

**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-20-07:** | \_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_  *(дата) (подпись)* | **Петровичева А.С.** |
| **Проверил**  **к.т.н., доцент** | \_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_  *(дата) (подпись)* | **Ковалев И.А.** |

**Москва 2023 г.**

**Содержание**

1. **Цель................................................................................................................2**
2. **Краткая теория по СКВ.............................................................................2**
3. **Выполнение лабораторной работы.........................................................3**
4. **Выводы.......................................................................................................17**

**Цель лабораторной работы**:

Ознакомление с системами контроля версий, получение навыков работы с ними на примере Git Hub.

**Краткая теория по СКВ:**

Система контроля версий (СКВ) — это система, регистрирующая изменения в одном или нескольких файлах с тем, чтобы в дальнейшем была возможность вернуться к определённым старым версиям этих файлов.

**Существуют:**

**Локальные системы контроля версий (rcs)**

Это замена обычному копированию. С их помощью можно с лёгкостью следить за изменением нужных файлов , не теряя их на устройстве. Кроме того ЛСКВ даёт возможность возвращать отдельные файлы к прежнему виду, возвращать к прежнему состоянию весь проект, просматривать происходящие со временем изменения и многое другое.

**Централизованные системы контроля версий(CVS, Subversion и Perforce)**

Замена ЛСКВ . Позволяет нескольким разработчикам за разными компьютерами сотрудничать между собой. Все знают, кто и чем занимается в проекте, администрировать ЦСКВ намного легче, чем локальные базы на каждом клиенте.

Однако есть и несколько серьёзных недостатков. Наиболее очевидный — централизованный сервер является уязвимым местом всей системы. Если сервер выключается на час, то в течение часа разработчики не могут взаимодействовать, и никто не может сохранить новой версии своей работы. Если же повреждается диск с центральной базой данных и нет резервной копии, вы теряете абсолютно всё — всю историю проекта, разве что за исключением нескольких рабочих версий, сохранившихся на рабочих машинах пользователей. Локальные системы контроля версий подвержены той же проблеме.

**Распределённые системы контроля версий (Git, Mercurial, Bazaar или Darcs)**

Замена ЦСКВ. В таких системах клиенты не просто выгружают последние версии файлов, а полностью копируют весь репозиторий. Поэтому в случае, когда "умирает" сервер, через который шла работа, любой клиентский репозиторий может быть скопирован обратно на сервер, чтобы восстановить базу данных. Каждый раз, когда клиент забирает свежую версию файлов, он создаёт себе полную копию всех данных.

Кроме того, в большей части этих систем можно работать с несколькими удалёнными репозиториями, таким образом, можно одновременно работать по-разному с разными группами людей в рамках одного проекта. Так, в одном проекте можно одновременно вести несколько типов рабочих процессов, что невозможно в централизованных системах.

