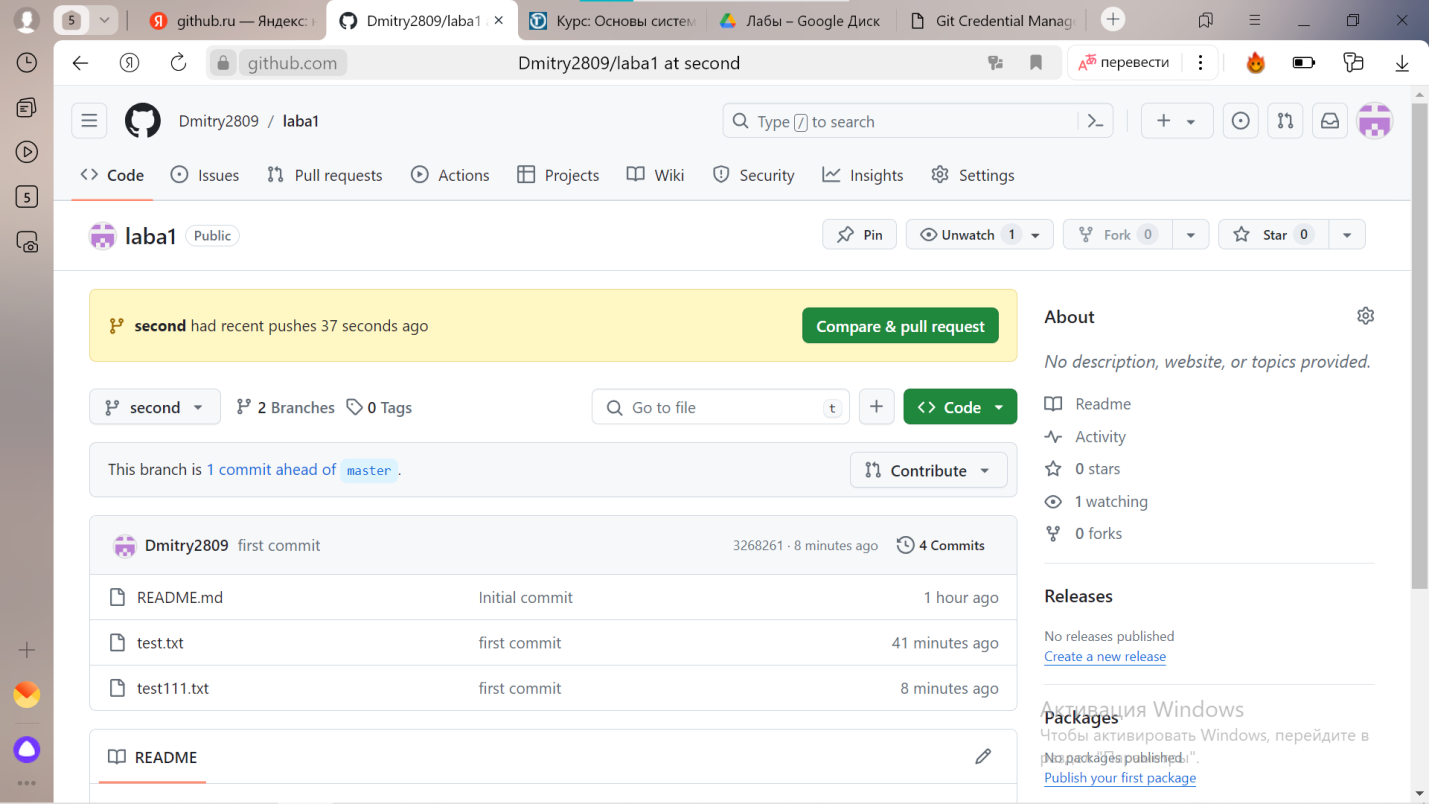
16) Создадим новый файл в нашем локальном репозитории и напишим в нем свою фамилию.



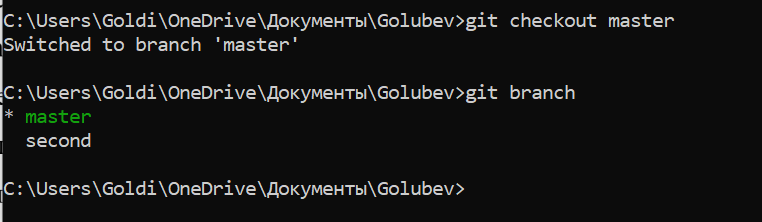
17) Добавим в область подготовленных файлов, закомитим и отправим на сервер.



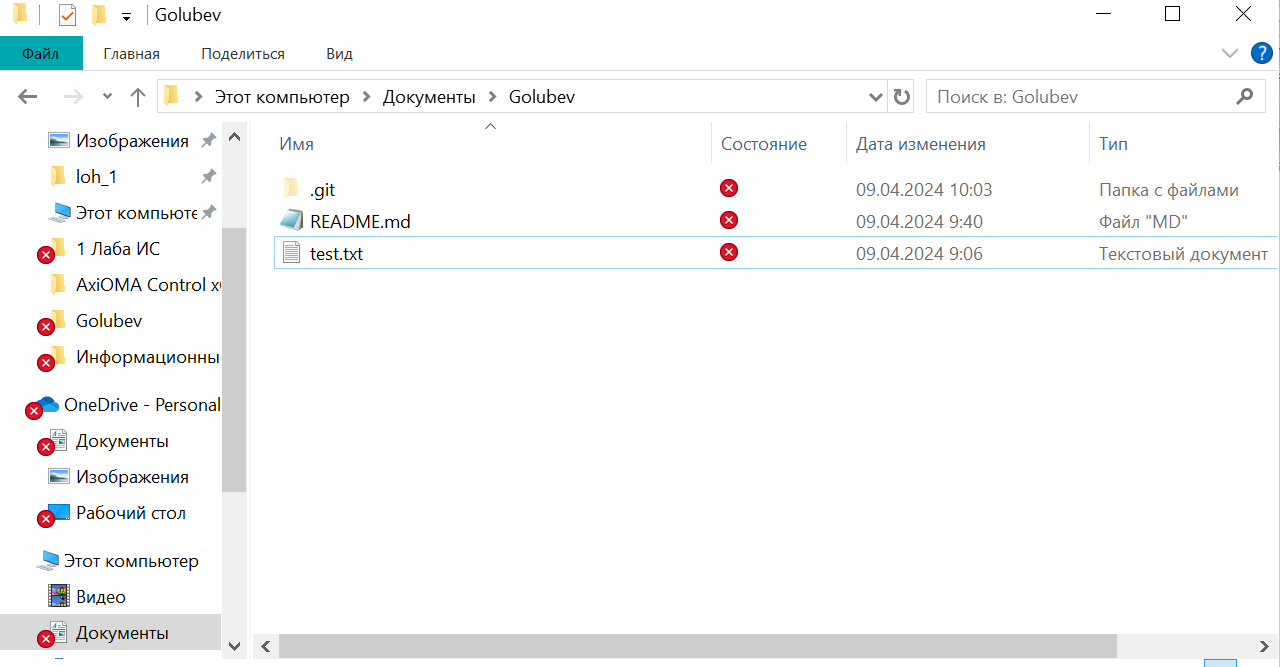




18) Переключимся снова на ветку master.

