

**МИНОБРНАУКИ РОССИИ**

**федеральное государственное бюджетное образовательное учреждение высшего образования**

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

# (ФГБОУ ВО «МГТУ «СТАНКИН»)

Институт цифровых Кафедра

интеллектуальных систем компьютерных систем управления

Дисциплина «Основы системного программного обеспечения»

**Отчет по лабораторной работе №\_1\_**

# Работа Git Hub

**Выполнил**

**студент гр. АДБ-20-08:** \_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_  **Кочуров В.А.**

*(дата) (подпись)*

**Проверил**

**к.т.н., доцент** \_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_ **Ковалев И.А.**

*(дата) (подпись)*

**Москва 2023 г.**

**Содержание**

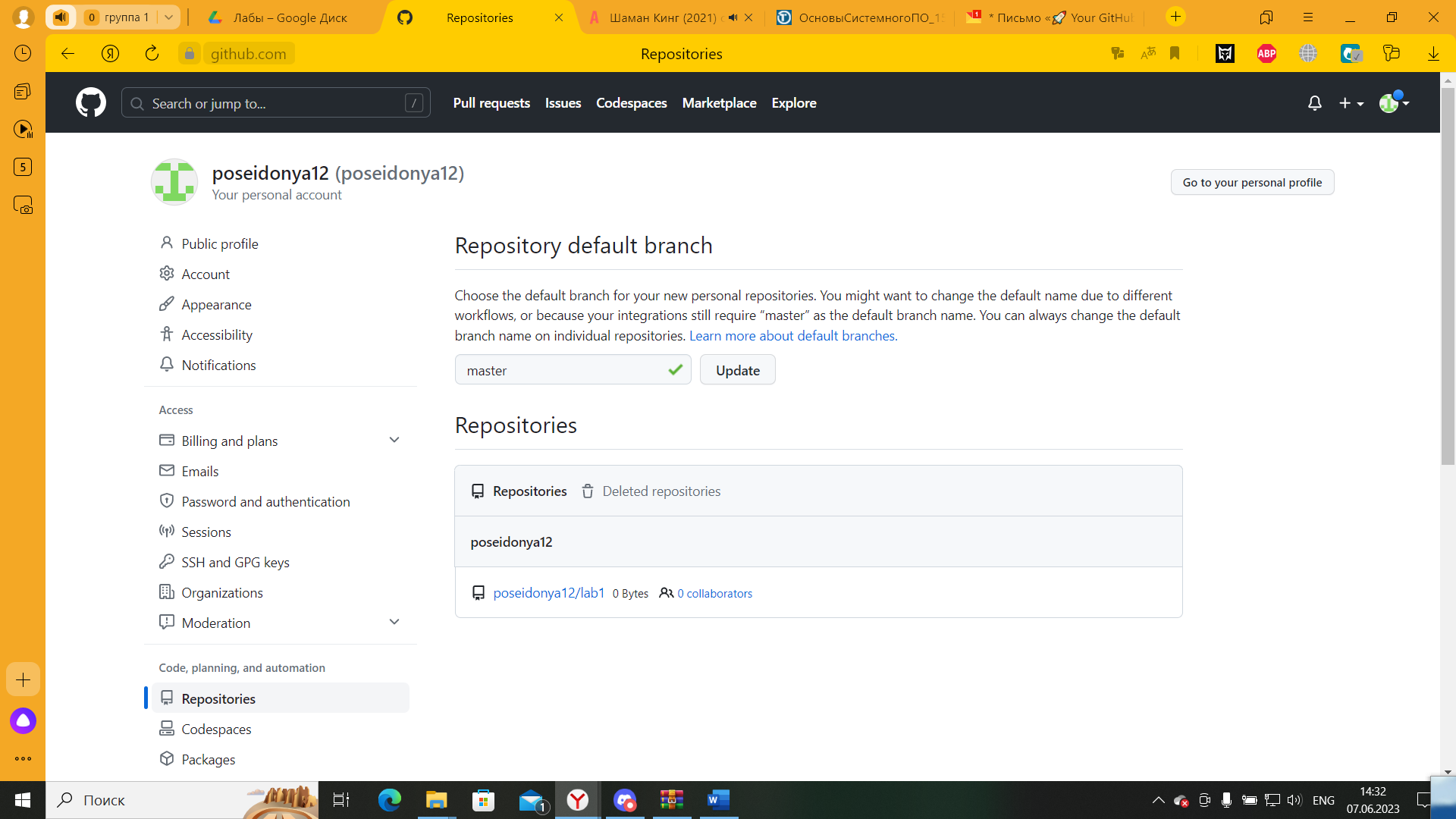
1. **Цель**
2. **Задачи работы**
3. **Описание и выполнение тестовых и индивидуальных заданий**