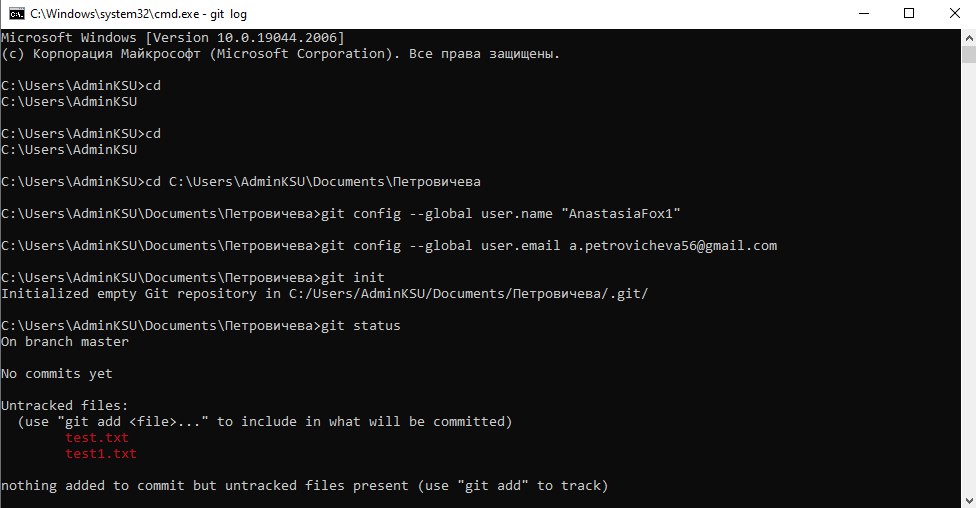
**Выполнение лабораторной работы:**

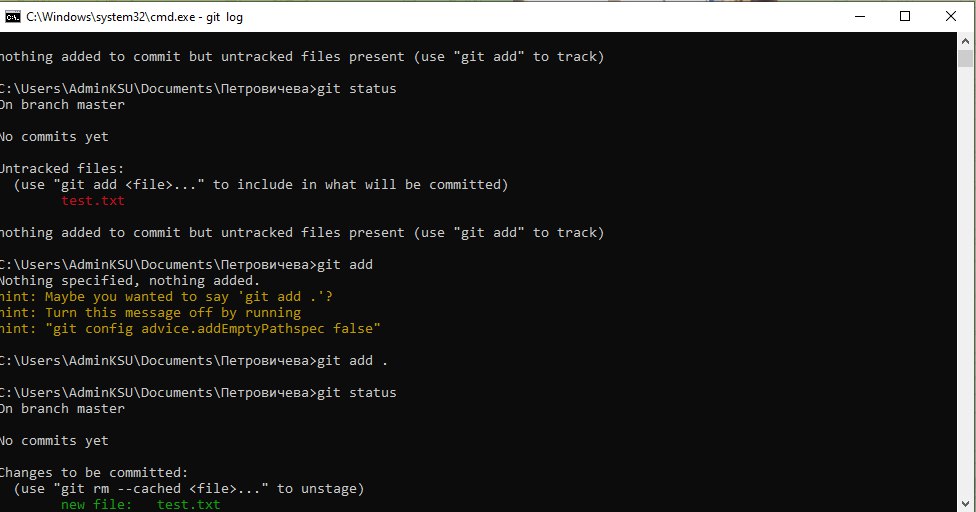
Создав учётную запись, репозиторий, настраиваем имя пользователя и адрес электронной почты. Затем инициализируем заранее выбранную папку как git репозиторий командой *git init* .

Создаём файл и пишем *git stаtus* , чтобы проверить состояние нашего локального репозитория. Видим , что есть новый файл, но он не отслеживается. Так как у нас один файл пишем команду *git add Text.txt* , чтобы его добавть в отслеживаемые, тем самым подготовив к коммиту.

Проверив состояние репозитория, убеждаемся в том что файл готов к коммиту и пишем команду *git commit –m “first commit”* . Далее локальный связываем репозиторий с репозиторием на Git Hub командой *git remote add origin https://github.com/AnastasiaFox1/Laba1 .*

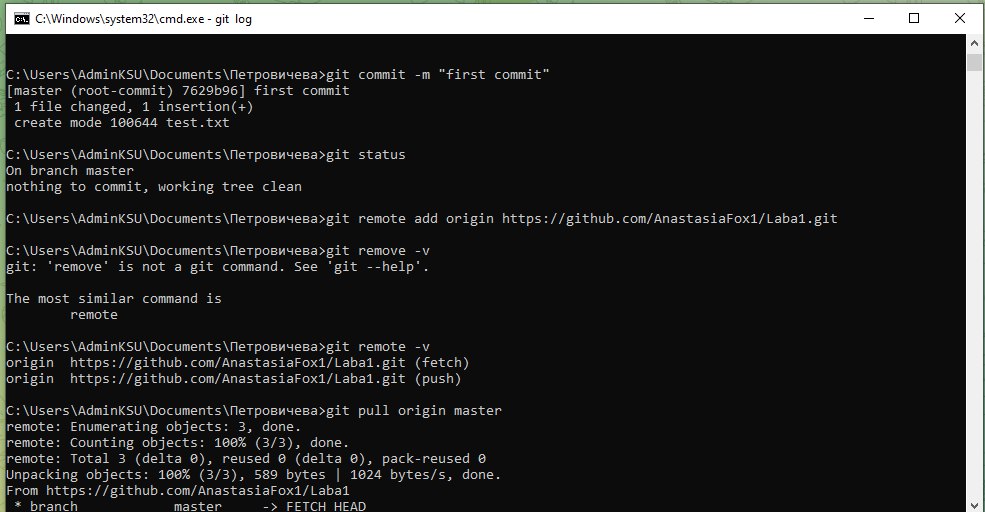
Далее, проверив к какому проекту мы подключены командой *git remote -v ,* попытаемся получить данные с репозитория ( пока что пустого , так что просто проверяем команду) командой *git pull origin master* . Так как не была произведена авторизация, то сделать это не получилось.