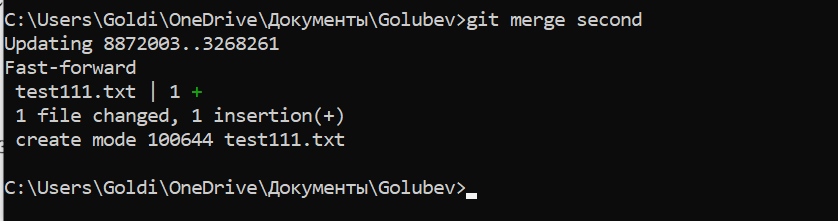


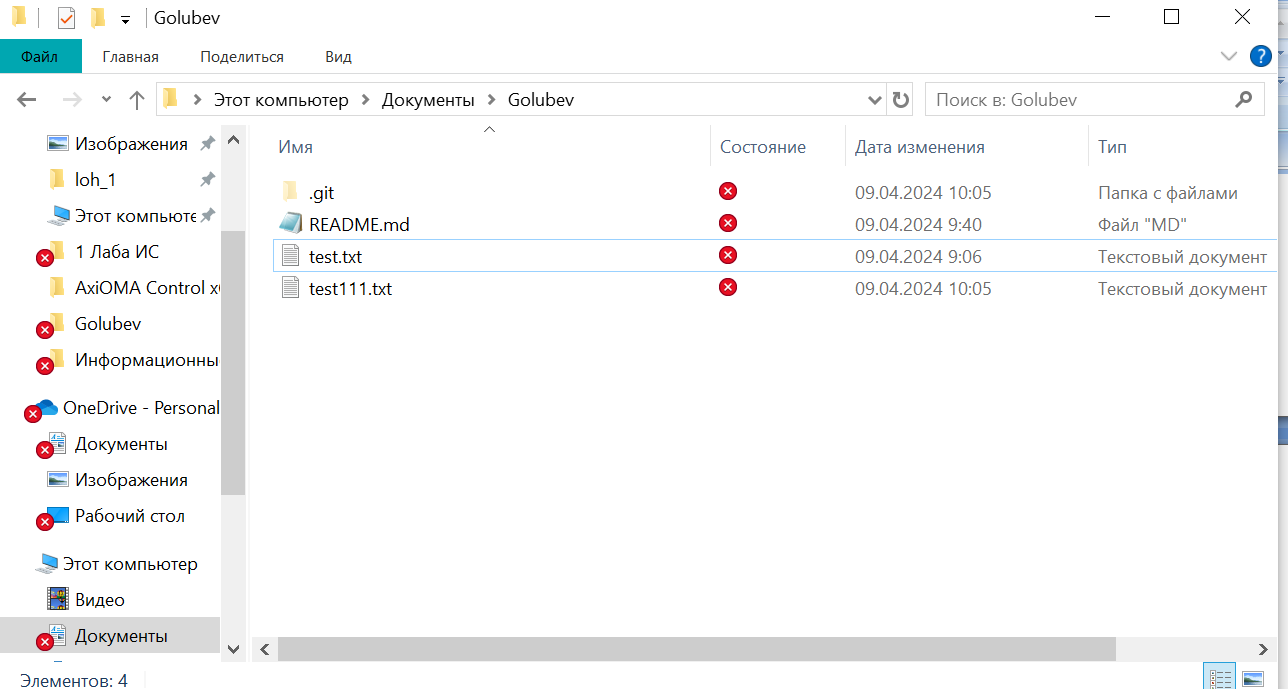
19) Если через проводник windows в папку с проектом, то файла test\_new.txt не будет виден, потому что мы переключились обратно на ветку master, в которой такого файла не существует.



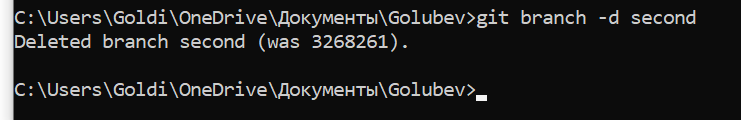
20) Чтобы он появился, нужно воспользоваться merge для объединения веток (применения изменений из ветки second к основной версии проекта).

git merge second

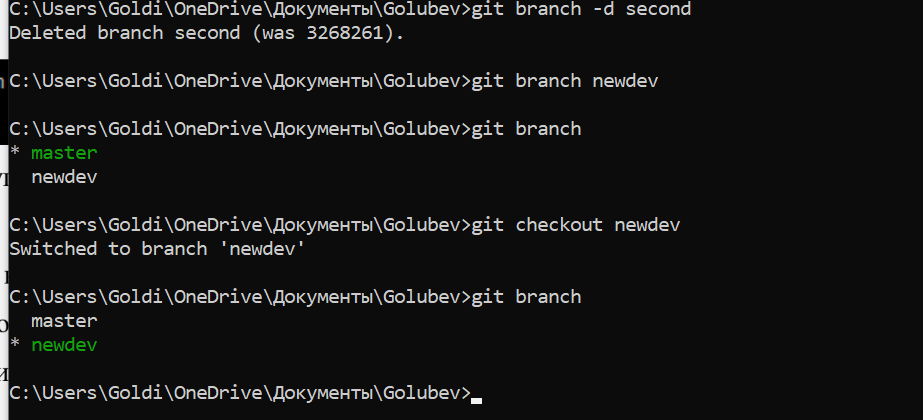




21) Если все прошло без ошибок, можно удалить ветку git branch –d second.



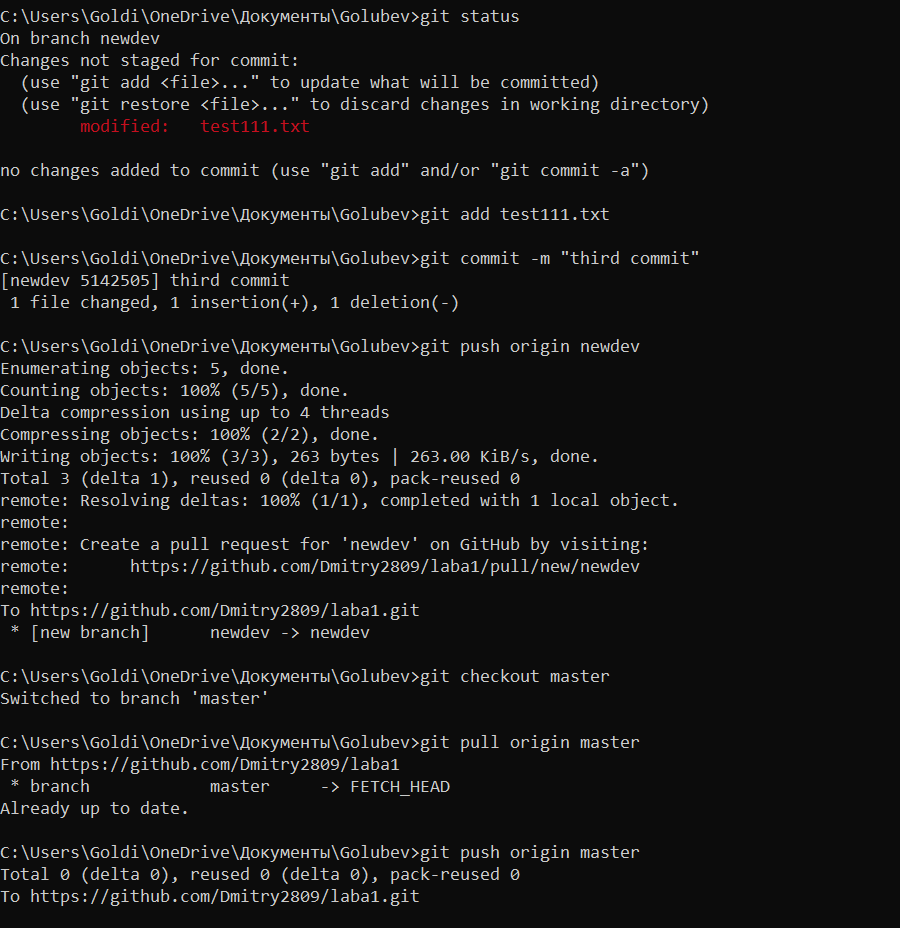
22) Создадим ветку с названием newdev переключимся на нее.



