

**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\_**

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

**Москва 2024 г.**

Оглавление

[Цель 3](#_Toc168260379)

[Задачи работы: 3](#_Toc168260380)

[Просмотр изменений 5](#_Toc168260381)

[Совместная работа с git 15](#_Toc168260382)

[Вывод 15](#_Toc168260383)

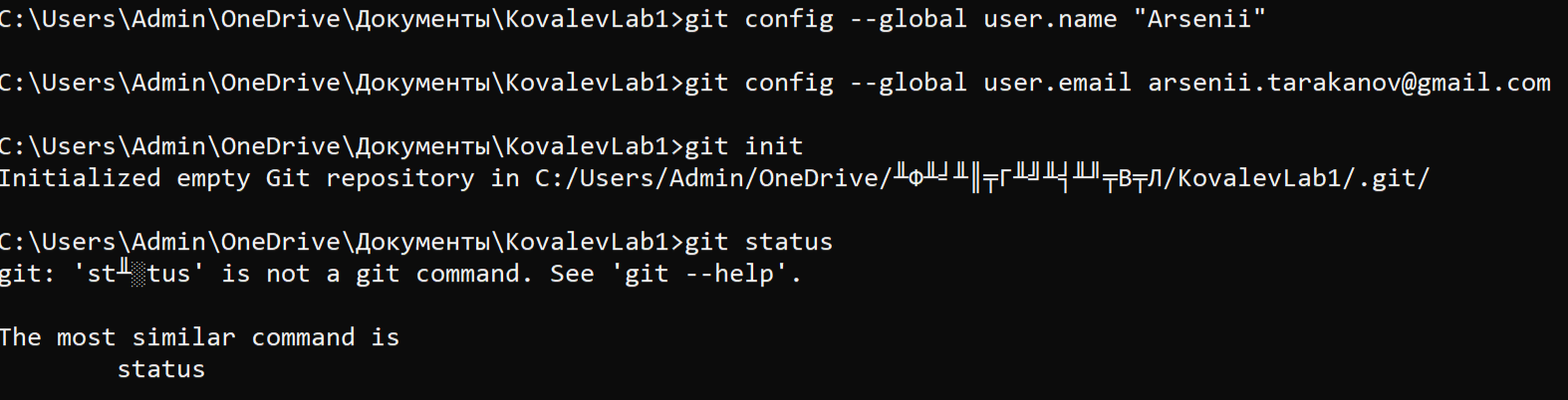
Цель**:** Ознакомиться с системой контроля версий Git, изучив ее основные понятия, локальные операции, а также принципы работы с репозиториями.

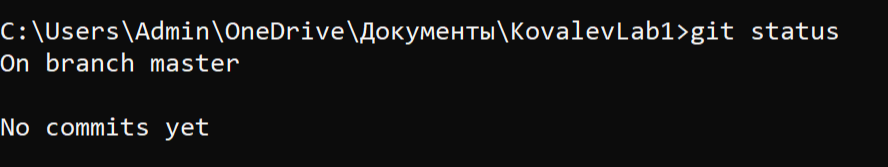
# **Задачи работы**:

1. Изучить основные понятия системы контроля версий Git: локальный репозиторий, рабочий каталог, область подготовки, коммит, индекс, хеш-сумма.

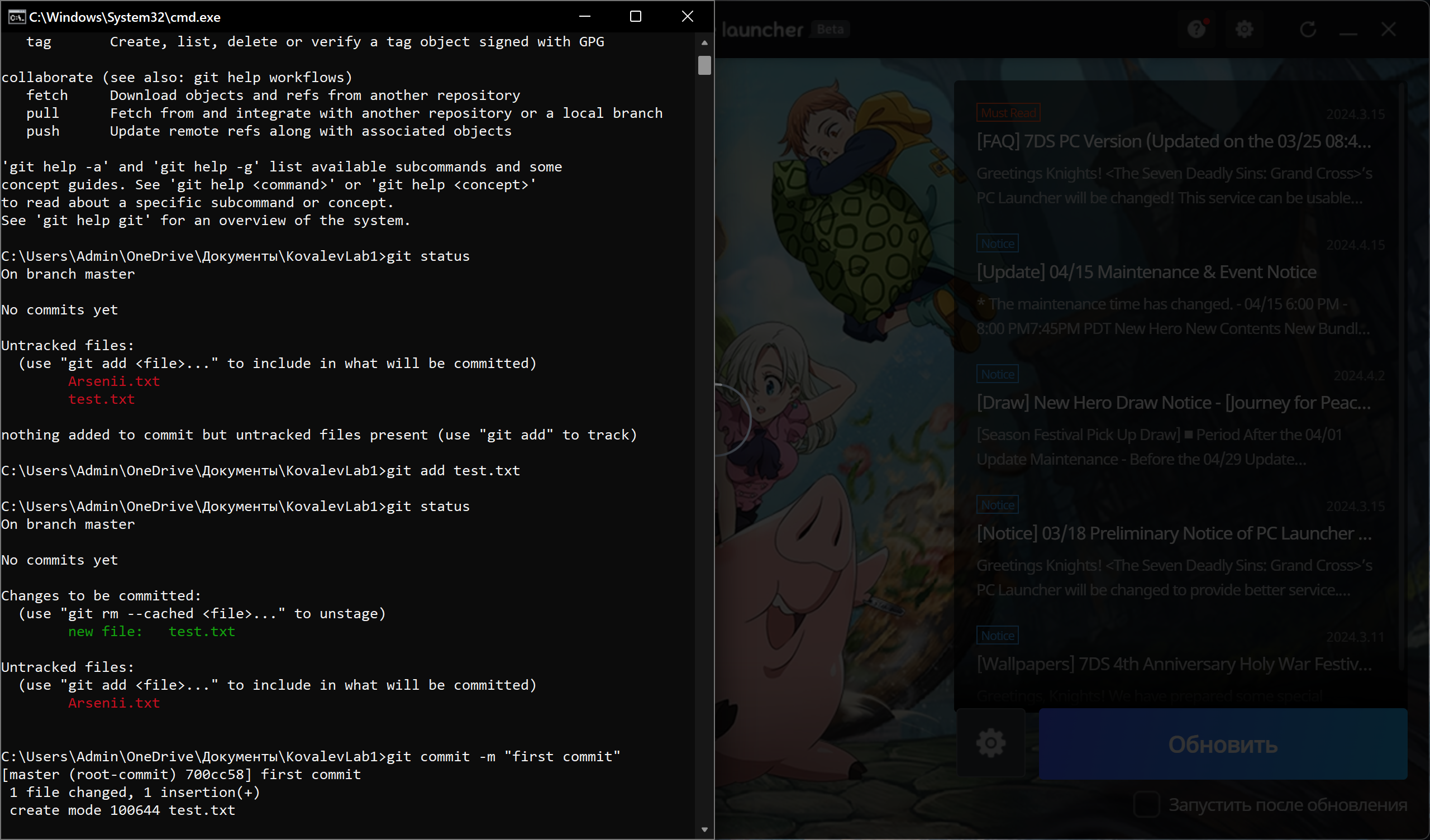
1. Научиться выполнять базовые локальные операции Git:
   1. git init - инициализация нового репозитория
   2. git add - добавление файлов в область подготовки
   3. git commit - создание коммита
   4. git status - отображение статуса файлов
   5. git log - просмотр истории коммитов
   6. git diff - отображение изменений файлов
   7. git reset - откат изменений
2. Научиться работать с репозиториями:
   1. git clone - клонирование репозитория
   2. git pull - обновление локального репозитория
   3. git push - отправка изменений в удаленный репозиторий

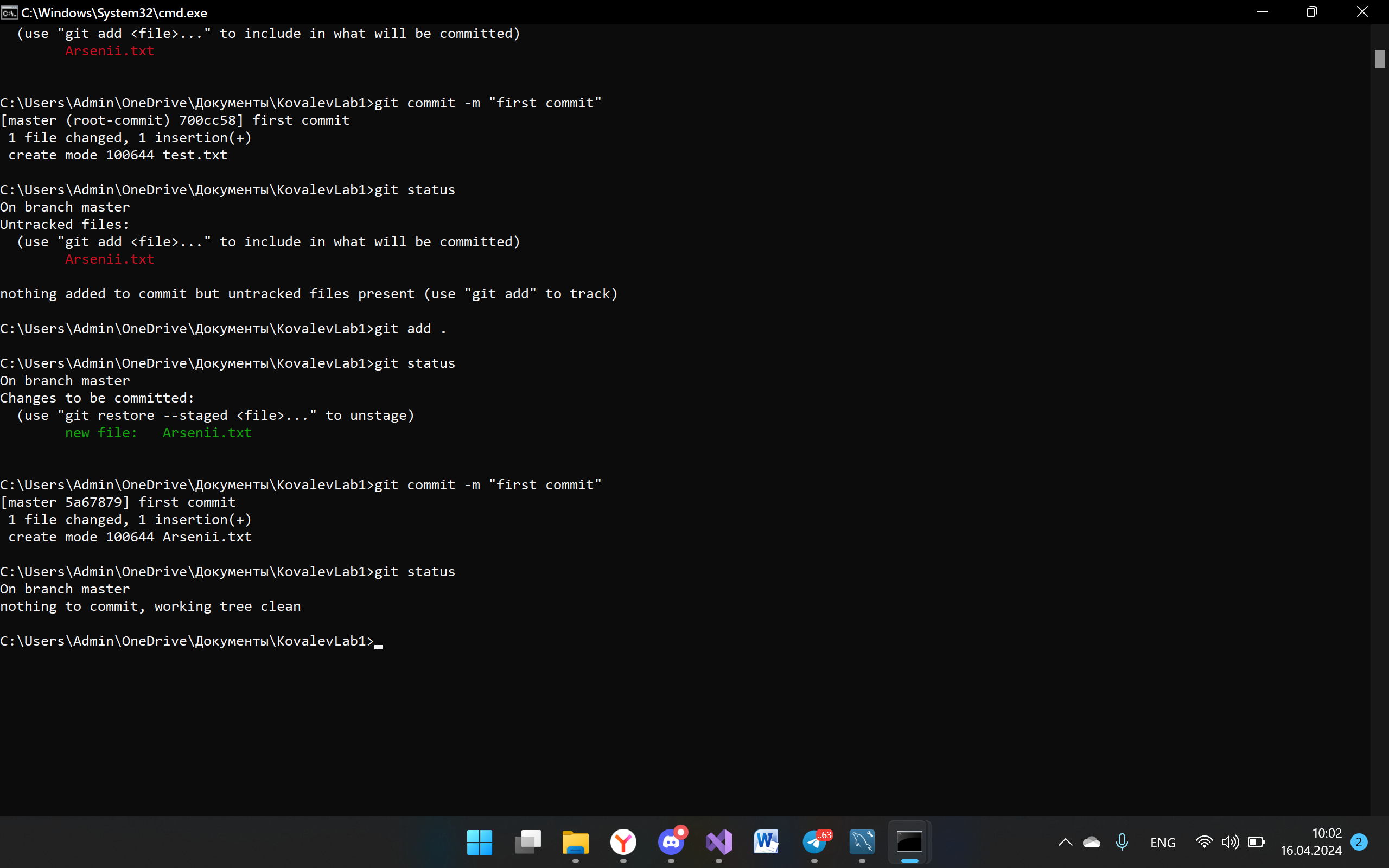
Настроим имя пользователя и адрес электронной почты. Перейдем в созданную папку в командной строке и выполним команду "git init" для инициализации папки как git репозитория. Создаем текстовый файл и выполняем команду "git status" в командной строке, чтобы увидеть, что файл не отслеживается.



