|  |
| --- |
| МИНОБРНАУКИ РОССИИ |
| Федеральное государственное бюджетное образовательное учреждение высшего образования **«МИРЭА − Российский технологический университет»**  **РТУ МИРЭА** |

**Институт информационных технологий (ИИТ)**

**Кафедра инструментального и прикладного программного обеспечения (ИППО)**

**ОТЧЕТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ**

по дисциплине «Технология разработки программных приложений»

**Практическое задание № 1**

|  |  |  |  |
| --- | --- | --- | --- |
| Студент группы ИНБО-01-17 | ИКБО-16-22 Трусов Е.С. | (подпись) | |
| Ассистент | Герасимов С.С. | (подпись) | |
| Отчет представлен | «\_\_\_»\_\_\_\_\_\_\_\_2023г. | |  | |

Москва 2023 г.

Оглавление

[Часть 1 и курсы 3](#_Toc160180870)

[Части 2 и 3 10](#_Toc160180871)

[Ответы на контрольные вопросы 22](#_Toc160180872)

[Вывод 24](#_Toc160180873)

# Часть 1 и курсы

Установите и настройте клиент git на своей рабочей станции (рисунок 1).

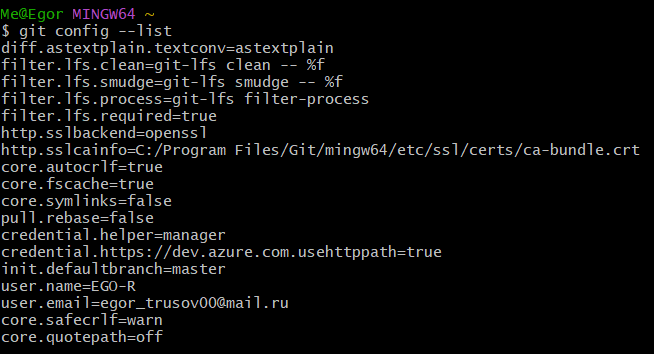


Рисунок 1 – Настройка git

Создайте локальный репозиторий и добавьте в него несколько файлов.



Рисунок 2 – Создание локального репозитория

Внесите изменения в один из файлов.



Рисунок 3 – Внесение изменения в файл

Проиндексируйте изменения и проверьте состояние.

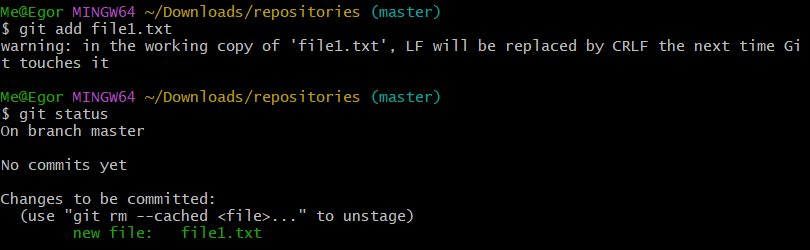


Рисунок 4 – Индексация репозитория

Сделайте коммит того, что было проиндексировано в репозиторий. Добавьте к коммиту комментарий.

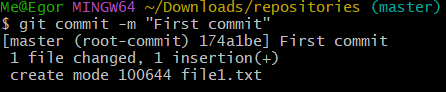


Рисунок 5 – Первый коммит

Измените еще один файл. Добавьте это изменение в индекс git. Измените файл еще раз. Проверьте состояние и произведите коммит проиндексированного изменения.