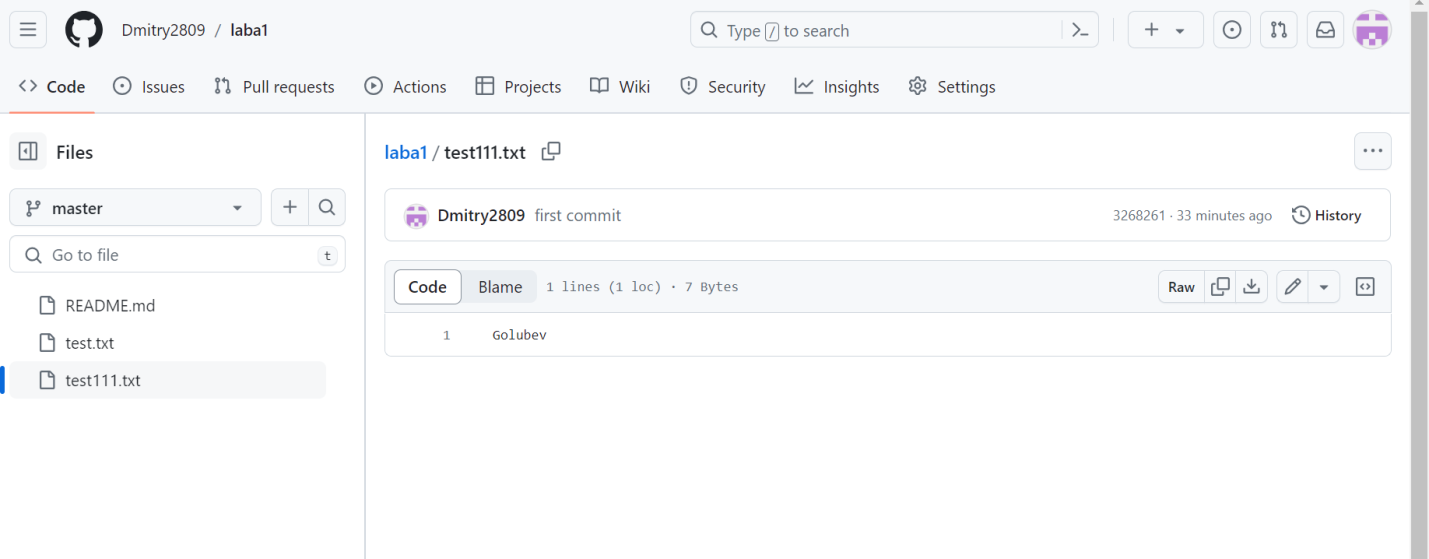
23) Добавьте в файл с именем отчество

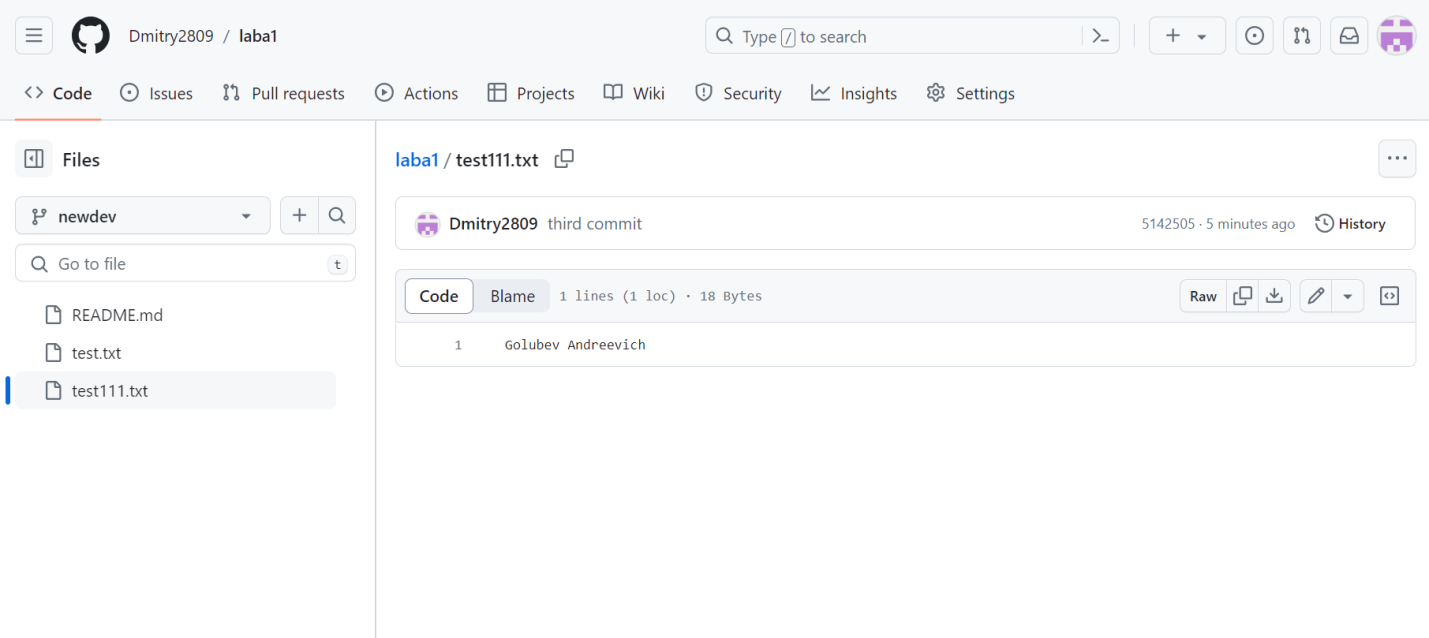
Зафиксируем изменения

Закомитим изменения



24) Попробуем переключиться обратно на ветку master и посмотреть файл: в нем только имя, отчество в другой ветке.