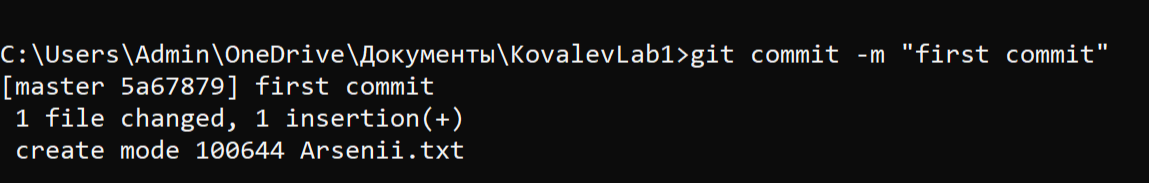


Чтобы начать отслеживать файл, используем команду "git add <имя файла>".

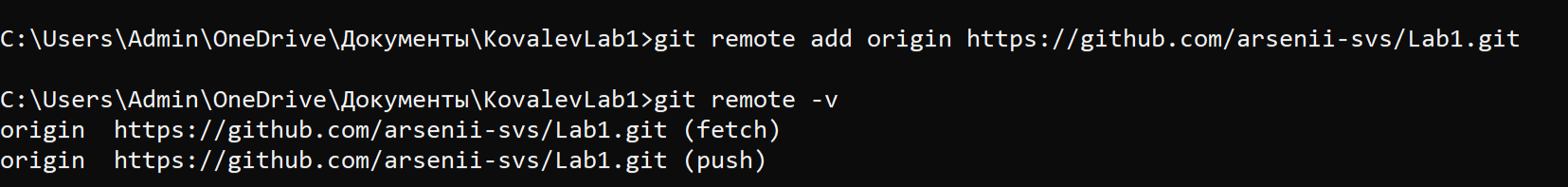




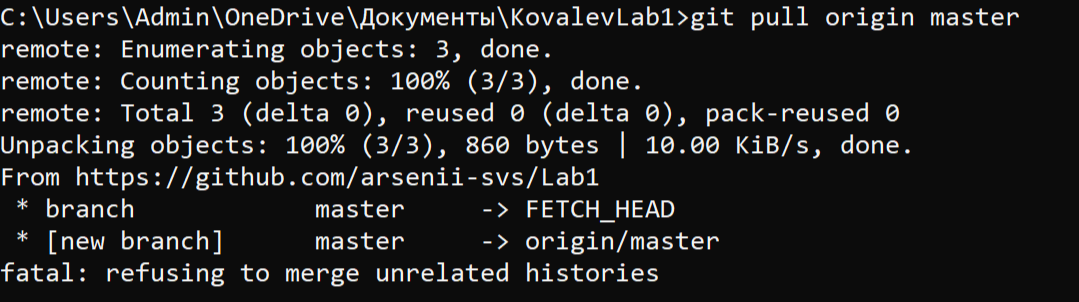
Теперь закомитим наши файлы. Коммит представляет собой состояние репозитория в определенный момент времени как некий опечаток во времени. Коммит отмечается hash суммой и к которому мы можем в любой момент времени вернуться.



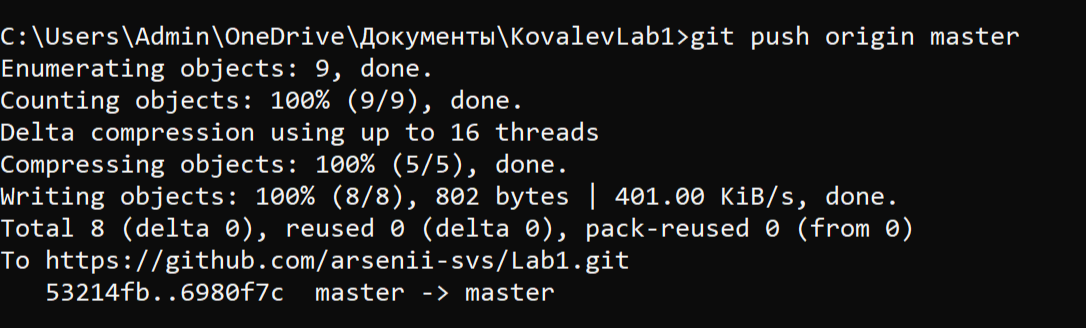
Чтобы связать наш локальный репозиторий с репозиторием на GitHub, выполним следующую команду *git remote add origin* в терминале. Далее вызовем команду, для просмотра, к какому проекту мы подключены git remote –v.



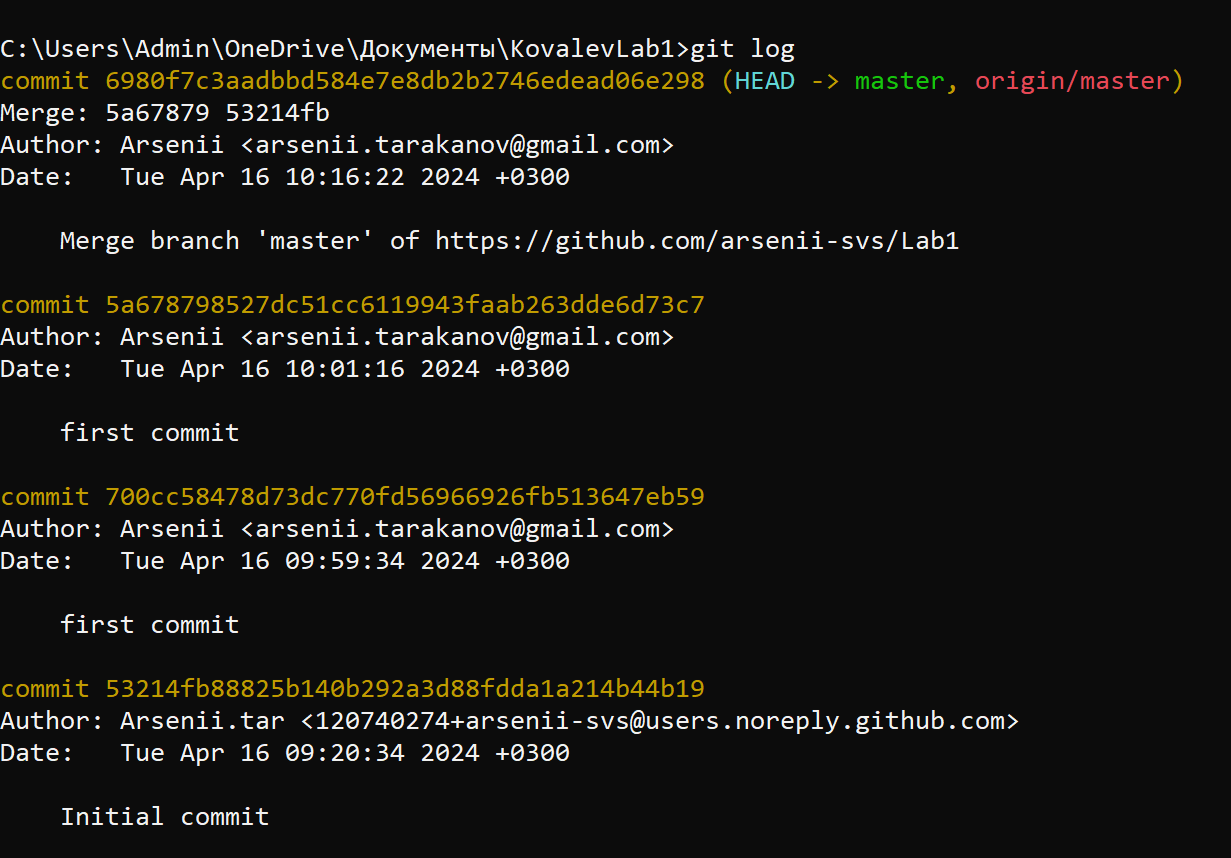
Запросим изменения с сервера. В нашем случае у нас есть файл и на сервере и локальном репозитории, поэтому нам вначале надо получить все данные с нашего репозитория, а потом уже их заливать на сервер нашими локальными данными.