**1.Цель:** научиться работать с системой контроля версий Git Hub.

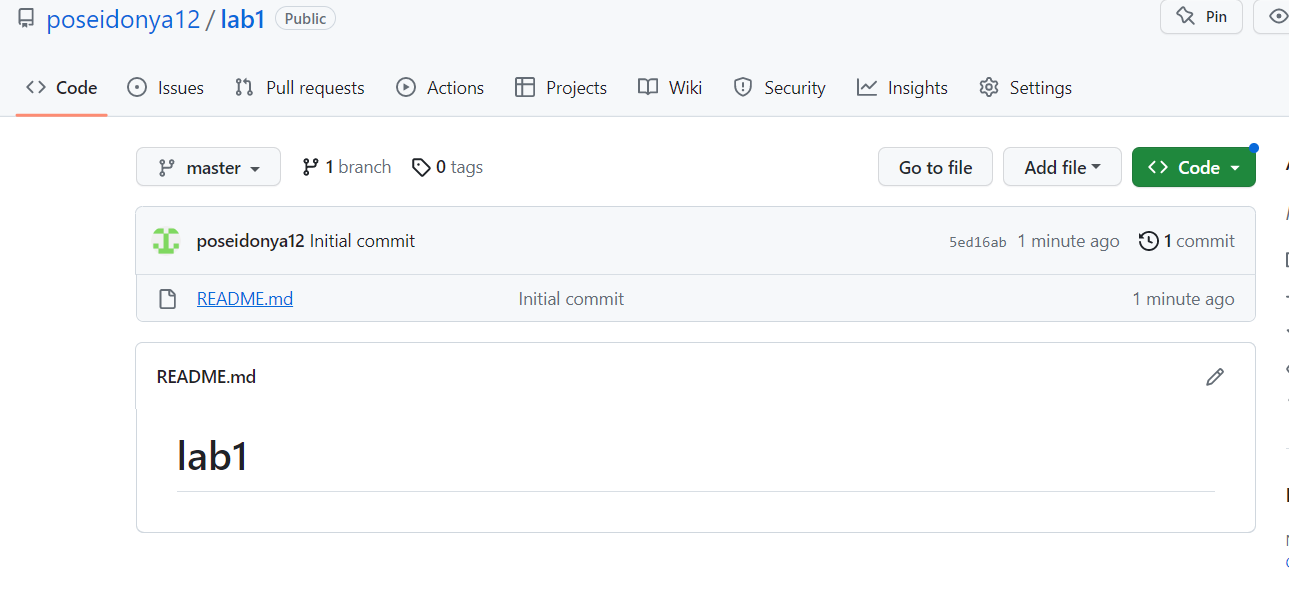
**2. Задачи работы:**

1. Создать учетную запись на Git Hub.
2. Создать репозиторий и научиться доgбавлять туда файлы.
3. Научиться отправлять изменения на сервер.

**3. Описание и выполнение тестовых и индивидуальных заданий** Создадим новый репозиторий LR1.

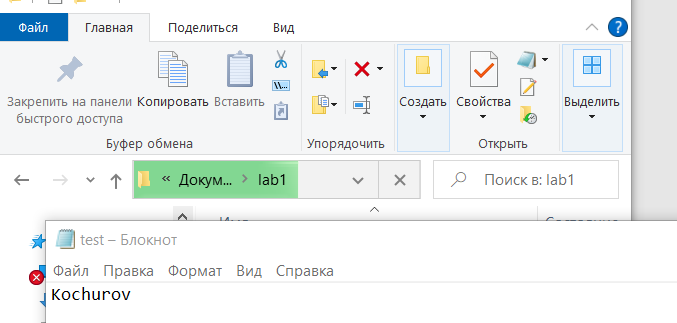


Увидим, что наш репозиторий создался.

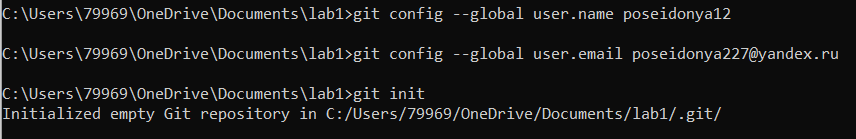


Создадим теперь локальную папку, которая будет связана с нашим облачным git хранилищем.

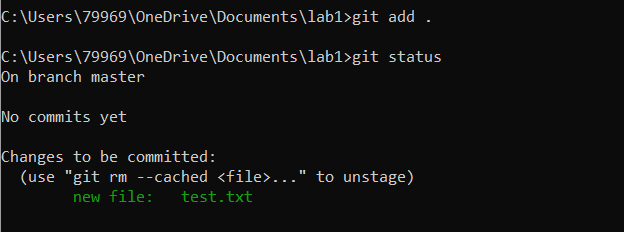
В ней создадим файлик test.txt с фамилией.



Настроим наше имя и адрес электронной почты. А также проинициализируем эту папку как git репозиторий.