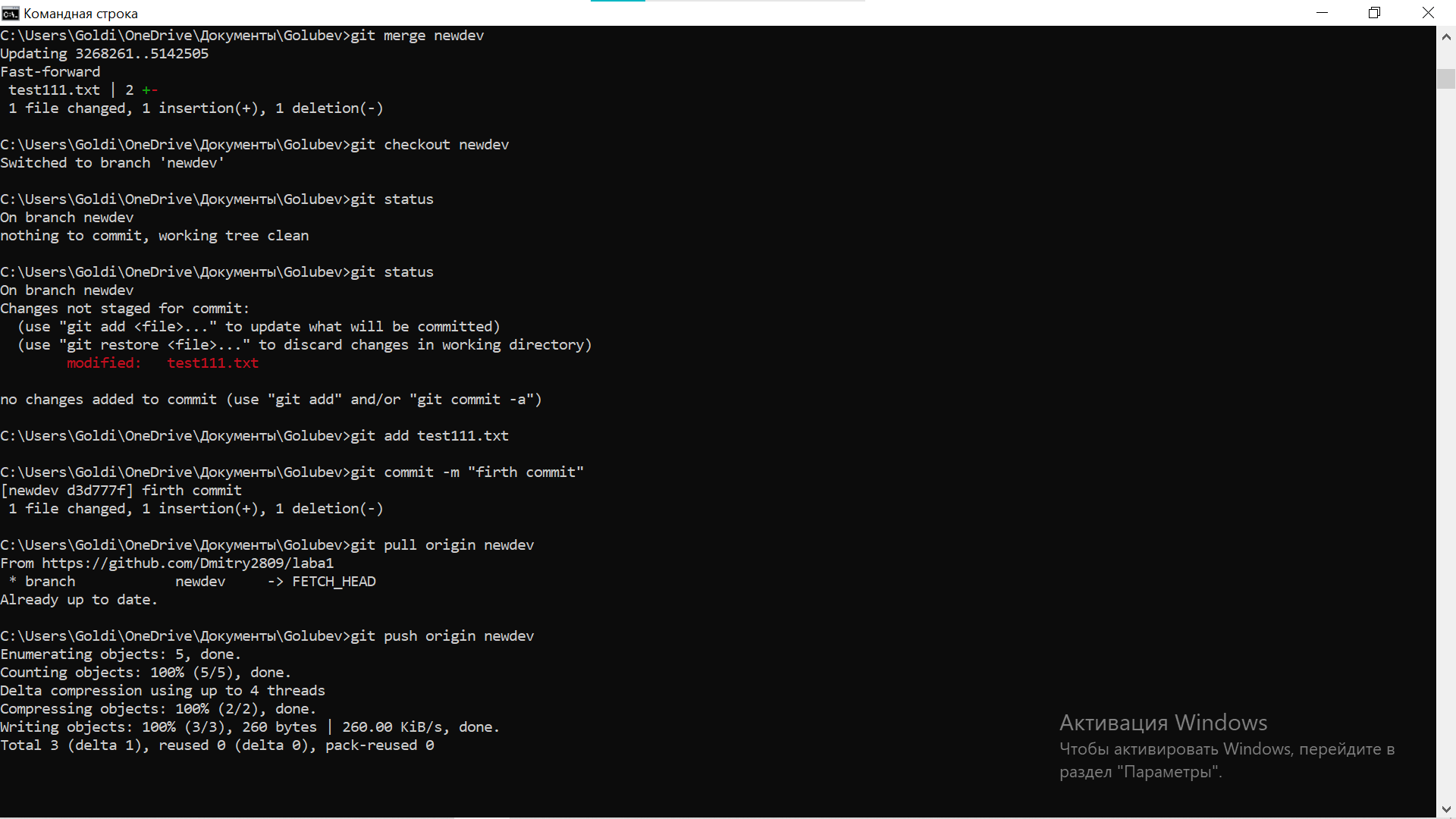


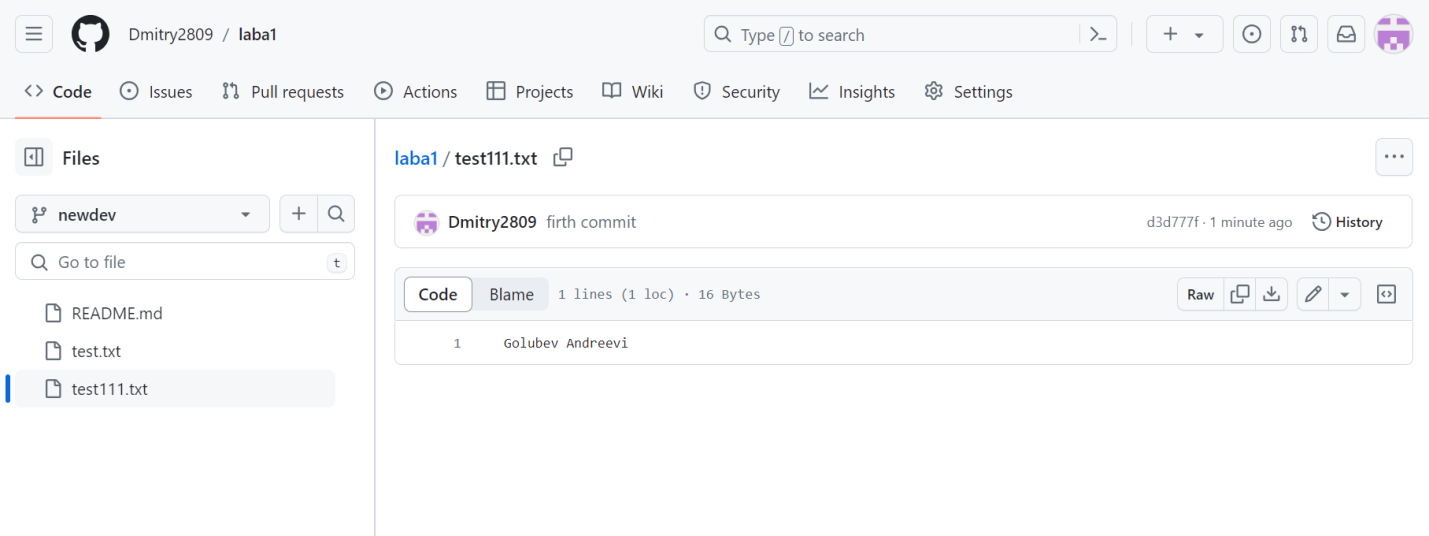


25) Объединим ветки git merge newdev

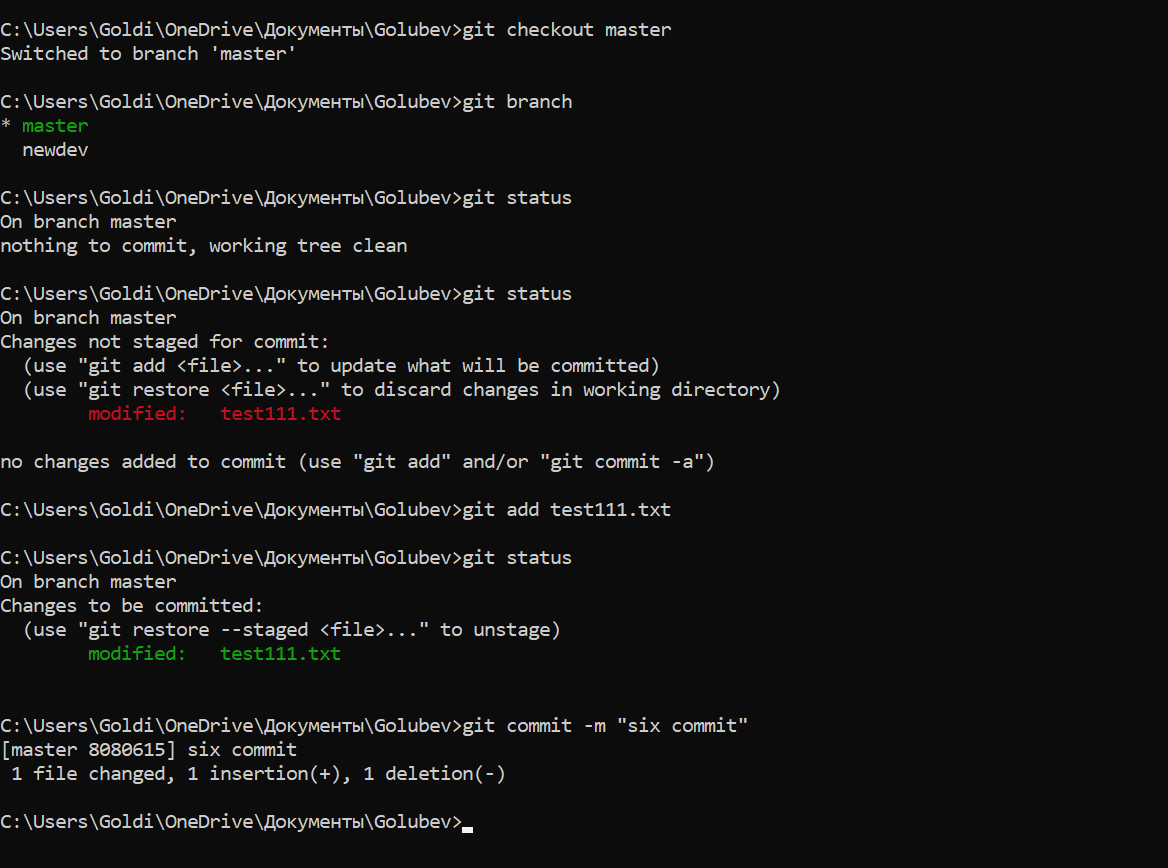
Откроем файл снова, там есть и имя, и отчество

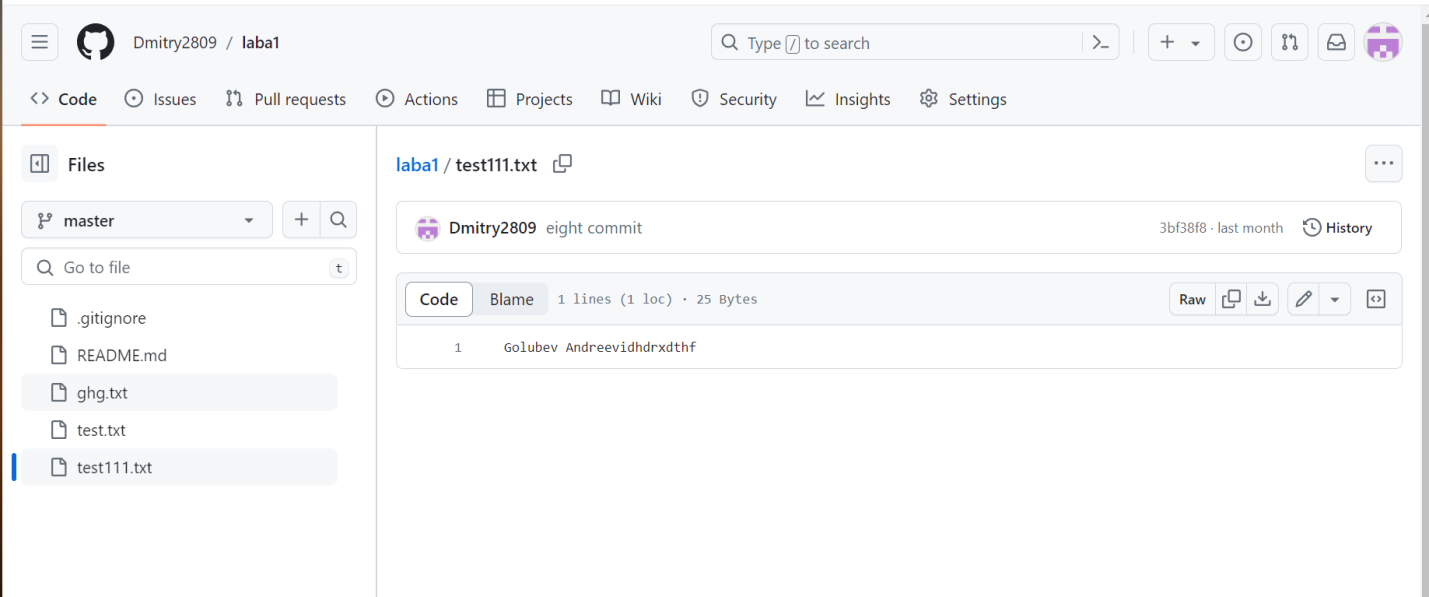
Переключимся на ветку newdev и удалим в отчестве несколько букв Зафиксируем, закоммитим.





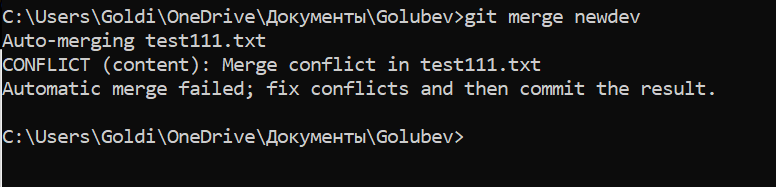
26) Переключимся на ветку master, добавим к отчеству несколько букв Зафиксируем, закоммитим.