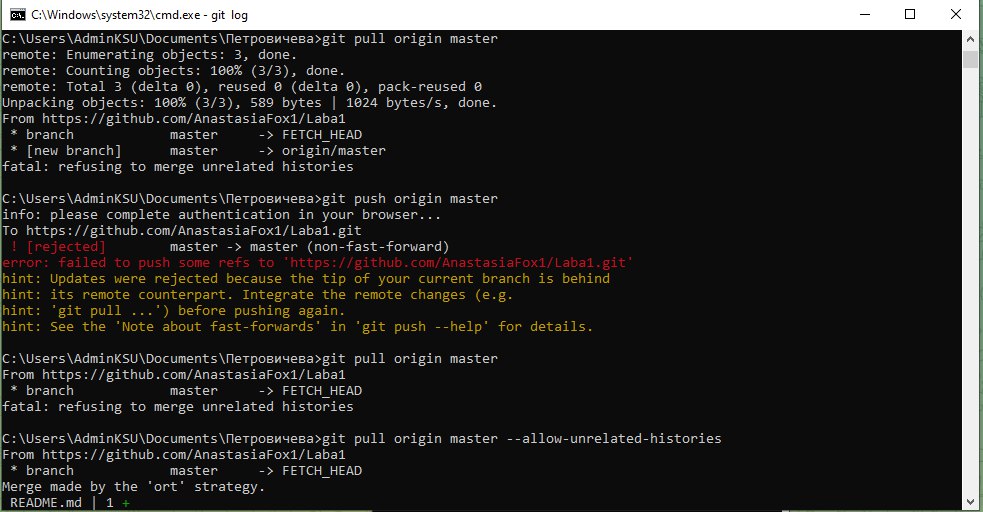


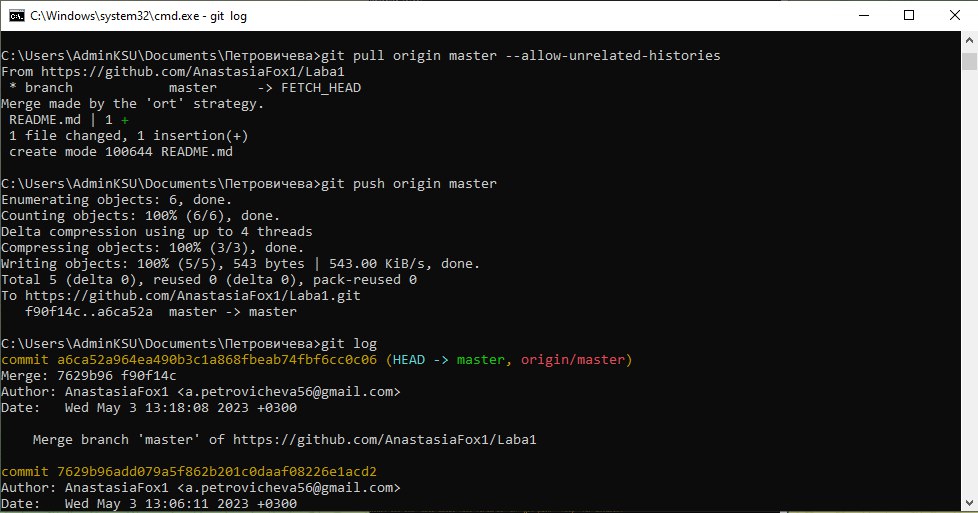


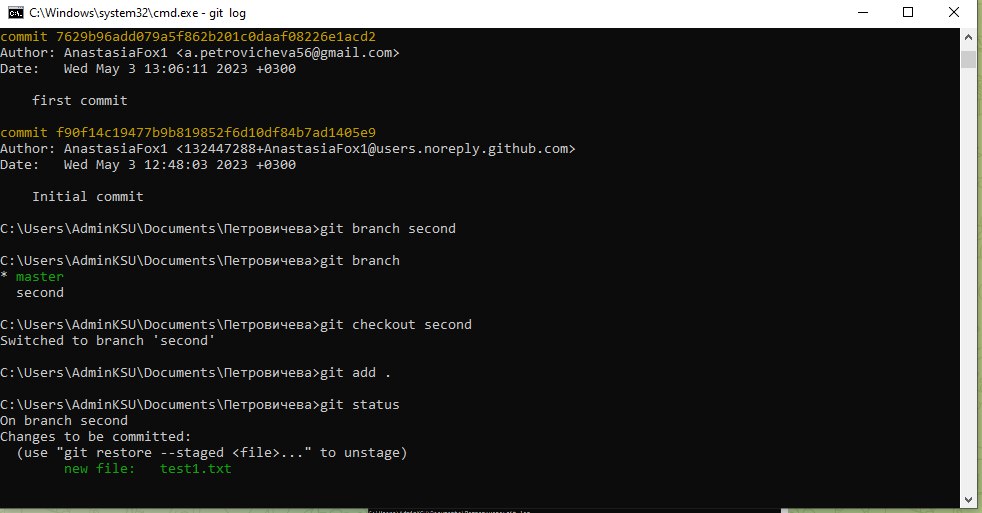
Произведём попытку отправить коммит на сервер и авторизироваться. Пишем *git push origin master* , авторизируемся в браузере и по итогу наш коммит оказывается на сервере. Посмотрим наши изменения с использованием команды *git log* . Затем создаём новую ветку second командой *git branch second*, затем проверяем какие у нас есть ветки.

Далее выбираем вторую ветку и создав файл добавляем его ( во вторую ветку).