Командная строка вернула нам сообщение, что проинициализирована пустая git директория. Теперь используя проводник windows создадим в своей папке любой текстовый файл и пропишем туда наше имя. Введем команду:

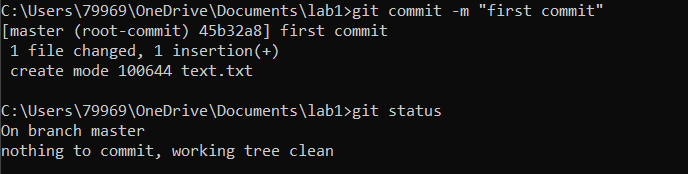


Нам отобразилось сообщение, что есть новый файл, но он не отслеживается. Это значит, что система еще не знает, нужно следить за изменениями в файле или его можно игнорировать. Для этого нужно добавить файлы командой add.

Выполнив эту команду посмотрим на статус репозитория и видим, что файл готов к коммиту.

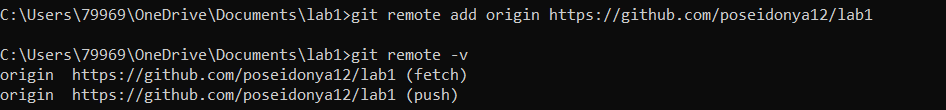
Теперь закомитим нащ файл.

Чтобы зафиксировать изменения, нам нужно хотя бы одно изменение в области подготовки.



Коммит создался. И после вызова git status видно, что файлов для коммитов больше нет.

Теперь свяжем наш локальный репозиторий с репозиторием на GitHub, для этого выполним следующее.



Мы указываем, что подключаемся к репозиторию по URI, так как проект может иметь несколько репозиториев одновременно нужно давать им разные имена, а также указать origin у главного репозитория.

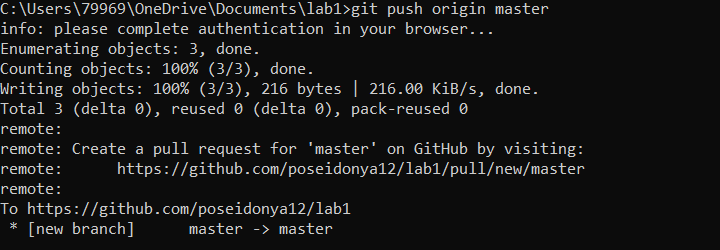
Затем просмотрим с помощью команды git remote –v к какому проекту мы подключены.

git pull origin master --allow-unrelated-histories – команда нужная, если у нас указано несколько репозиториев и у нас образовалось запутывание веток. В нашем случаем таких проблем не было. Так же можно удалить не правильный репозиторий командой git remote rm origin.

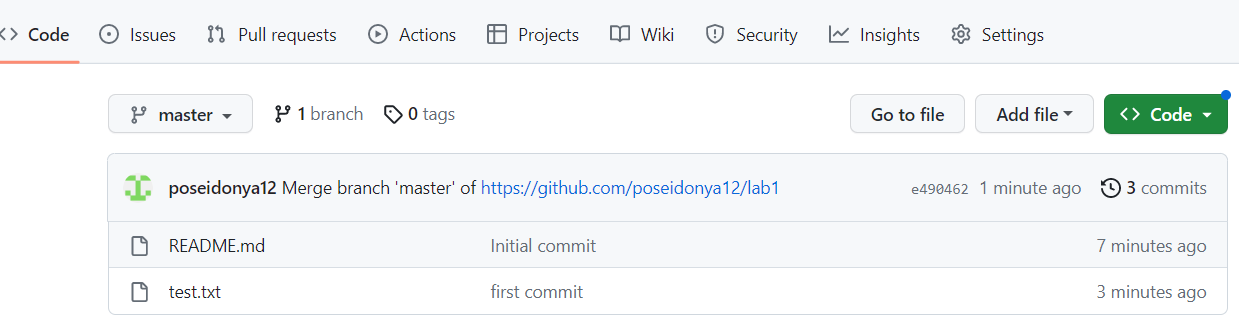
В нашем случае у нас есть файл и на сервере и локальном репозитории, поэтому нам вначале надо получить все данные с нашего репозитория, а потом уже их заливать на сервер нашими локальными данными.