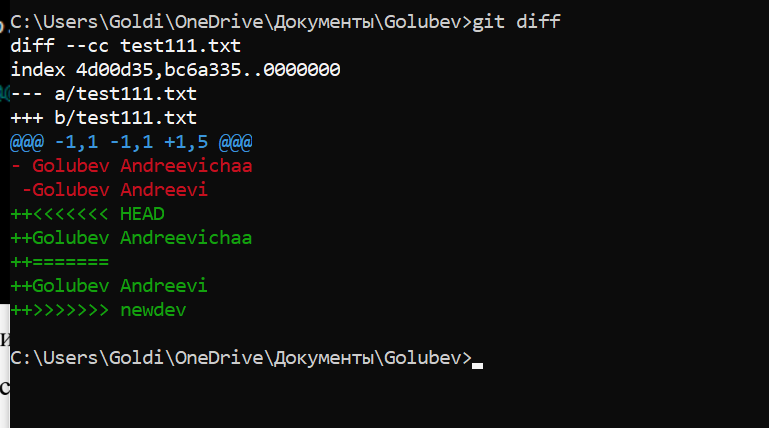


27) Объединим ветки git merge newdev

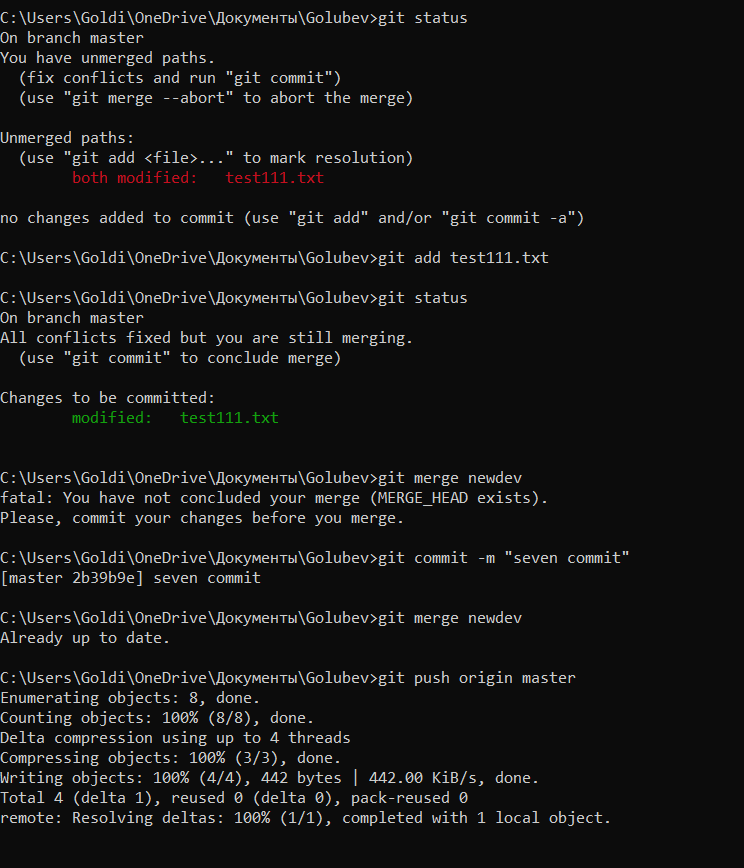
Теперь ничего не получится, т.к. есть изменения в обеих ветках



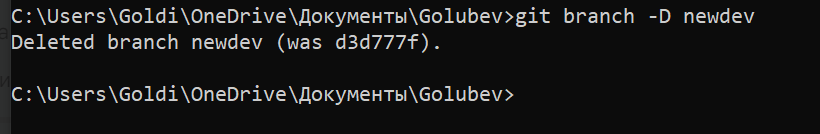
28) Наберем команду, для просмотра изменений git diff.



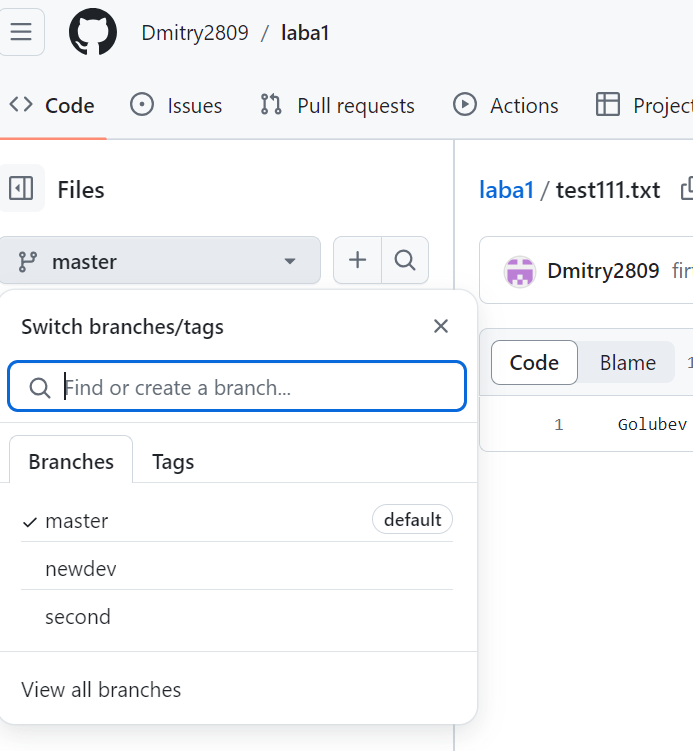
29) Выбираем нужную строку (какую строку оставить, может вообще все хотим оставить), удаляем разделители, сохраняем файл, фиксируем изменения, коммитим, отправляем на сервер



30) Удаляем ветку newdev.



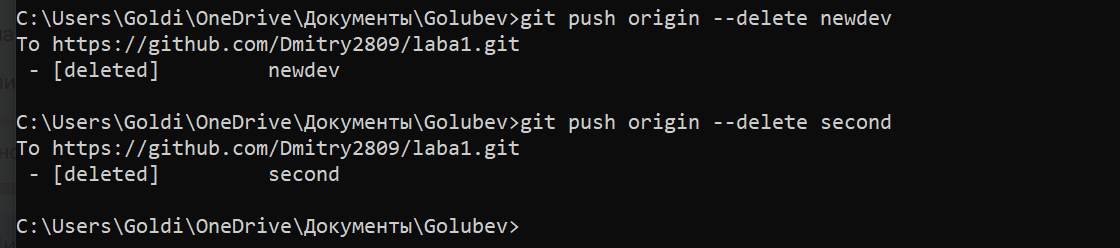
31) Кстати, если посмотреть на github, то наши ветки там остались:

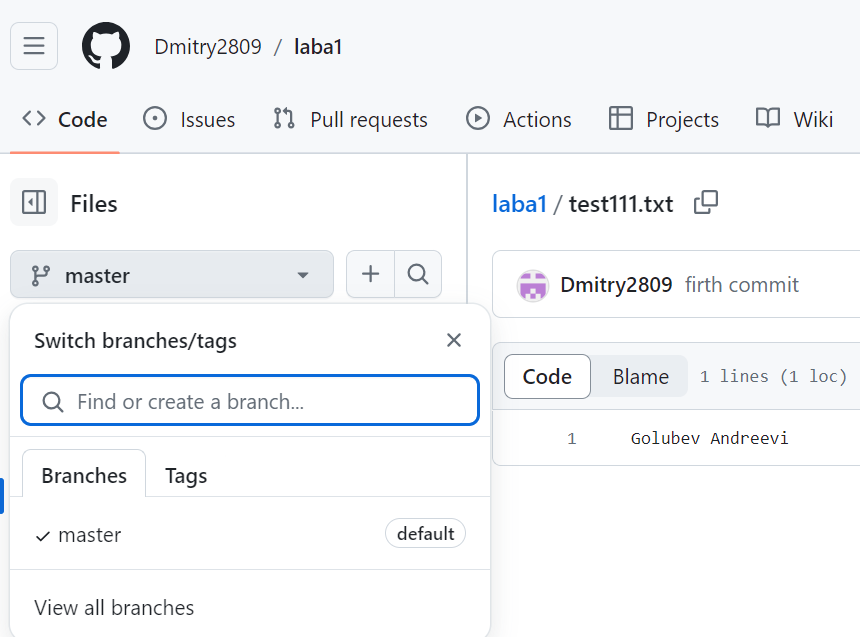


32) Давайте удалим их из github git

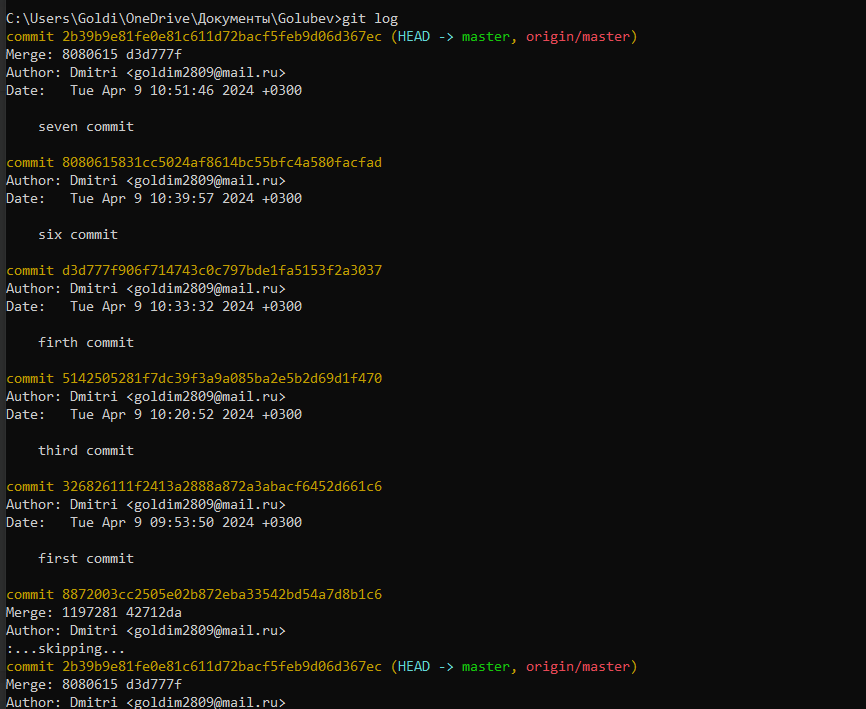
push origin --delete newdev

git push origin --delete second

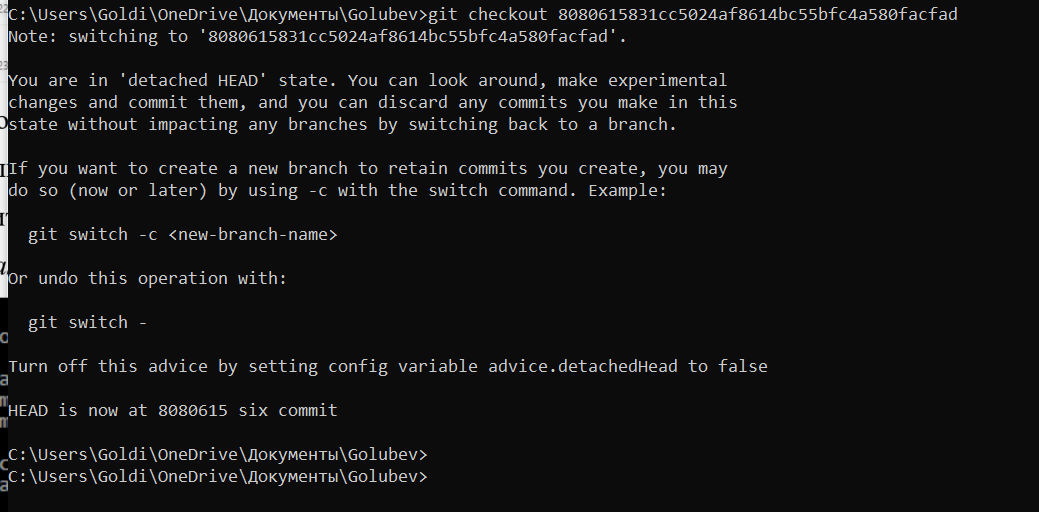




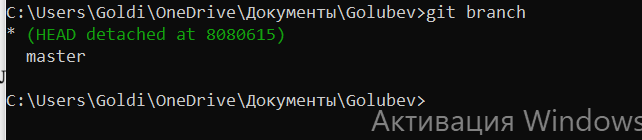
33) Чтобы посмотреть все комиты, можно использовать команду git log.



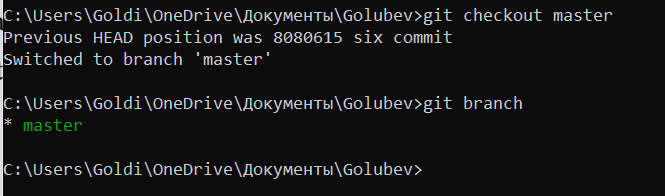
34) Выбираем любой коммит, на который хотим откатится, достаточно указать его первые несколько символов (перед выполнением посмотрите свои файлы, чтобы запомнить, что там находится) git checkout [hash commit].



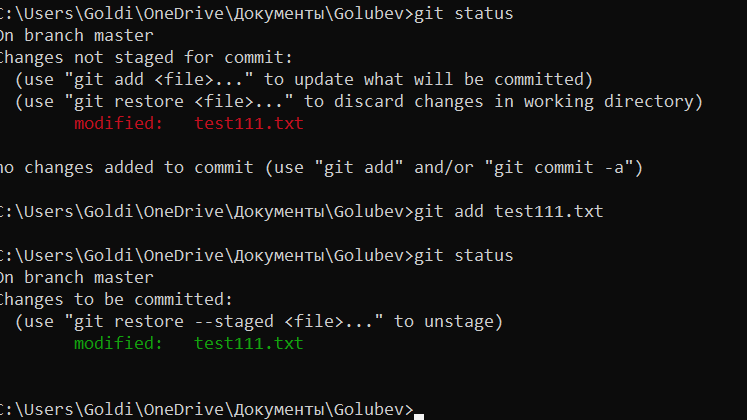
35) Чтобы это произошло создалась псевдо-ветка начинающаяся на этом коммите, посмотрим ветки и увидим:

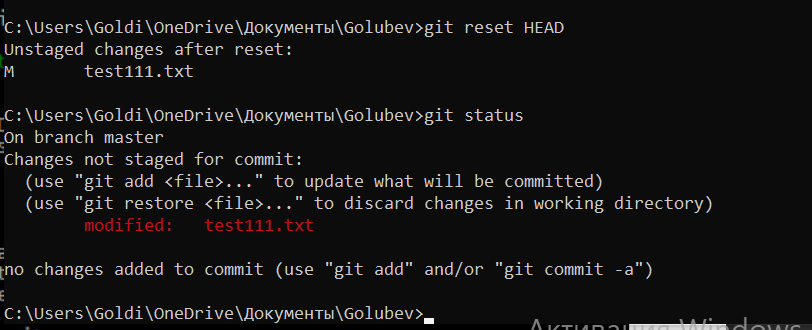


36) Откатимся обратно на master ветку.

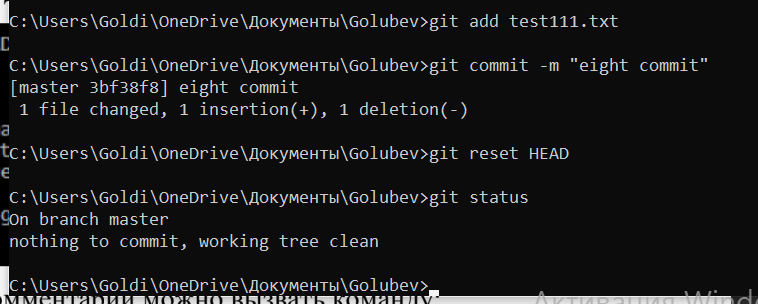


37) Возможно ситуация, что вы закрепили файлы (поменяйте что-то в файле и закрепите их, вызовите git status), но еще не коммитили, хотите убрать файлы из области закрепления, вызовим команду: git reset HEAD.