Отравим коммит на сервер, команда, предназначенная для этого — push. Она принимает два параметра: имя удаленного репозитория (origin) и ветку, в которую необходимо внести изменения (master).

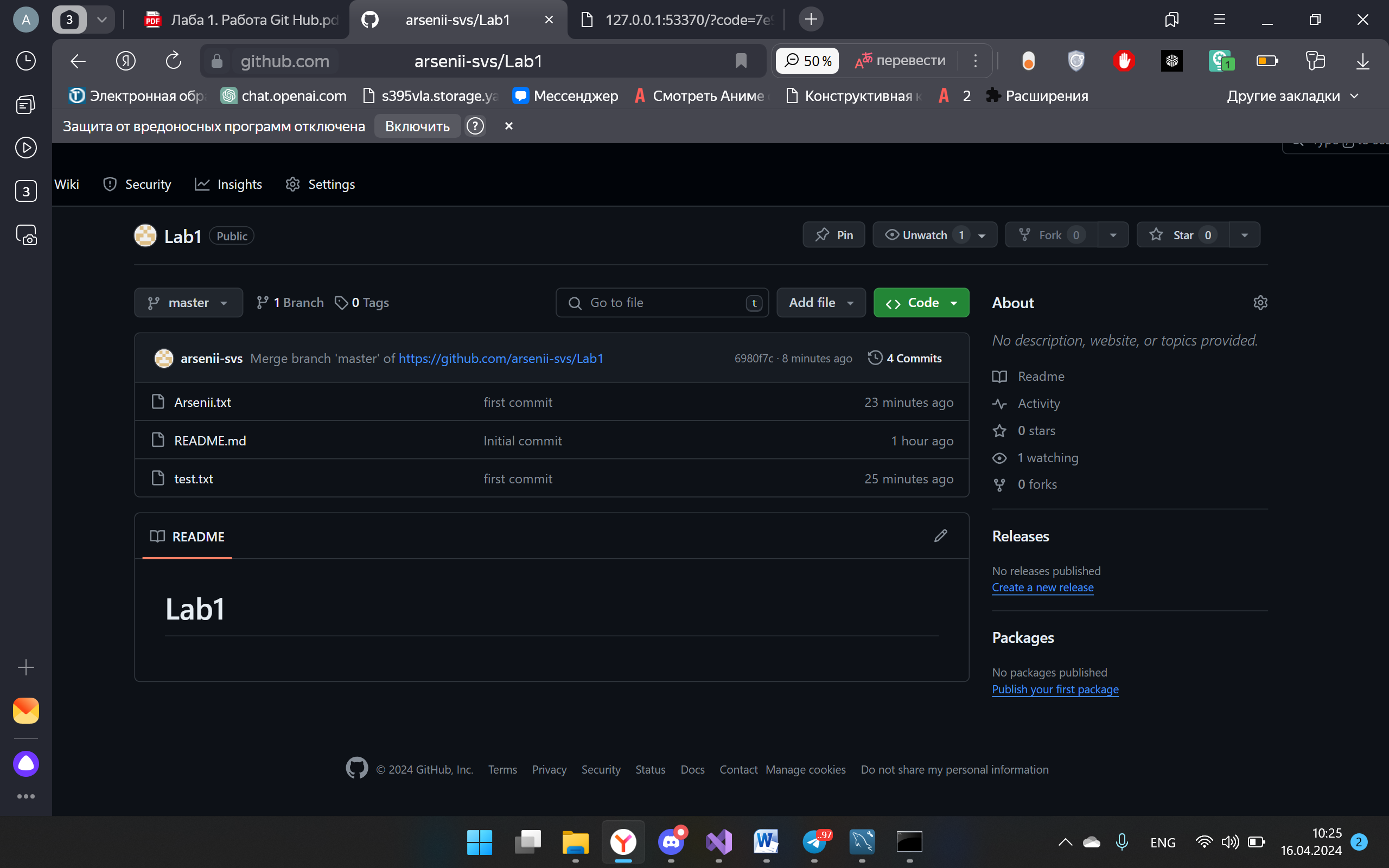


Просмотр изменений с использованием команды git log.

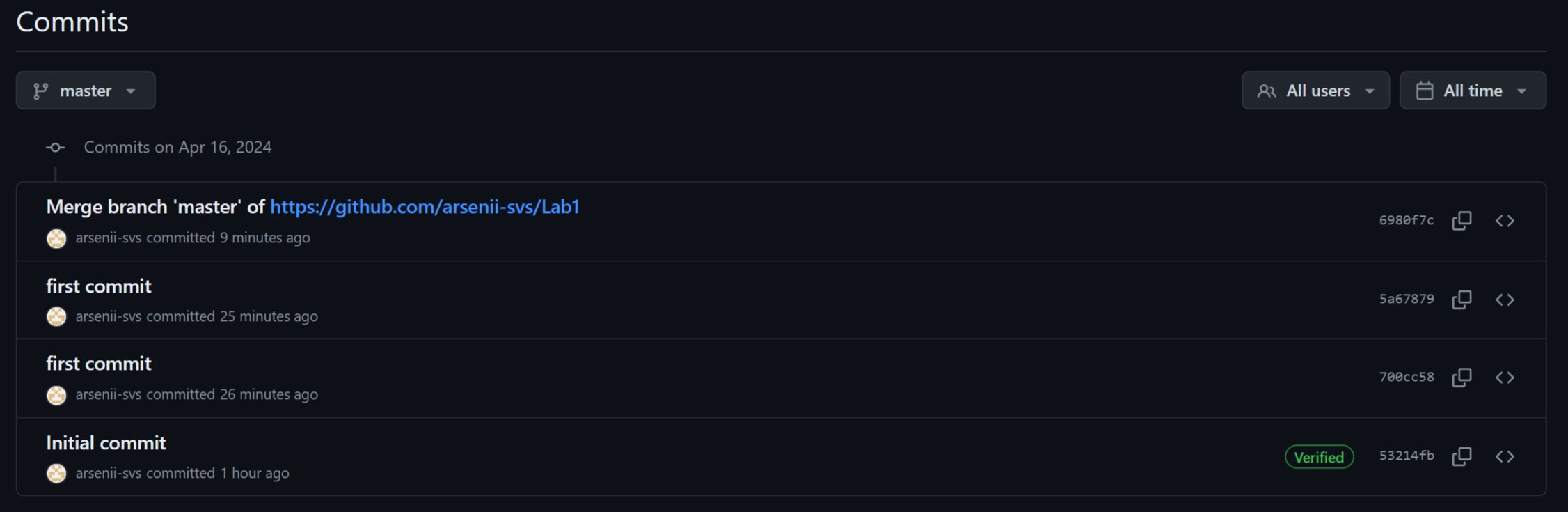


Видны коммиты по изменениям в файле Readme.md и в файле с именем (first commit), чтобы выйти из просмотра лога надо нажать q.

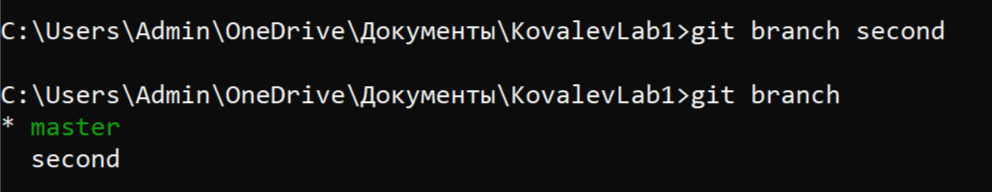
Посмотрим изменения на github.com.



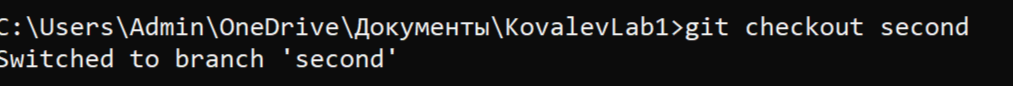
В «commint» проверим все наши изменения.



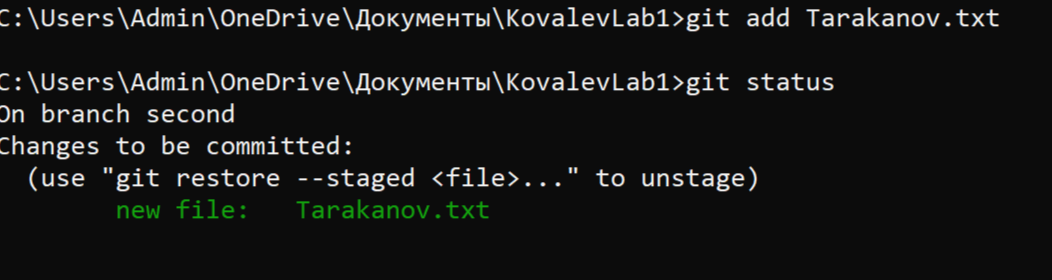
Создадим новую ветку, во время разработки новой функциональности будем работать с копией оригинального проекта, которую называют веткой. Ветви имеют свою собственную историю и изолированные друг от друга изменения до тех пор, пока вы не решаете слить изменения вместе. Обычно в проекте создается минимум 2 ветки (branch) – master и ветка для разработки. Ветка master создается сама, а дополнительные ветки создаются по желанию пользователя (локально), заливаются они в удаленный репозиторий уже с учетом доступных прав. Создадим новую ветку second: git branch second Можно посмотреть в какой ветки находимся, используя команду git branch.