Мы сделали изменения в нашем репозитории, теперь другие пользователи могут скачать изменения при помощи команды pull. git pull origin master

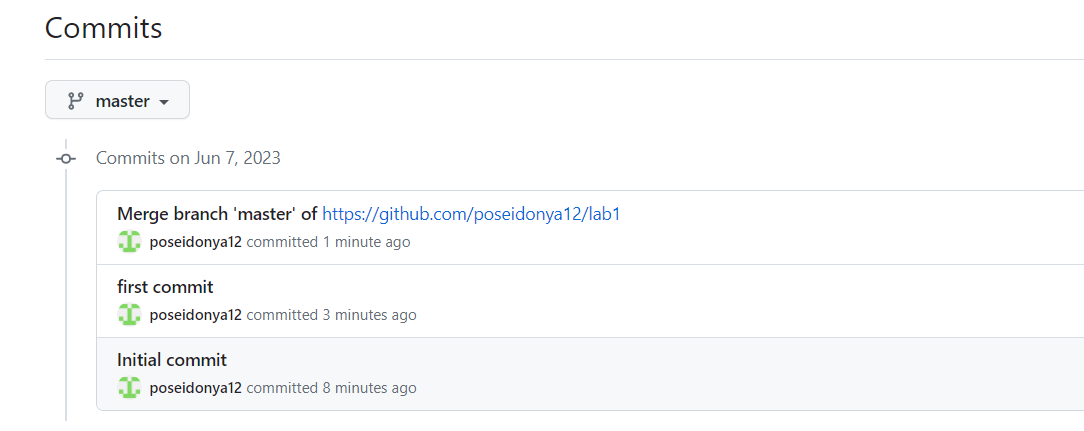
Теперь перешлем локальный коммит на сервер. Для этого существует команда push. Она принимает два параметра: имя удаленного репозитория (мы назвали нашу ветку origin) и ветку, в которую необходимо внести изменения (master – это ветка по умолчанию для всех репозиториев).



Видим, что изменения отобразились и на нашем сервере.

 Если нажмем на

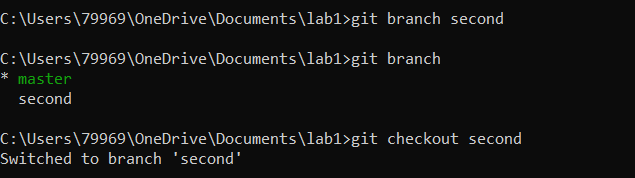
«commit», то увидим все наши изменения.



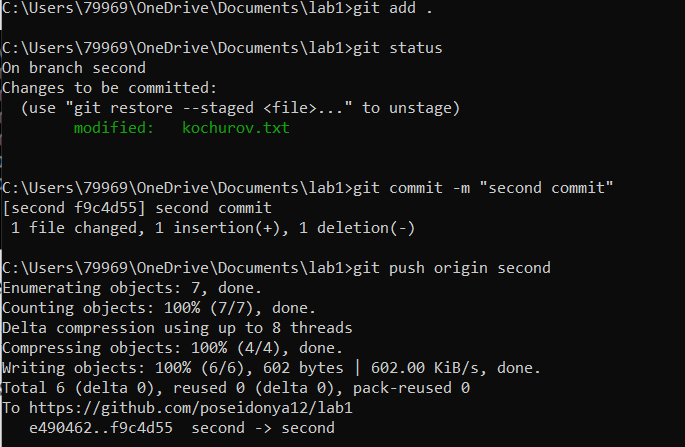
Теперь создадим новую ветку.

Ветка master создается сама, а дополнительные, мы создаем по своему желанию, с помощью команды git branch.

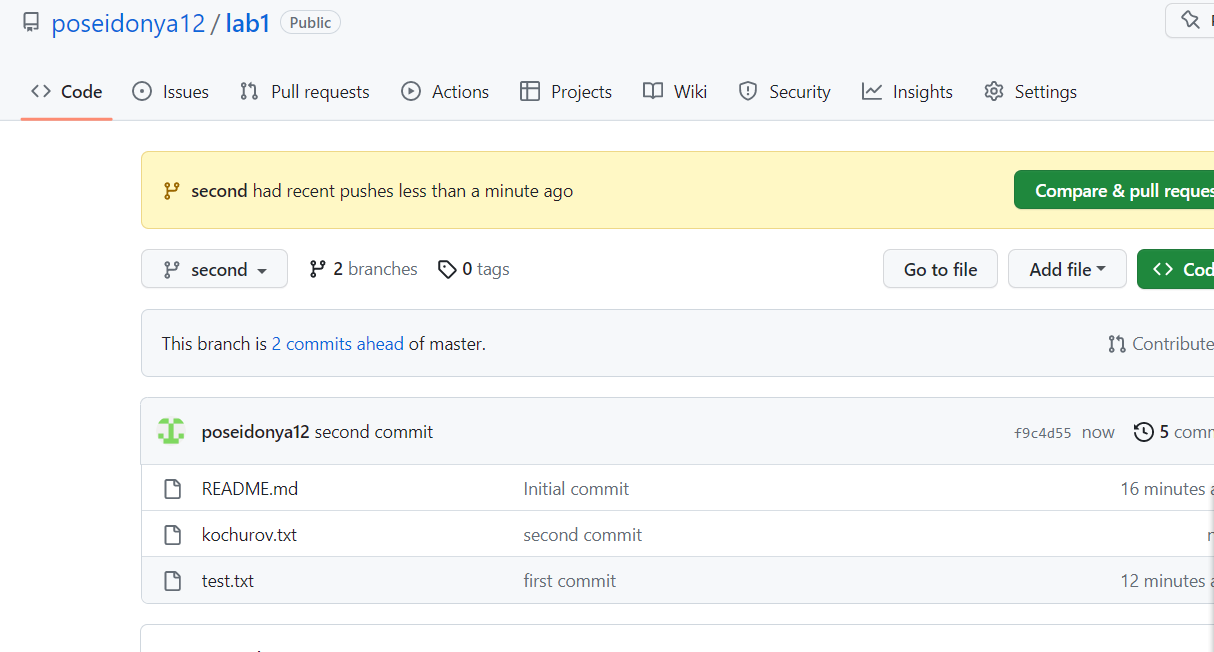
Теперь переключимся на другую ветку. Это мы делаем с помощью команды checkout