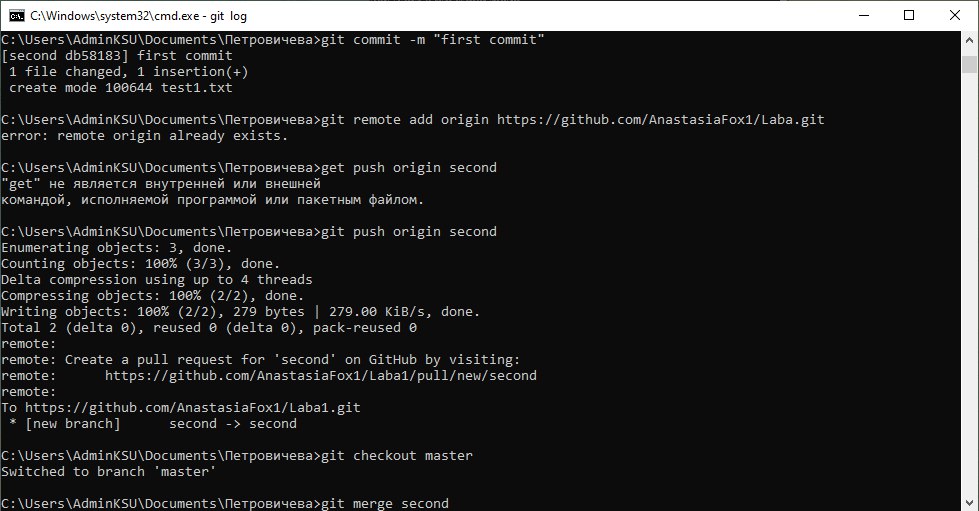


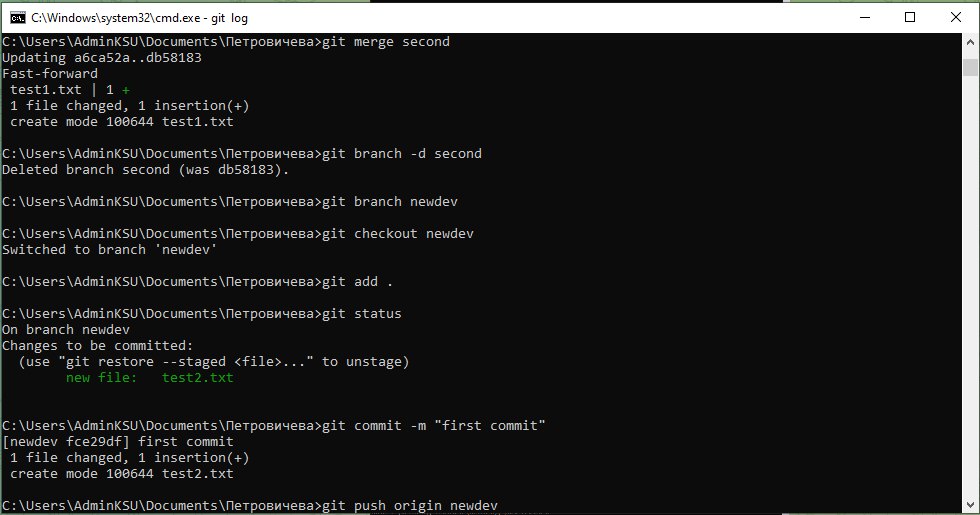




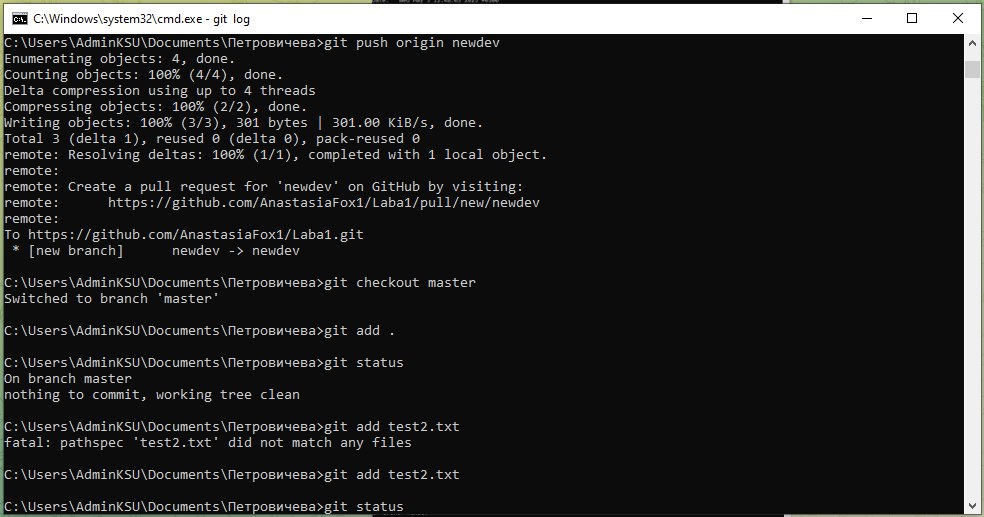


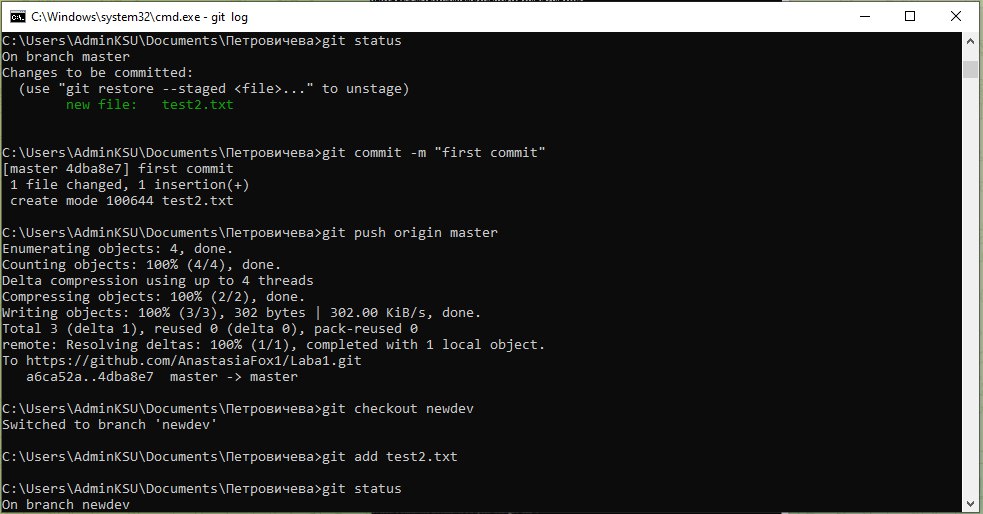
Коммитим, проверяем и отправляем на сервер ( новую ветку с файлом). Переключившись на основную ветку , видим , что файл пропал. Попробуем слить ветки. Для этого пишем команду *git merge second* . После удаляем вторую ветку.