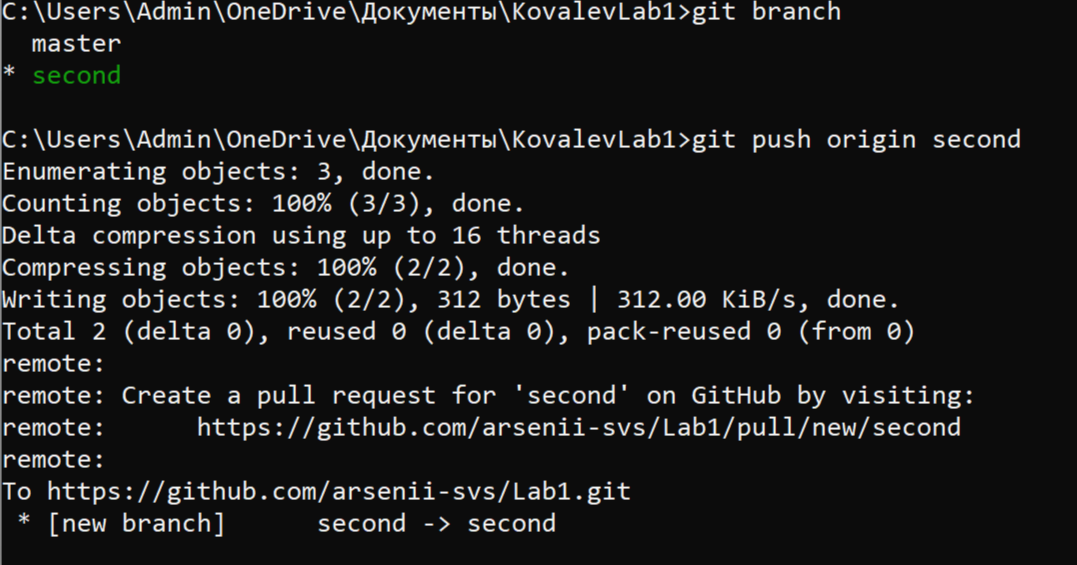


Переключимся на другую ветку, используя команду git checkout second

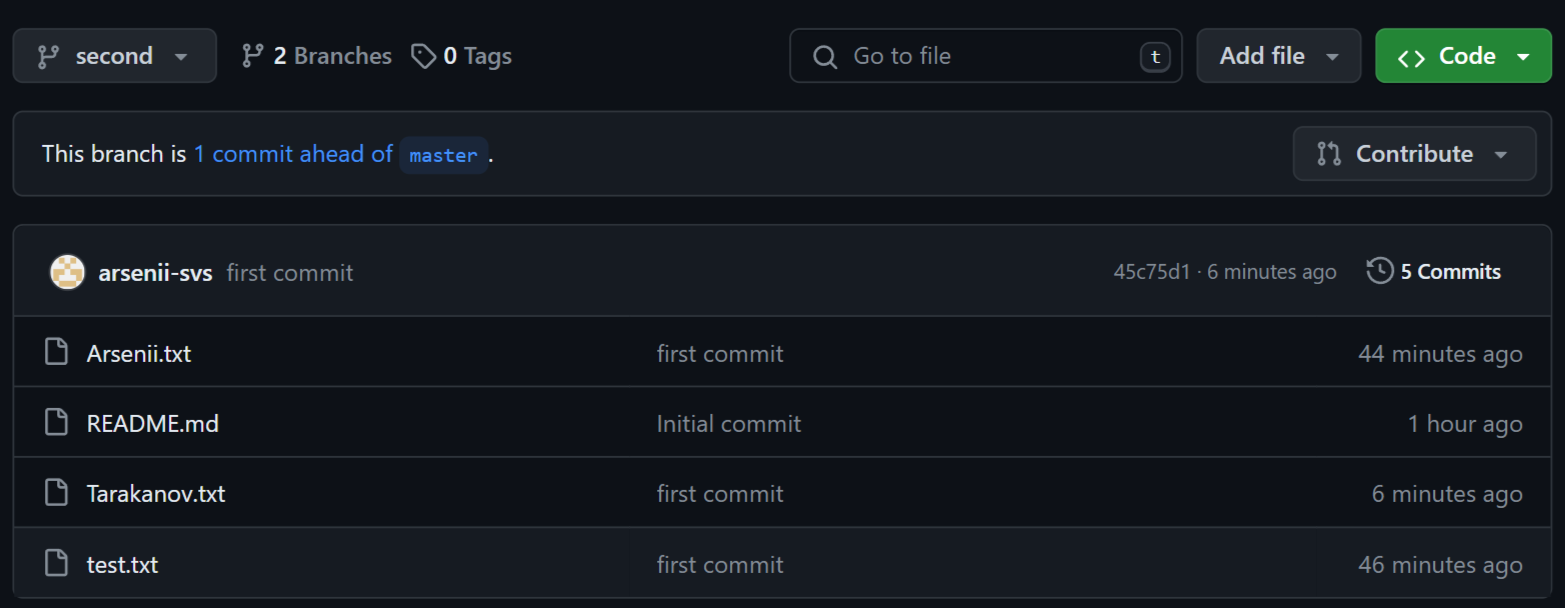


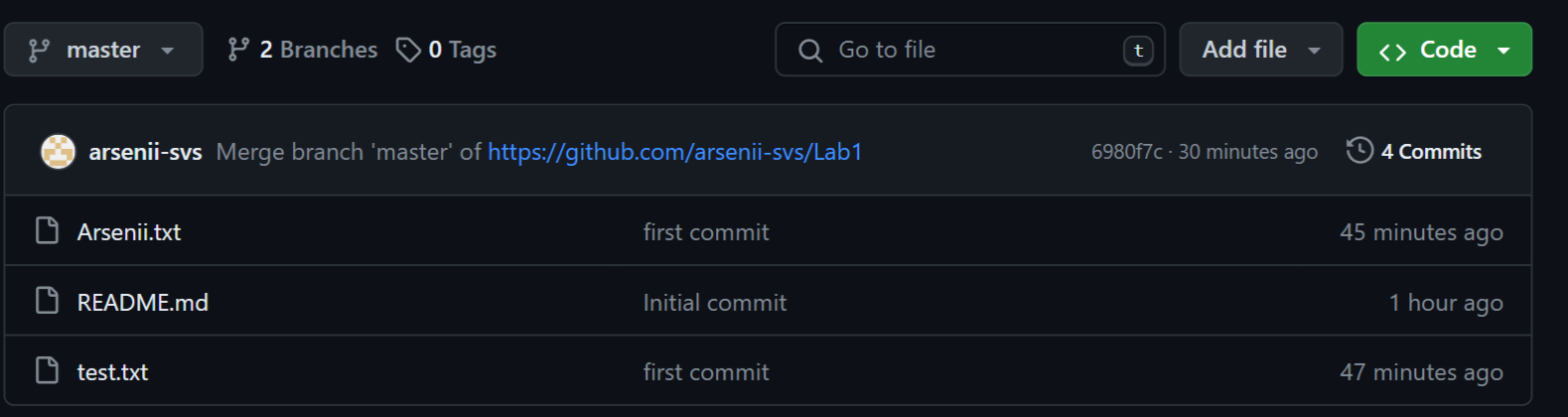
Создадим новый файл в нашем локальном репозитории и напишем в нем свою фамилию, добавим в область подготовленных файлов, закомитим и отправим на сервер.





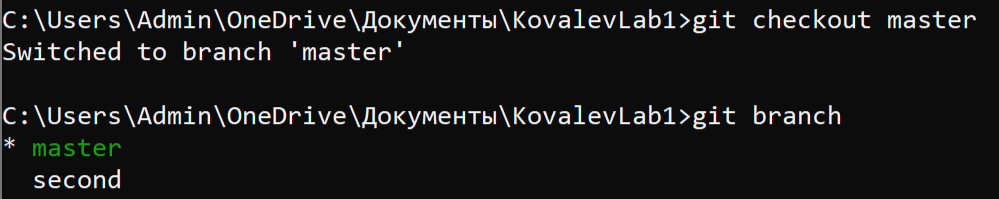
Зайдем на github и посмотрим, что в ветке master нет нашего файла, а в ветке second он присутсует.





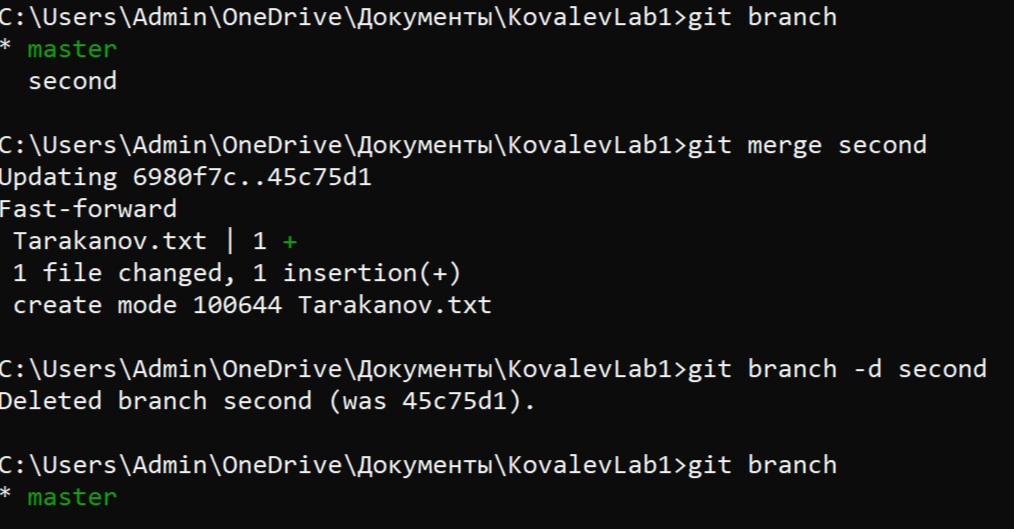
Слияние веток.