Создаем в нашем локальном репозитории kochurov.txt и напишем там свою фамилию, затем добавляем в области подготовленных файлов, закомитем и отправим на сервер.

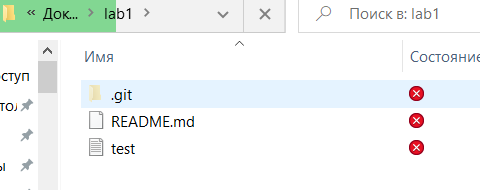


Зайдем на Git Hub и посмотрим, что в ветке master нет нашего файла, а в ветке second он присутствует.

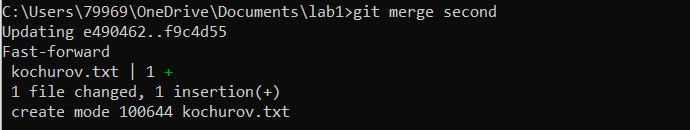


Переключимся снова на ветку master. Теперь если через проводник windows зайти в папку с проектом, то файла kochurov.txt, там не будет видно.





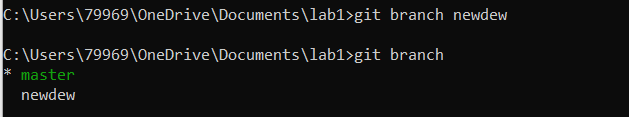
Чтобы он появился, нужно воспользоваться merge для объединения веток.



Теперь удалим ветку second.



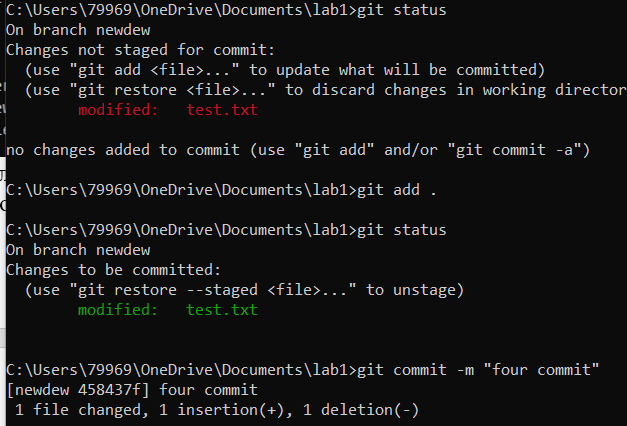
Создадим ветку, в которой будут одинаковые файлы.



Переключимся на неё



Добавим в файл изменения, а, то есть имя и фамилию и закомитим их.



Переключимся на ветку master и посмотрим файл, в нем только имя отчество в другой ветке. Теперь объединим ветки

Откроем файл снова, там есть и имя, и отчество

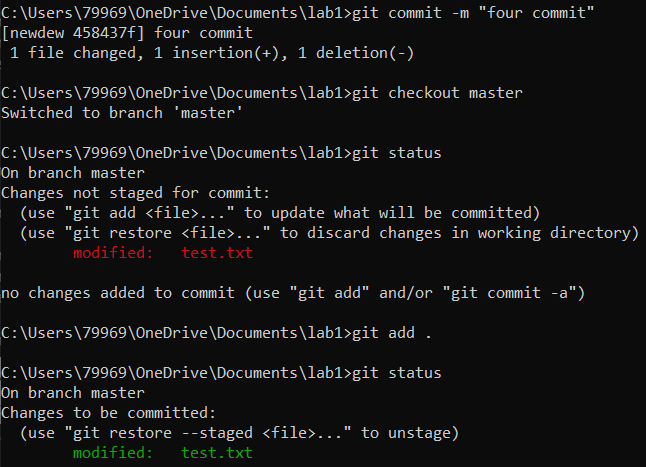
Переключимся на ветку newdev и удалим в имени несколько букв