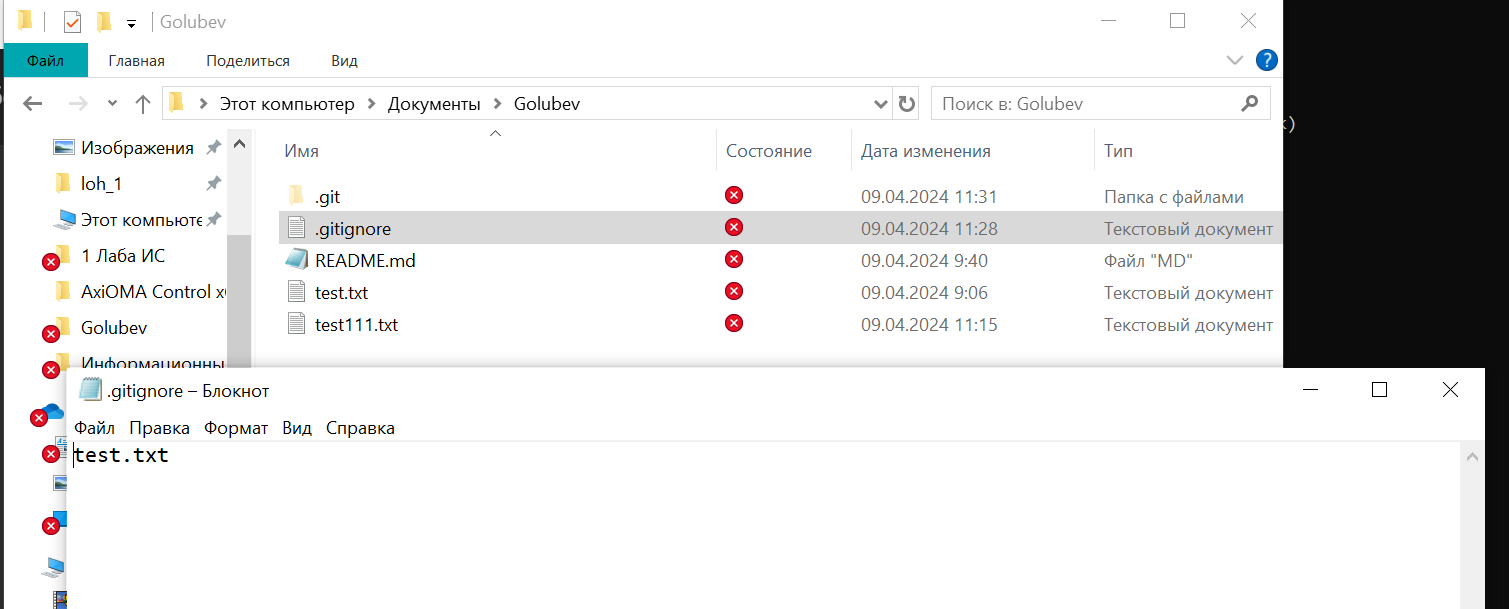




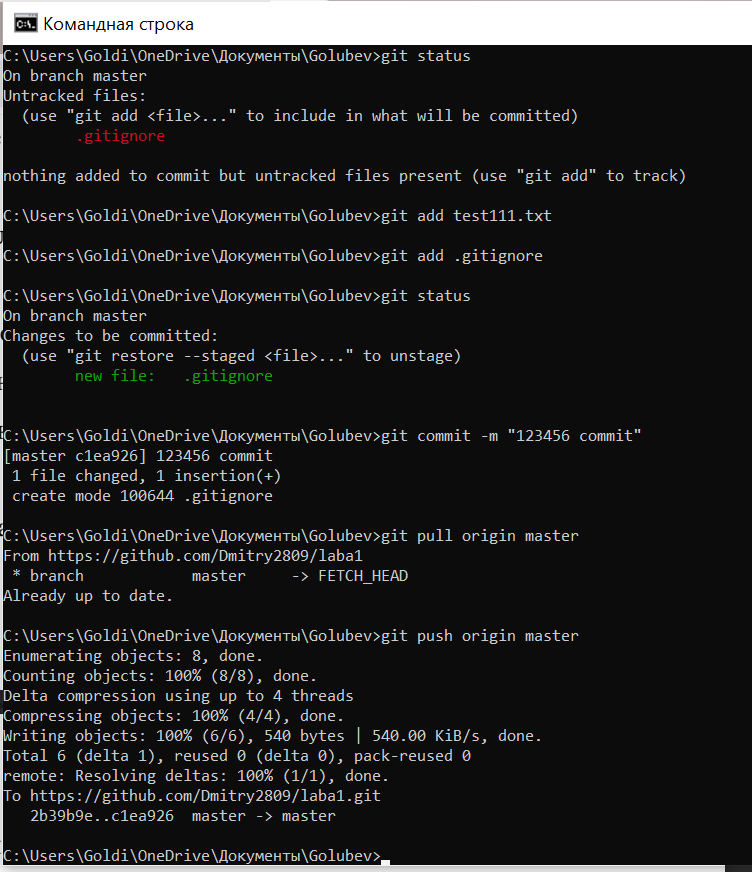
38) Теперь снова добавим в область закрепления и закомитим, но еще не отправляем на сервер, можно вызвать ту же команду и тогда все, что осталось незакомиченным будет удалено.

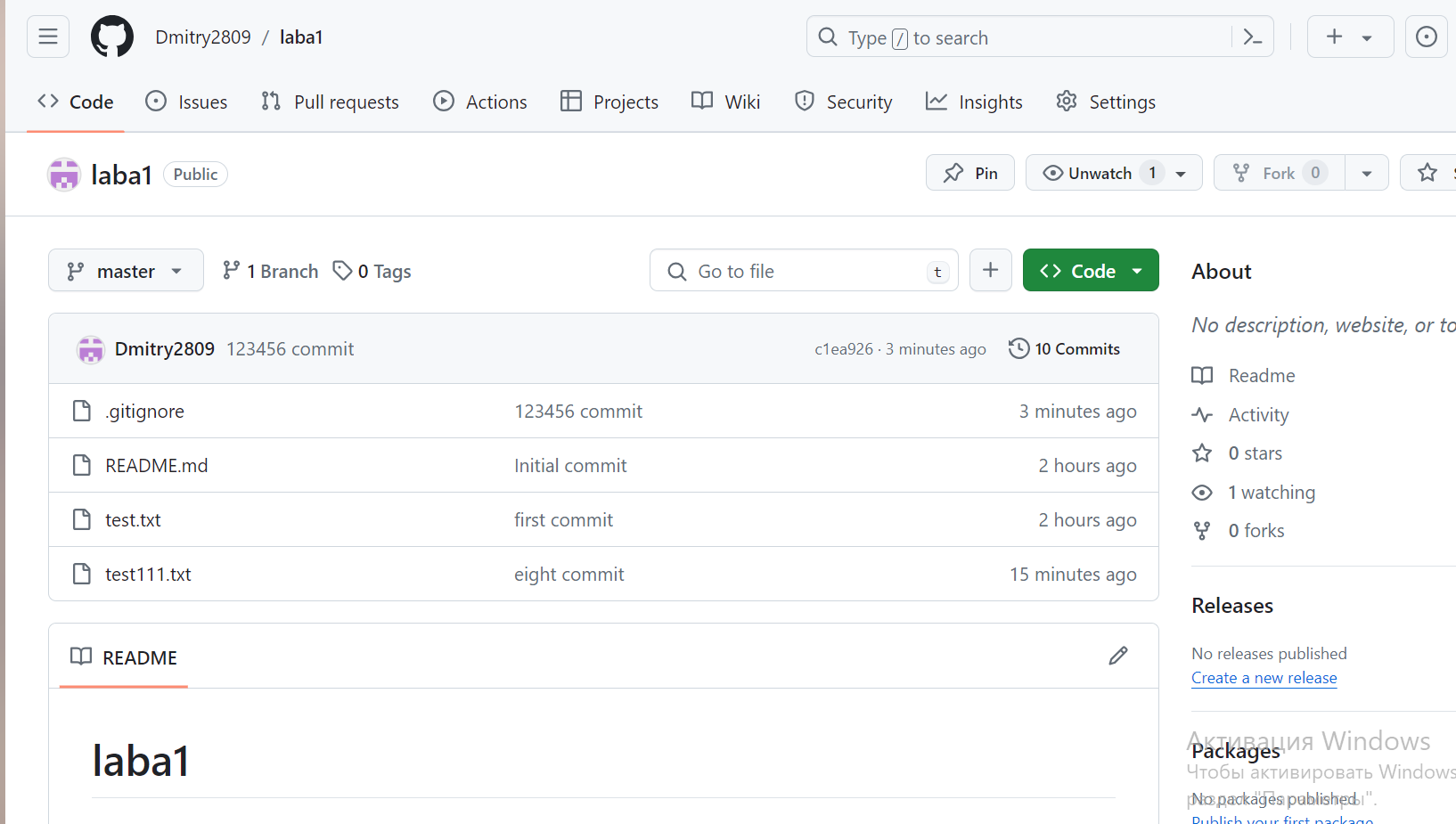


39) Создадим вручную файл под названием «.gitignore» и сохраним его в директорию проекта. Внутри файла перечислим названия файлов/папок, которые нужно игнорировать, каждый с новой строки.



40) Зафиксируем изменения, закоммитим, отправим на сервер.