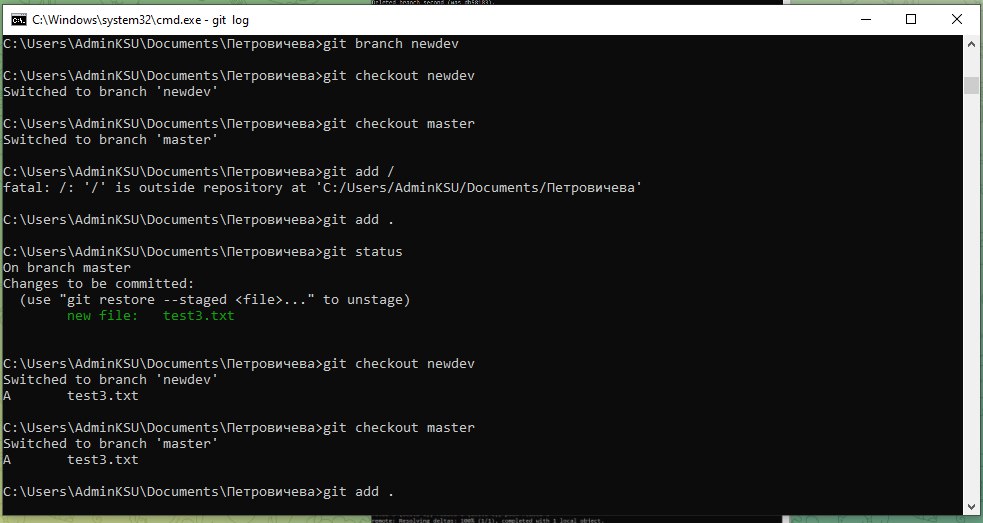


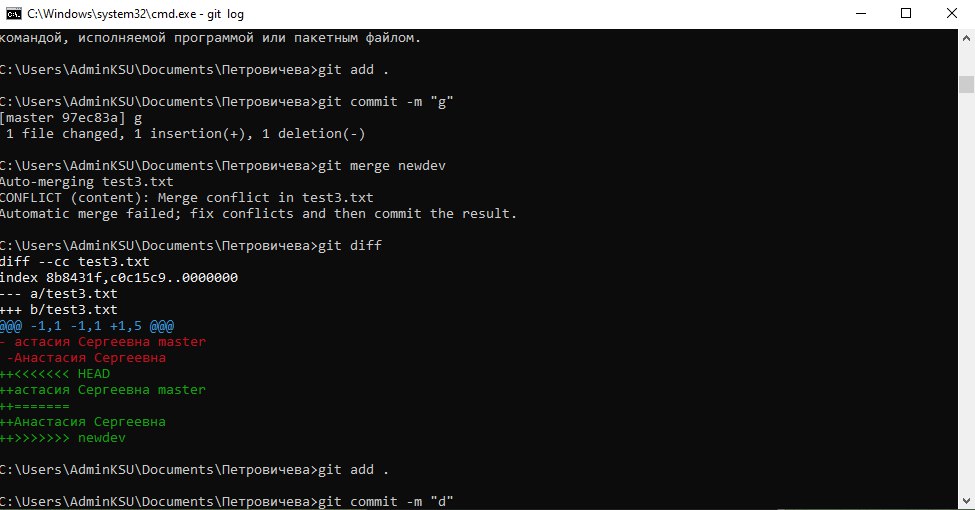
Создаём новую ветку , переключаемся на неё, добавляем новый файл, коммитим. Переключившись на основную ветку сливаем две ветки, затем переключаемся на основную ветку.