Зафиксируем, закоммитем

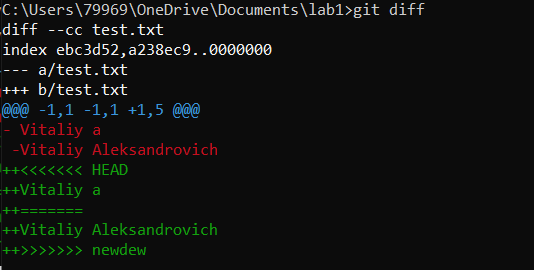
Переключимся на ветку master, добавим к отчеству несколько букв

Зафиксируем, закоммитем

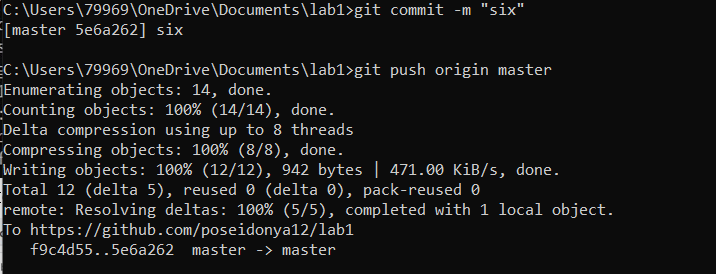
Объединим ветки



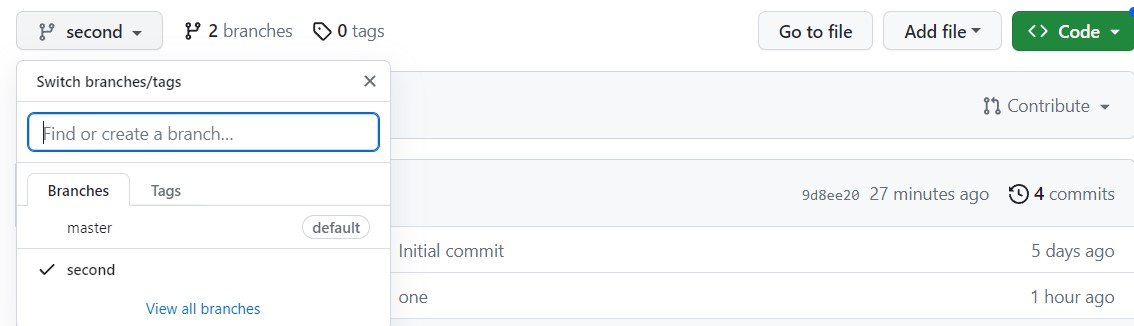
Теперь ничего не получится, так как изменения есть в обеих ветках. Разрешим конфликт.



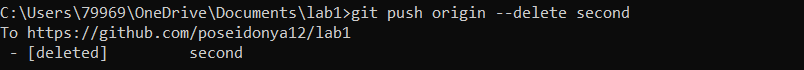
Исправим ошибку вручную, удалив разделители и выбрав нужный нам вариант. Сохраняем файл, фиксируем изменения, коммитим, отправляем на сервер. И удаляем ветку newdev.



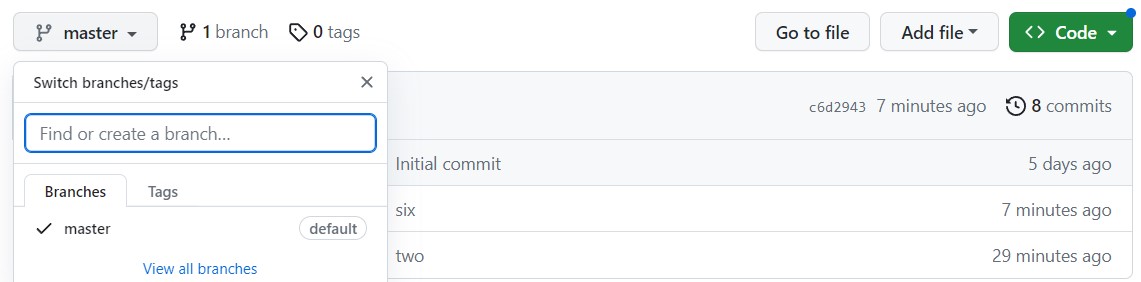
Если посмотрим на Git Hub, то наши ветки там остались.