Переключимся снова на ветку master

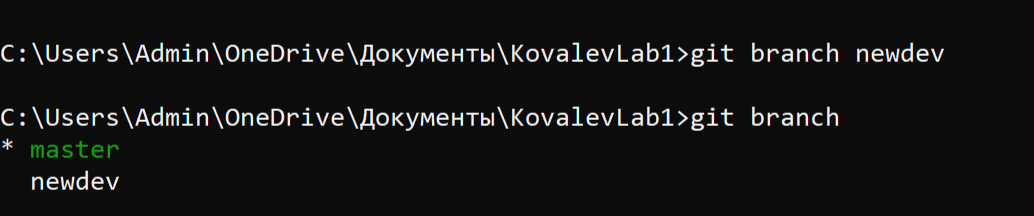


Если через проводник windows в папку с проектом, то файла test\_new.txt не будет виден, потому что мы переключились обратно на ветку master, в которой такого файла не существует. Чтобы он появился, нужно воспользоваться merge для объединения веток (применения изменений из ветки second к основной версии проекта). git merge second.

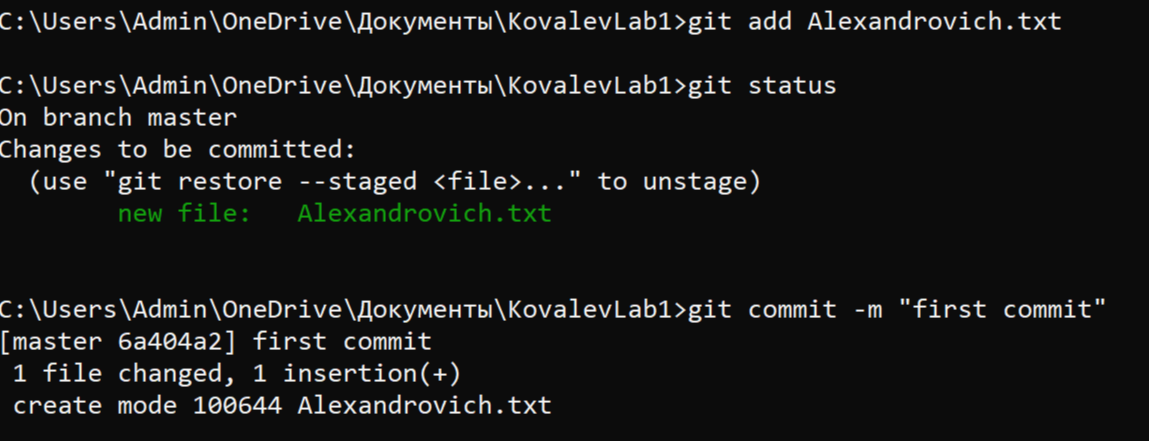
Удалим ветку git branch –d second.

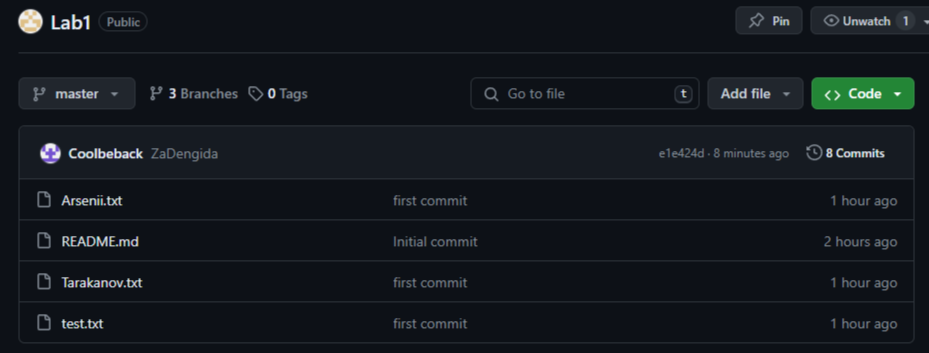


Немного усложним это задание, предположив, что в двух ветках могут быть одинаковые файлы и над ними работают разные разработчики. Создадим ветку с названием newdev Переключитесь на нее

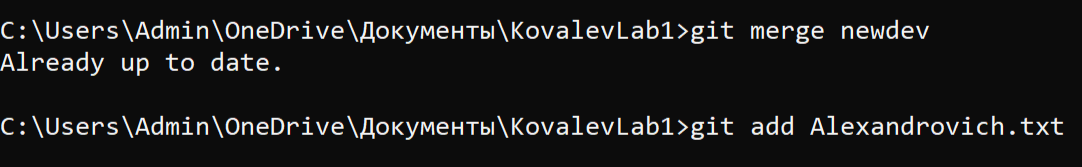


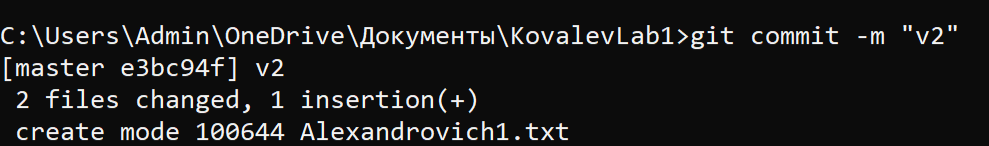
Добавим в файл с именем отчество и закомитим изменения, опробуйте переключиться обратно на ветку master и посмотреть файл: в нем только имя, отчество в другой ветке





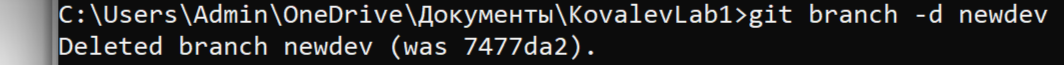
Объединим ветки git merge newdev





Удаление веток на сервере

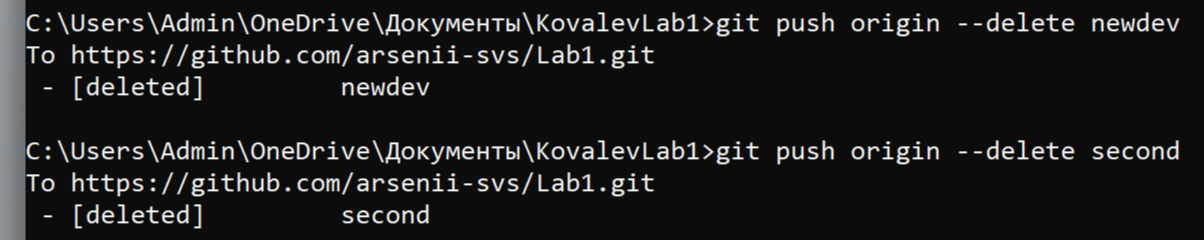
Если у нас есть несмерженная ветка, но вы хотите ее удалить, необходимо использовать команду: git branch –D second

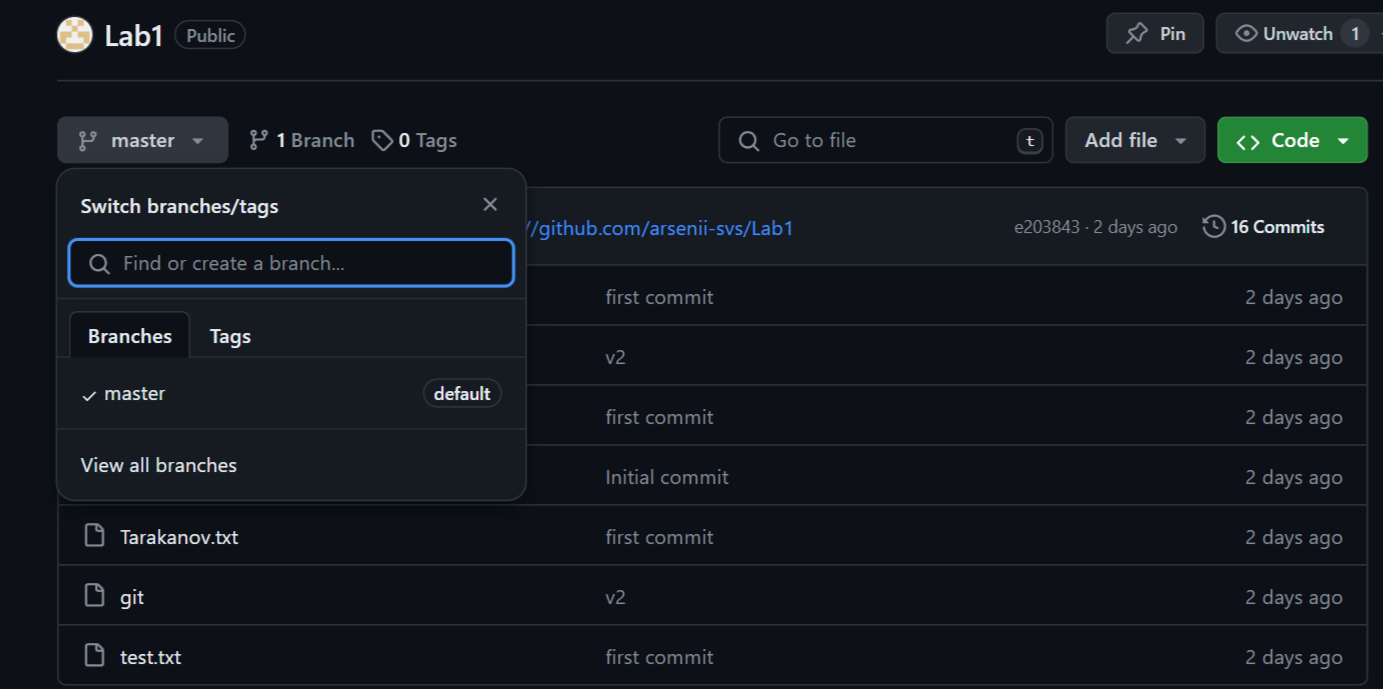


Давайте удалим их из github, введем команды:

git push origin --delete newdev

git push origin --delete second



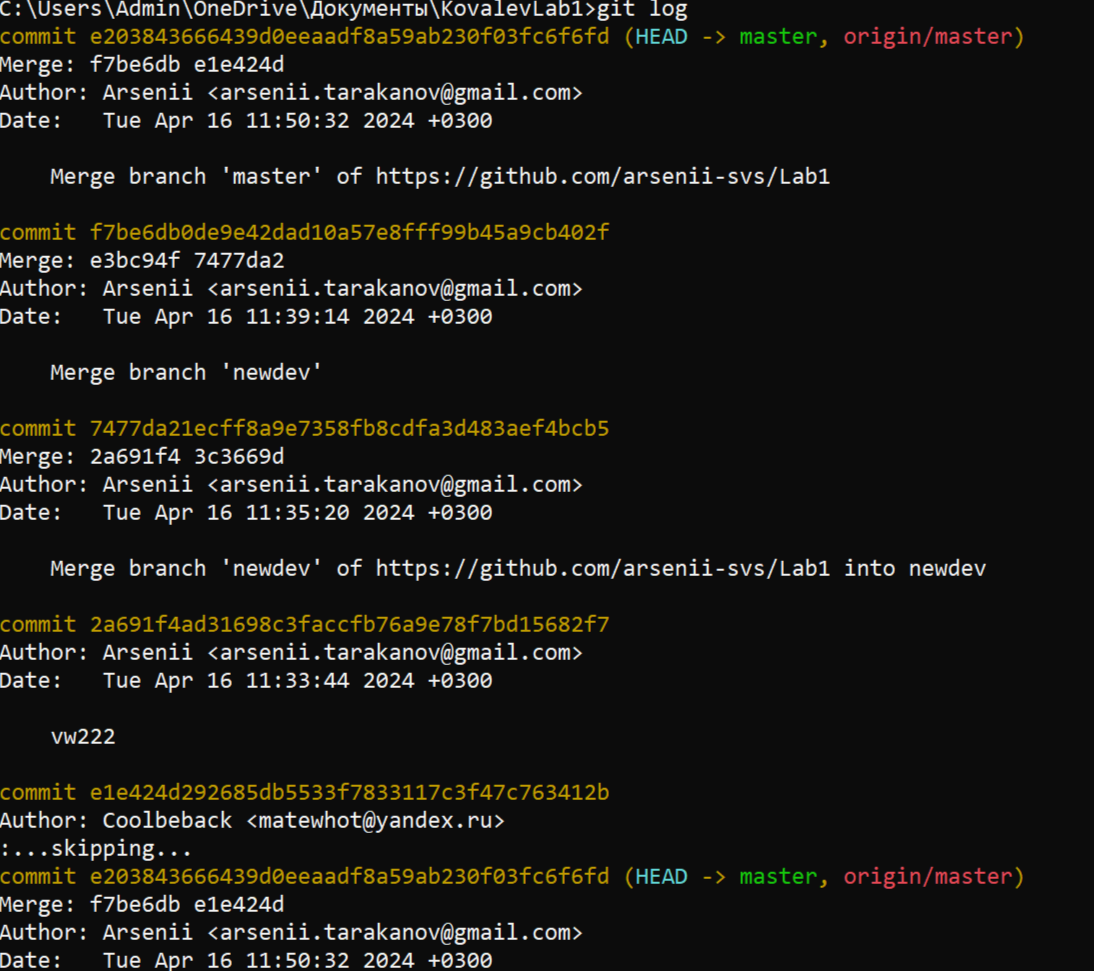


Таким образом, все ветки удалены.

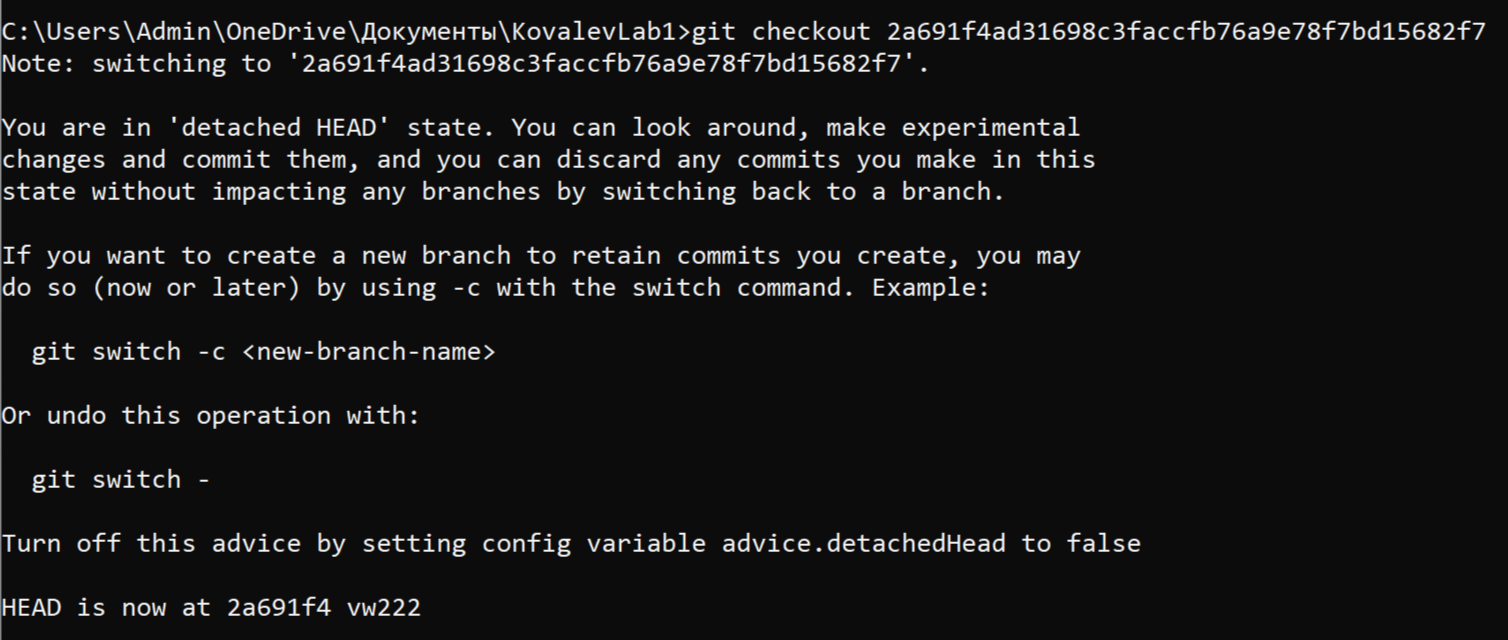
Возврат к предыдущему состоянию.

Гит позволяет вернуть выбранный файл к состоянию на момент определенного коммита. Это делается уже знакомой нам командой checkout, которую мы ранее использовали для переключения между ветками. Но она также может быть использована для переключения между коммитами.

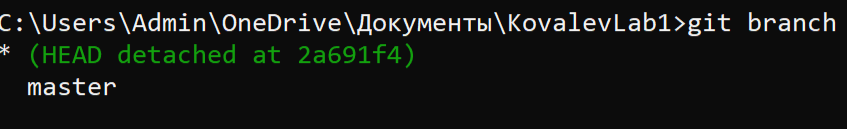
Чтобы посмотреть все комиты, можно использовать команду git log.



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



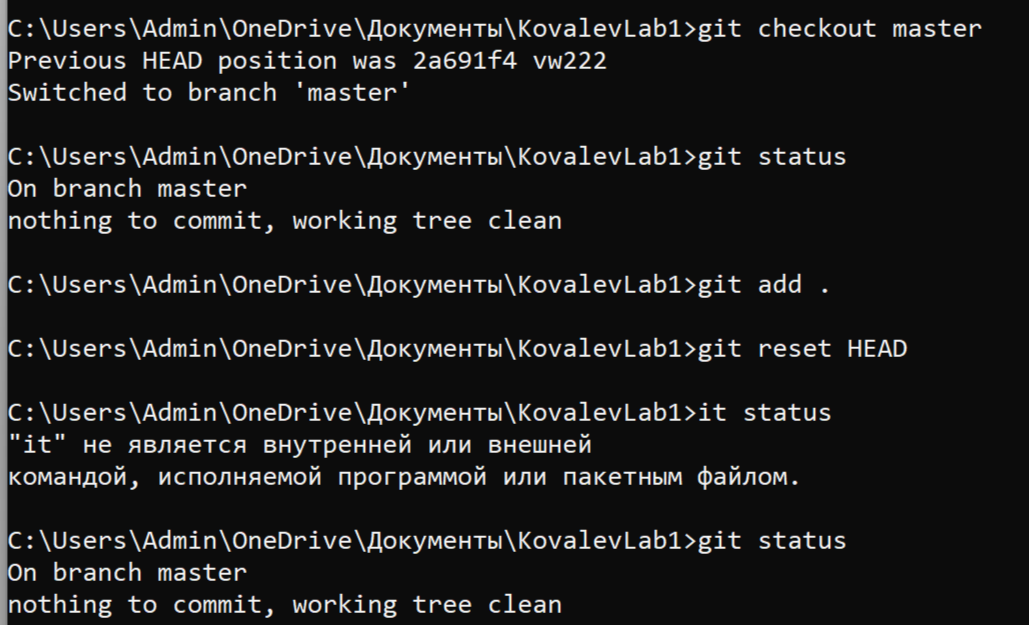
Если посмотреть файлы теперь, то увидим, что они поменялись – мы вернулись назад. Чтобы это произошло, создалась псевдо-ветка начинающаяся на этом коммите, посмотрим ветки и увидим:

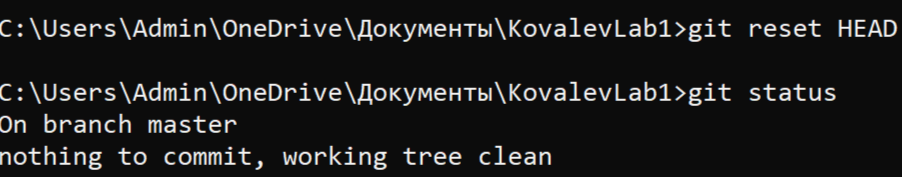


При отмене старых коммитов нужно быть готовым к тому, что возникнут конфликты. Такое случается, если файл был изменен еще одним, более новым коммитом. И теперь git не может найти строчки, состояние которых нужно откатить, так как они больше не существуют.