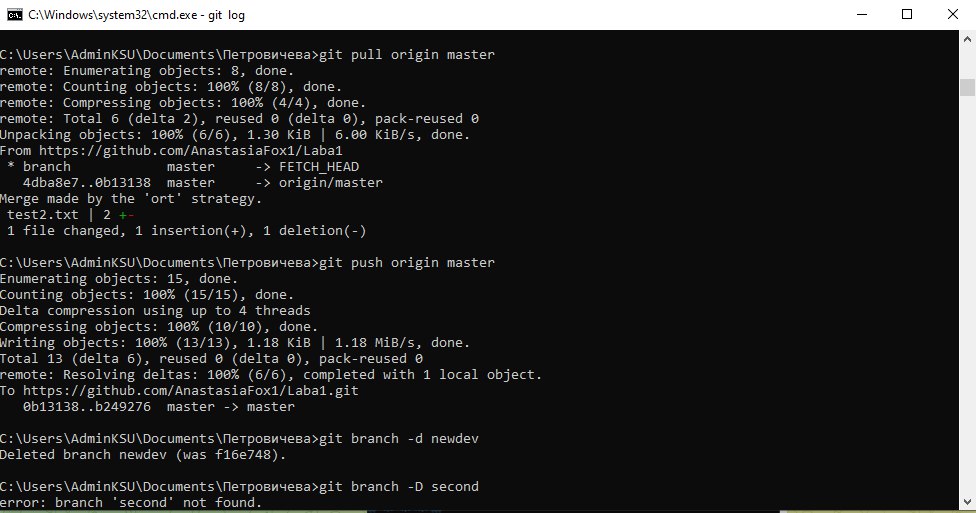




Так же добавляем новый файл в основную ветку, по итогу файлы в должны отличаться. Попытавшись соеденить ветки получаем ошибку, так как файлы имеют одинаковое название , но разное содержание.

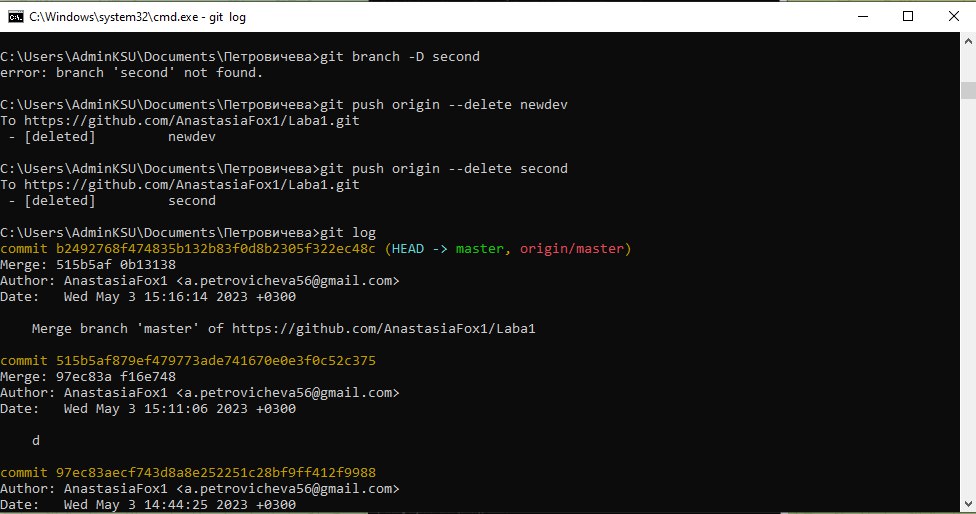




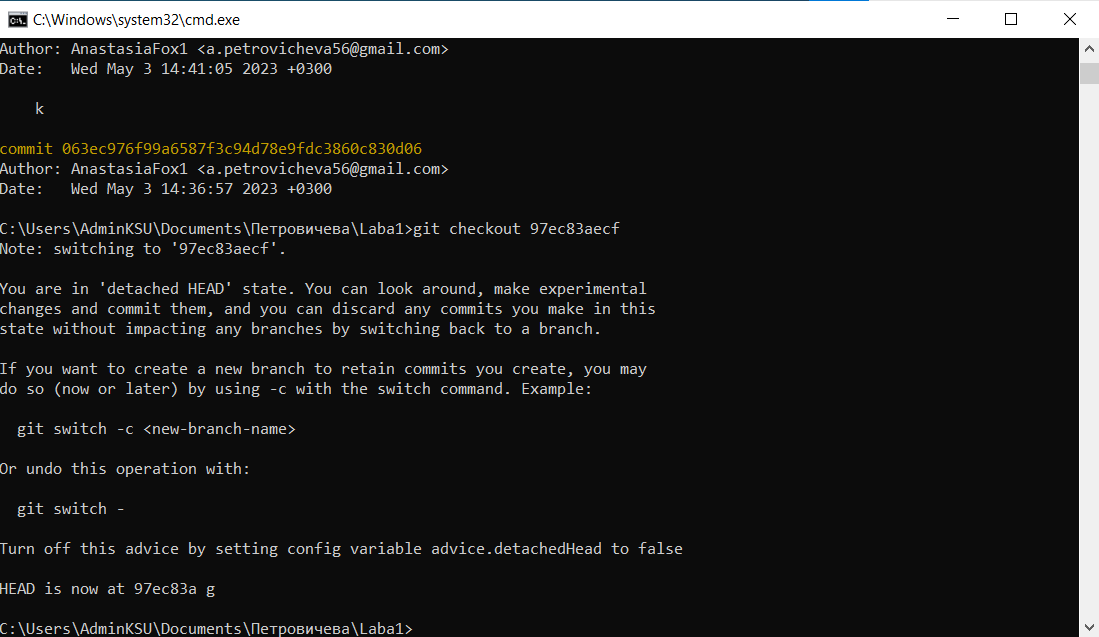


Разрешаем конфликт , выбрав что оставить , а что убрать внутри файла. После чего добавив файл, коммитим завершая слияние. Удаляем доп ветку.