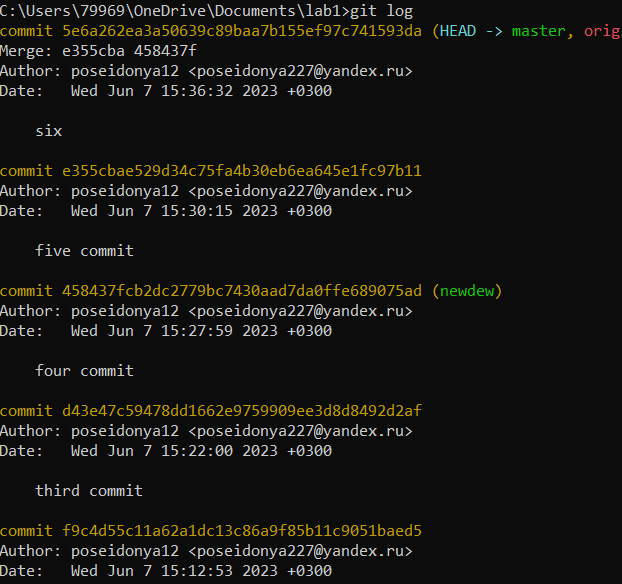
Удалим их.



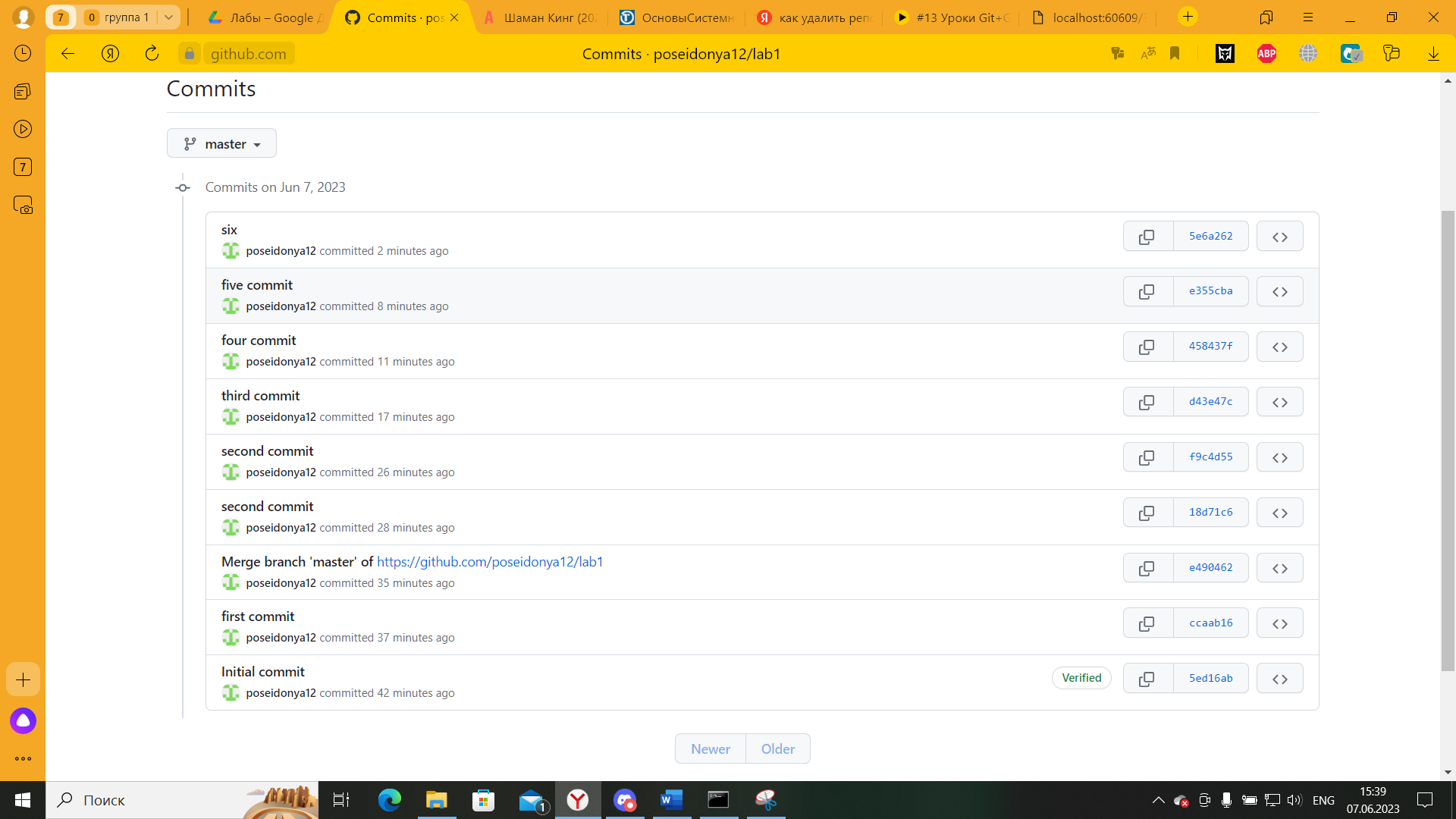
Посмотрим изменения.