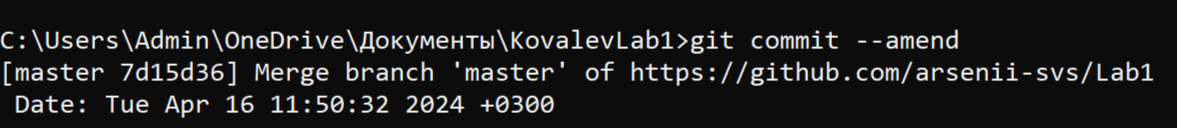
Исправление коммита

Возможна ситуация, что мы закрепили файлы (поменяйте что-то в файле и закрепите их, вызовите git status), но еще не коммитили, тогда, если хотим убрать файлы из области закрепления, вызовем команду: git reset HEAD. Наши файлы останутся такими же, но уйдут из области закрепления (вызовем git status) и снова будет показано, что есть измененные файлы





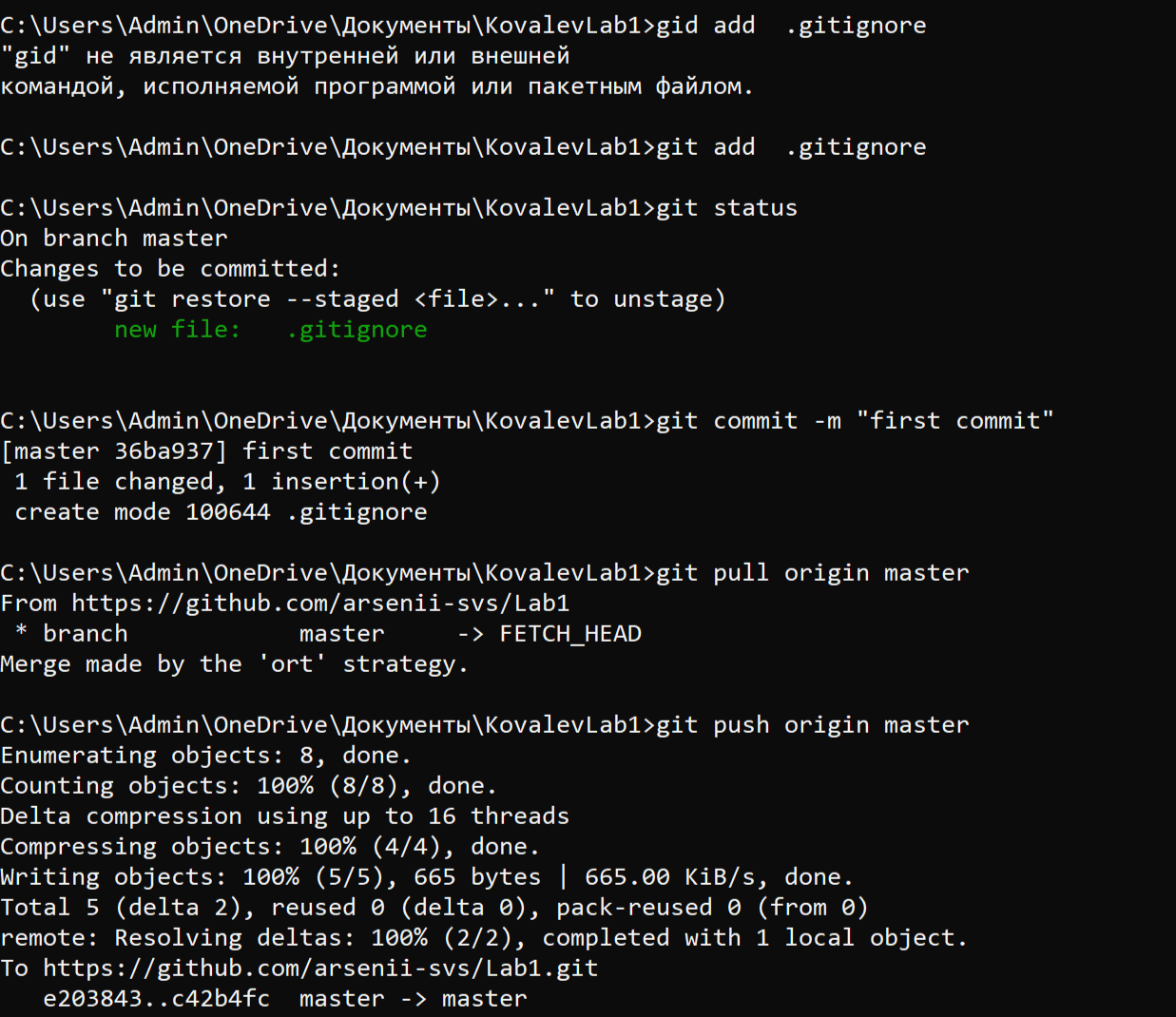
Если хотим изменить комментарий можно вызвать команду: git commit –amend. Откроется консоль в редакторе Vi, чтобы начать что-то менять, надо нажать клавишу Insert, когда изменения в комментарий внесены, необходимо снова нажать Insert. Чтобы сохранить изменения, нажмите Esc, потом нажмите двоеточие (появится внизу экрана), напишите w! и нажмите Enter. Для выхода снова нажмите двоеточие, введите q! и нажмите Enter.



Отправка только нужных файлов на сервер

В большинстве проектов есть файлы или целые директории, в которые мы не хотим (и, скорее всего, не захотим) коммитить. Мы можем удостовериться, что они случайно не попадут в git add -A при помощи файла «.gitignore» 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. Файл «.gitignore» должен быть добавлен, закоммичен и отправлен на сервер, как любой другой файл в проекте.

Впишем в «.gitignore» файл, который не хотим отправлять на сервер. Зафиксируем изменения, закоммитим, отправим на сервер. Все изменения можно проводить непосредственно на github, там также можно делать коммиты, только фиксация изменений происходит при сохранении файла.