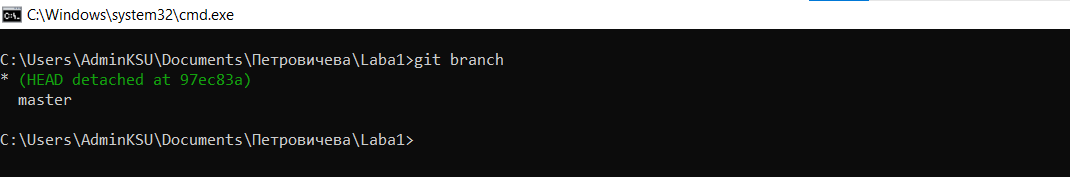
Также удаляем ветку second , которую удалили локально, но не с сервера.



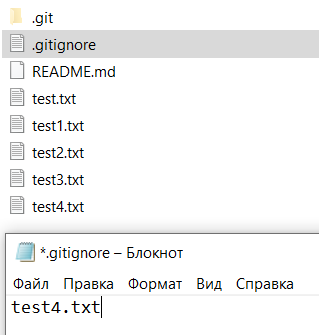
Смотрим историю изменений нашего репозитория.

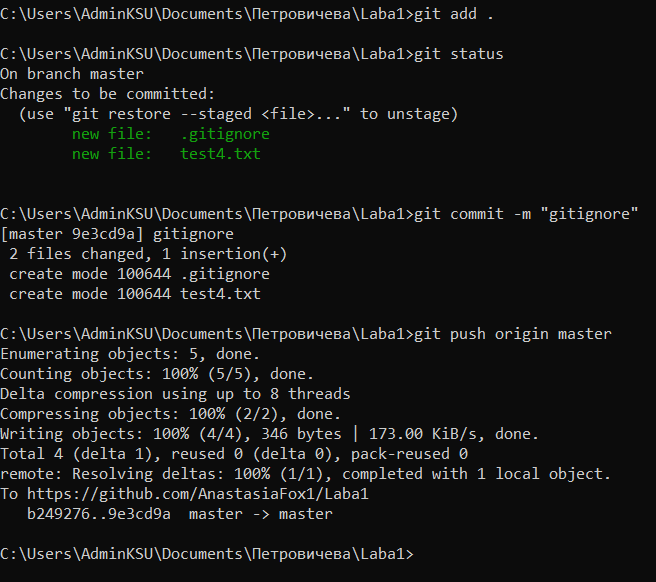


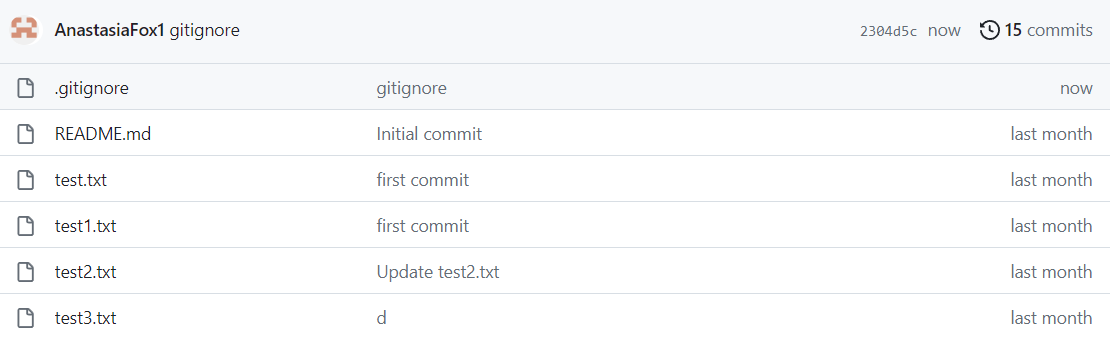
Откатываемся на более ранний коммит. После чего возвращаемся обратно.