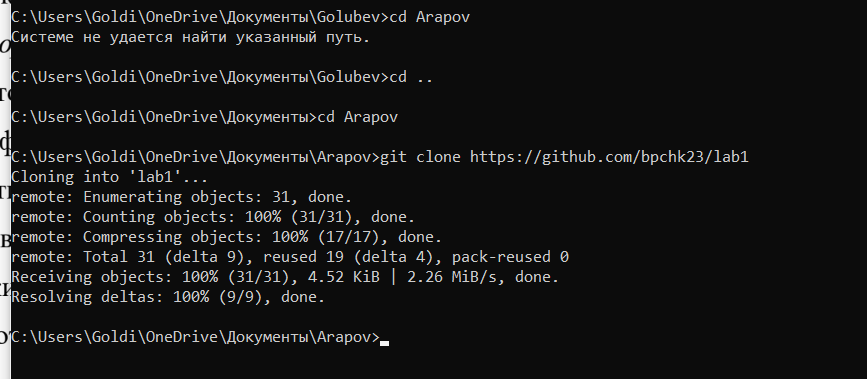




41) Совместная работа с git

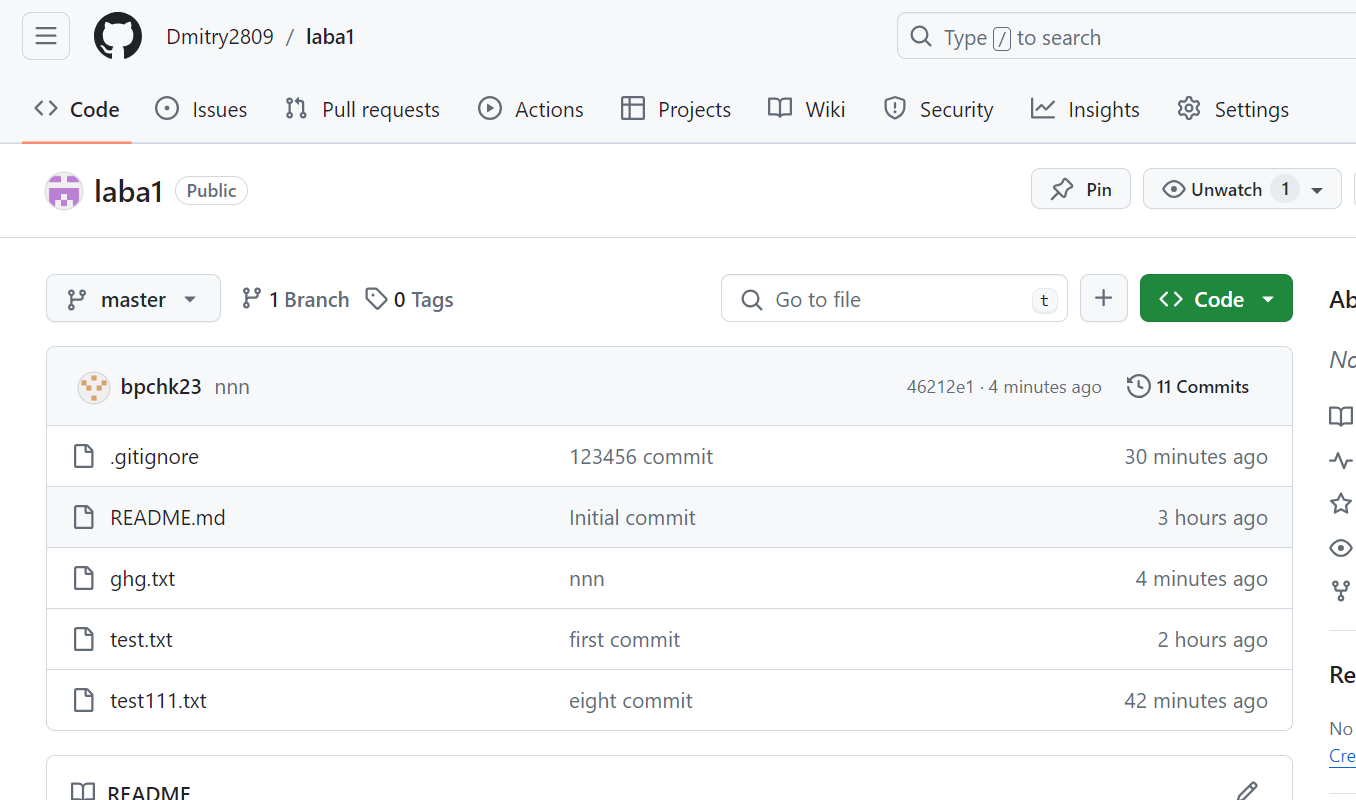
Выбираем себе напарника, скидываем друг другу адреса своих репозиториев. Переходим по полученным ссылкам на репозиторий и нажимаем fork.

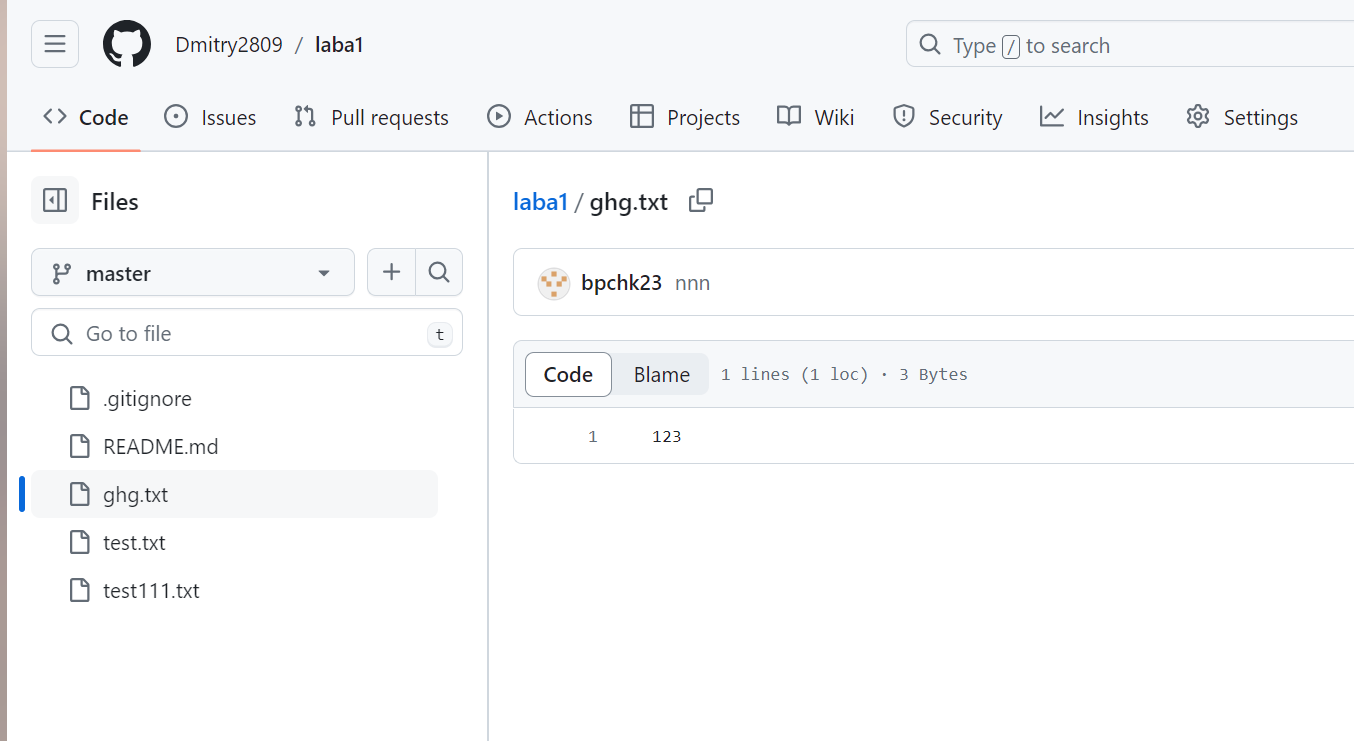
Далее через консоль клонируем этот репозитории на локальный компьютер, предварительно создаем папку, в которую будем клонировать и в консоли переходим в нее git clone «адрес репозитория».



42) Необходимо дать права доступа для записи в ваш репозиторий. Для этого переходим на github в раздел Settings, раздел Colloboration.

Попробуем теперь что-то поменять, закоммитим и отправим на сервер.





Вывод: Мы научились работать с системами контроля версий GitHub. Освоили такие навыки как изменение репозитория, закомиттивание и отправка файлов на сервер, а также объединение и удаление веток. И провели данные операции совместно с коллегами по учёбе.