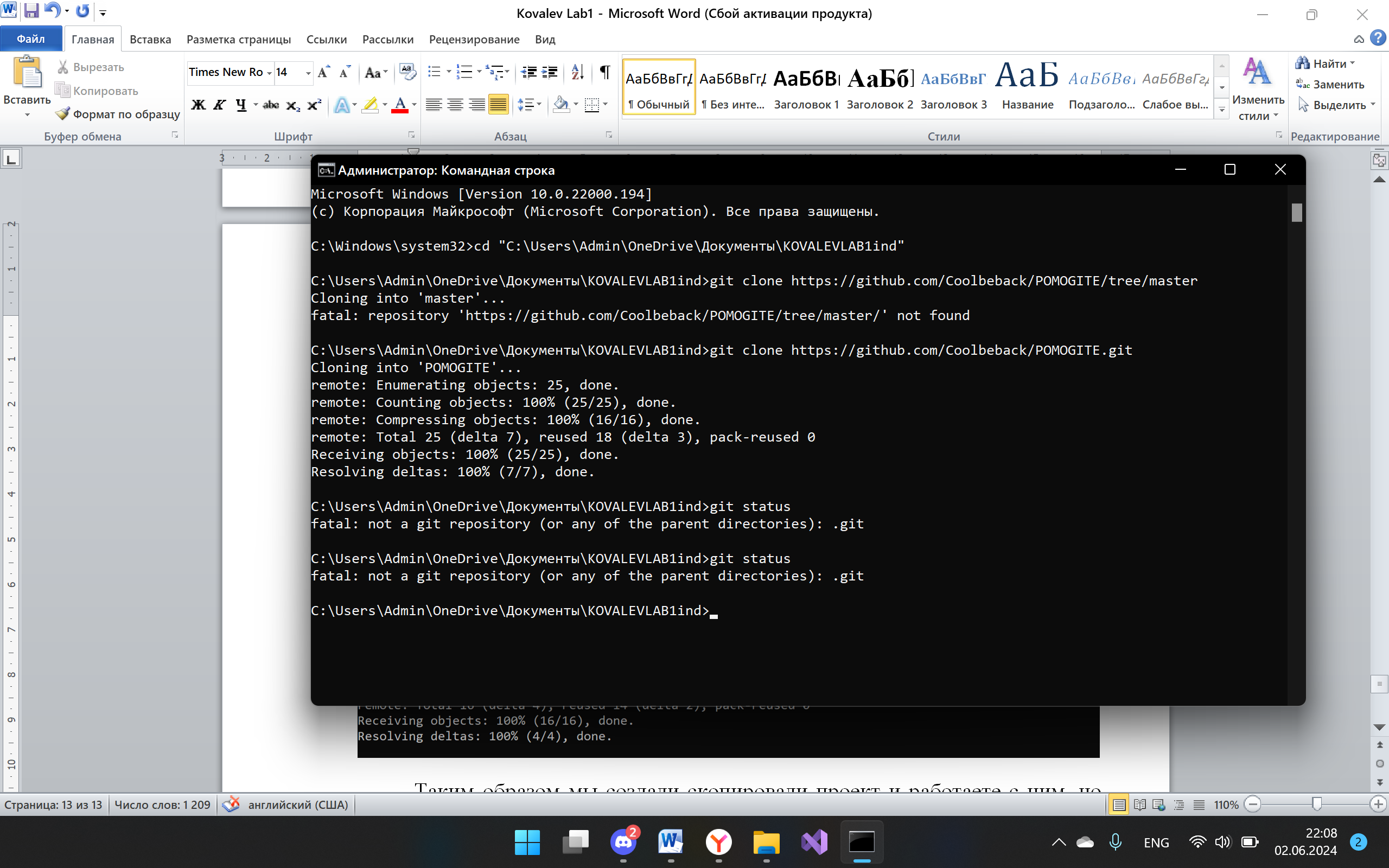




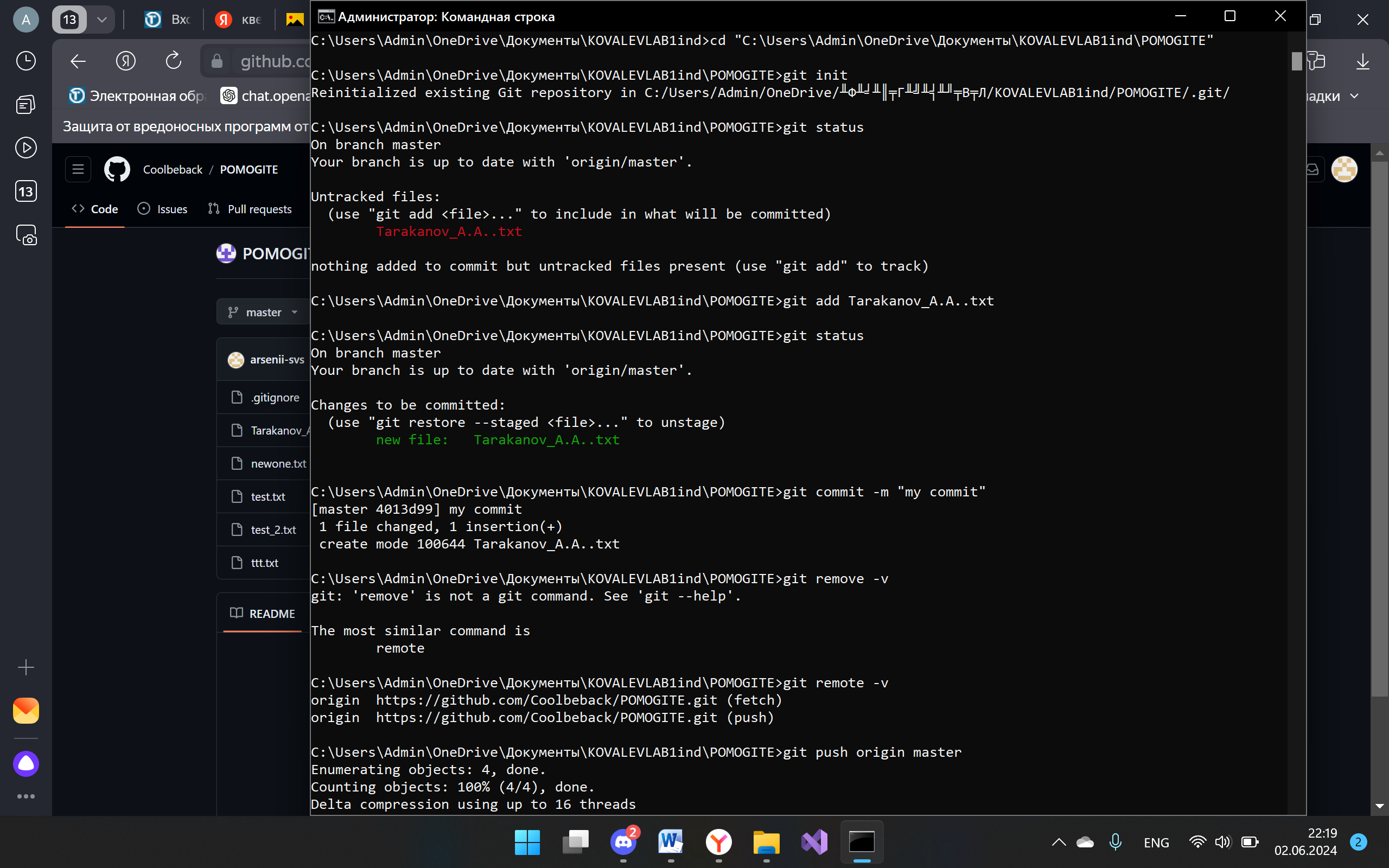
# Совместная работа с git

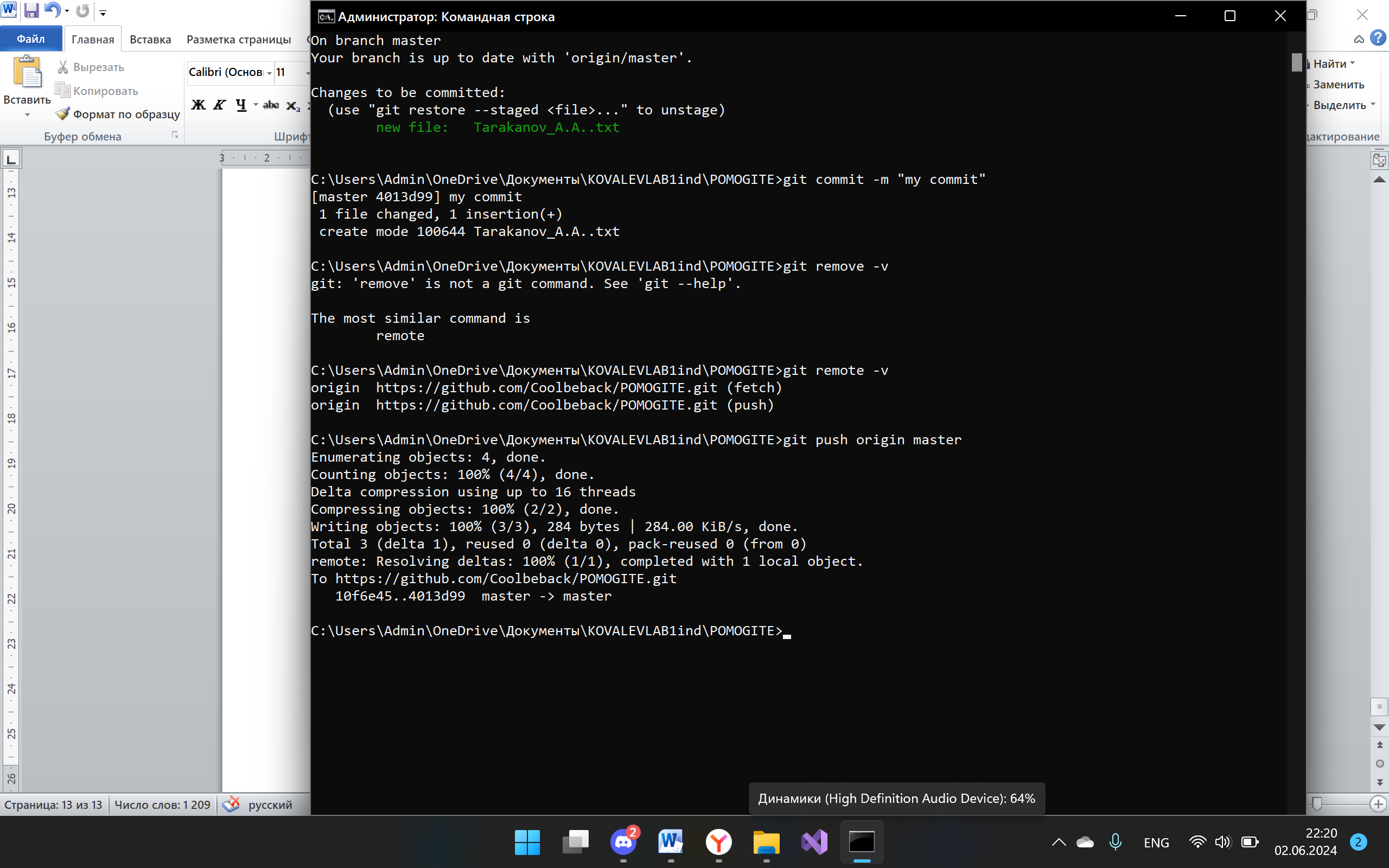
Выберем себе напарника, скинем друг другу адреса своих репозиториев Переходим по полученным ссылкам на репозиторий и нажимаем fork.

Теперь это репозитоий добавляется к нам в проекты github. Далее через консоль клонируем этот репозиторий на локальный компьютер, предварительно создадим папку, в которую будем клонировать и в консоли перейдем в нее git clone «адрес репозитория»



Перейдем в файл с клонированным репозиторием, закоммитим и запушим новый файл в github.





# Вывод

Таким образом мы в ходе проделанной работы освоили навыки работы с GitHub и командной строкой.