Также Git позволяет вернуть выбранный файл к состоянию на момент определенного коммита. Это делается командой checkout. Для того, чтобы просмотреть все наши комиты используем команду git log

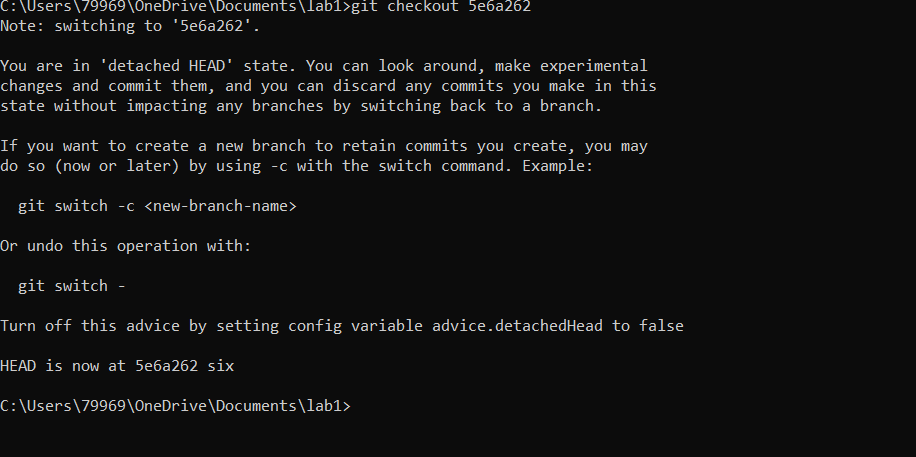


Или можно использовать github.



Выберем любой коммит, на который хотим откатится, указав его первые несколько символов.

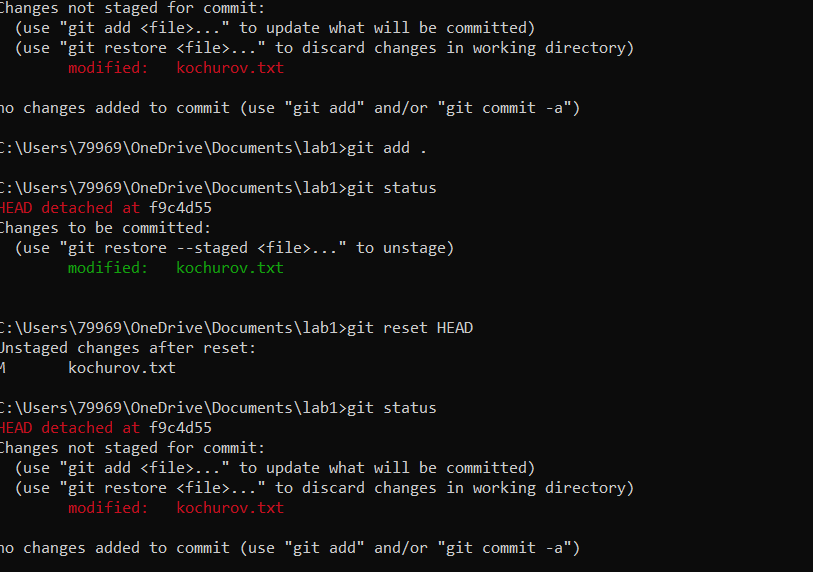
Git



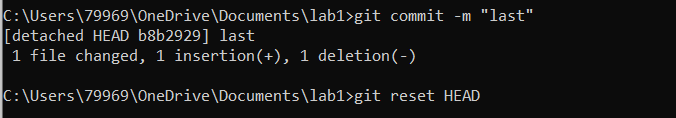
Если посмотреть файлы теперь, то видно, что они поменялись – мы вернулись назад.

Чтобы это произошло создалась псевдо-ветка начинающаяся на этом коммите. Дальнейший правильный вариант заключается в том, что нужно продолжить работать на новой ветке, а затем смерживаться.

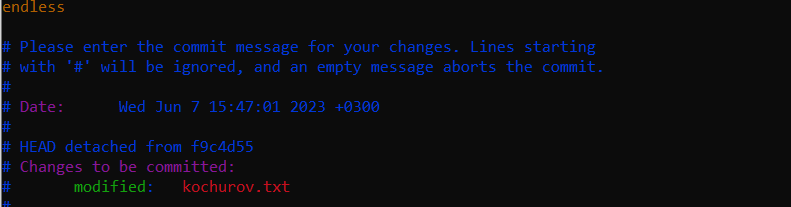
Рассмотрим ситуацию, если мы закрепим файлы, но еще не комитили и хотим убрать их из области закрепления.



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



Если хотим изменить комментарий можно вызвать команду: git commit – amend

Откроется консоле в редакторе Vi. 

В большинстве проектов есть файлы или целые директории, в которые мы не хотим (и, скорее всего, не захотим) коммитить. Создадим файл .gitignore и сохраним его в директорию проекта.

И перечислим файлы которые хотим игнорировать.