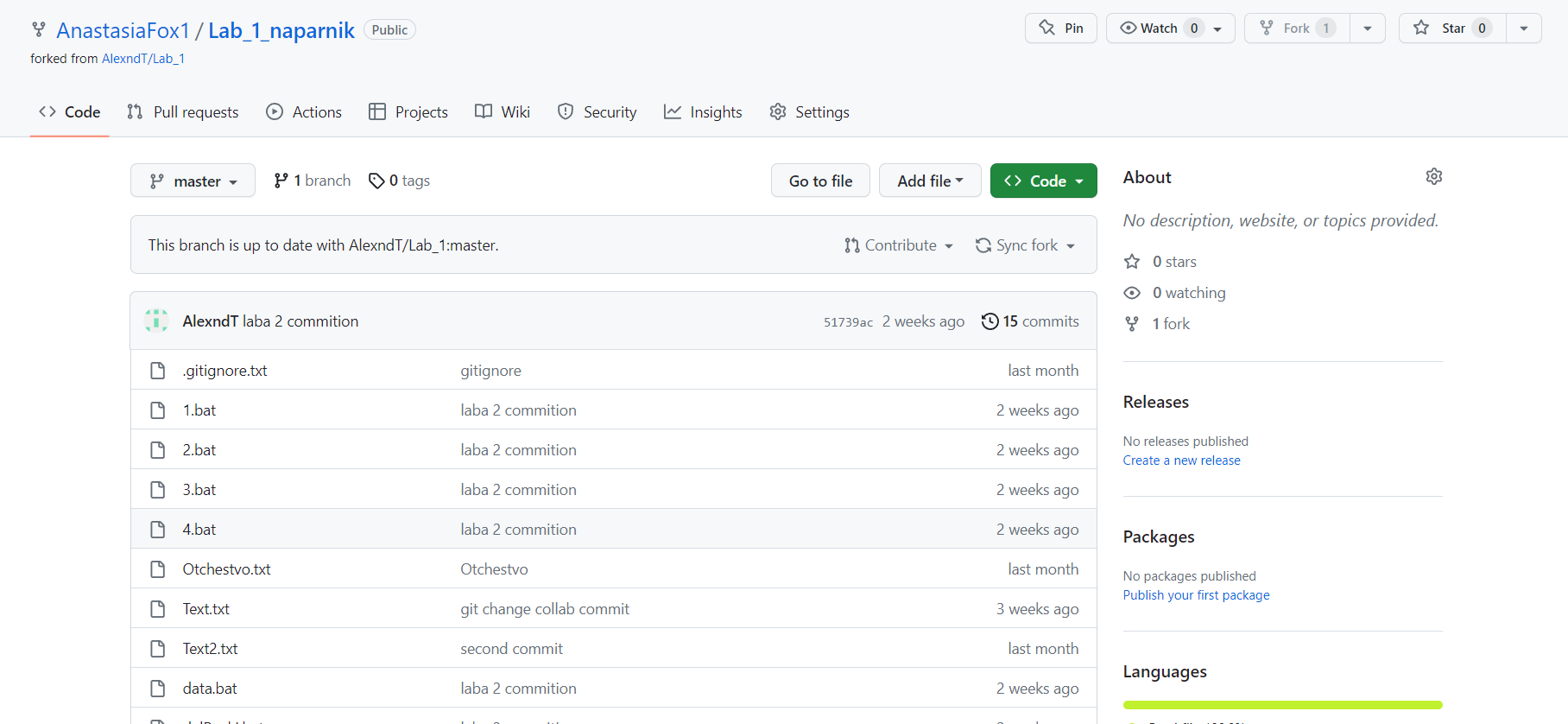
Попытаемся использовать файл .gitignore. Данный файл позволяет не отправлять определённые файлы, папки или даже целые директории на сервер, просто добавив их названия в файл.

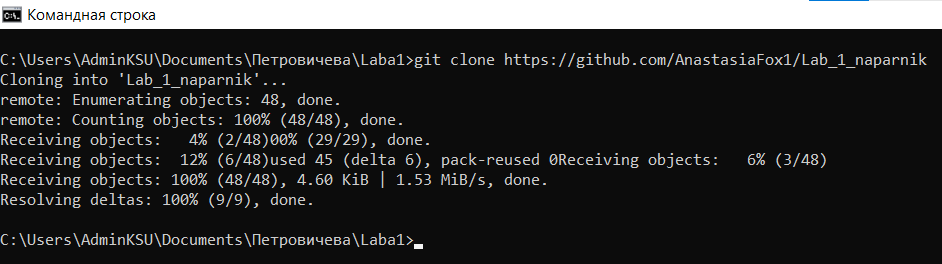


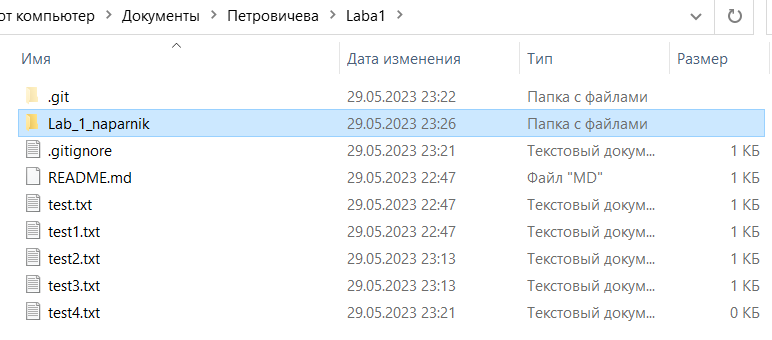




Далее попытаемся форкнуть репозиторий. Перейдя по полученной ссылке, форкаем и клонируем репозиторий к нам на компьютер. Однако, изменения в репозитории не будут отображаться в репозитории напарника. Поэтому, получив доступ, клонируем репозиторий , после чего добавляем файл, коммитим , проверяем изменения на сервере.







**Выводы:**

В данной лабораторной работе мы научились работать с Git Hub, а именно настраивать репозиторий, отправлять файлы на сервер (предварительно добавив и закоммитив), добавлять/удалять ветки , сливать их , устранять конфликты и так далее.