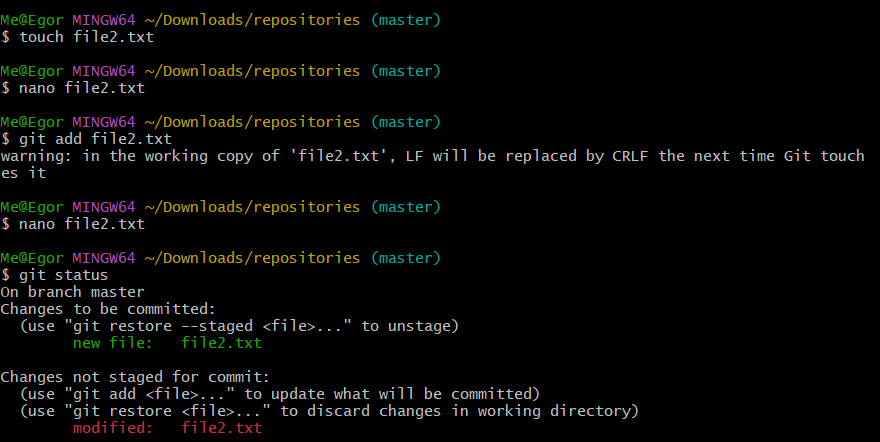


Рисунок 6 - Индексирование

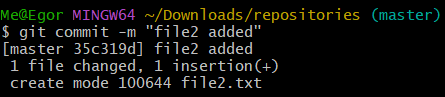


Рисунок 7 - Коммит

Теперь добавьте второе изменение в индекс, а затем проверьте состояние с помощью команды git status. Сделайте коммит второго изменения.

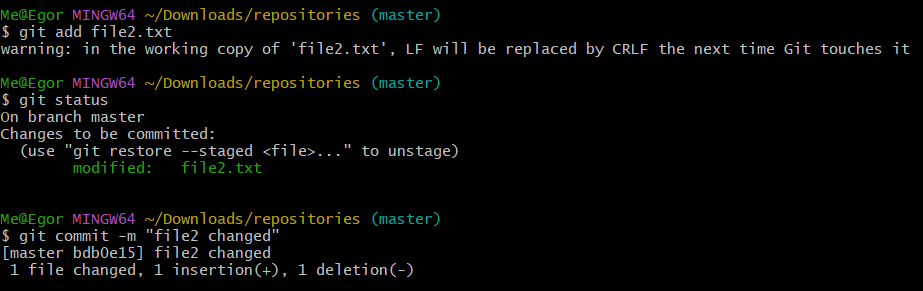


Рисунок 8 – Повторный коммит

Просмотрите историю коммитов с помощью команды git log. Ознакомьтесь с параметрами команды и используйте некоторые из них для различного формата отображения истории коммитов.

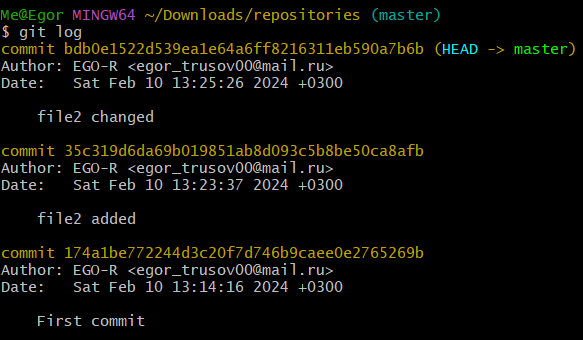


Рисунок 9 – Вывод истории коммитов часть 1

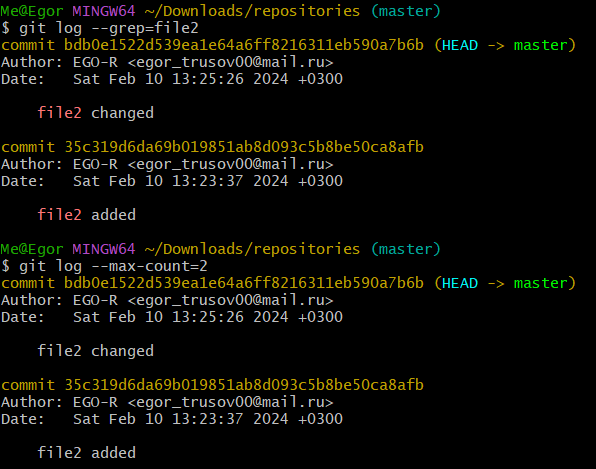


Рисунок 10 – Вывод истории коммитов часть 2

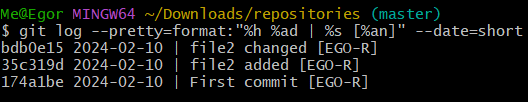


Рисунок 11 – Вывод истории коммитов часть 3

Верните рабочий каталог к одному из предыдущих состояний.

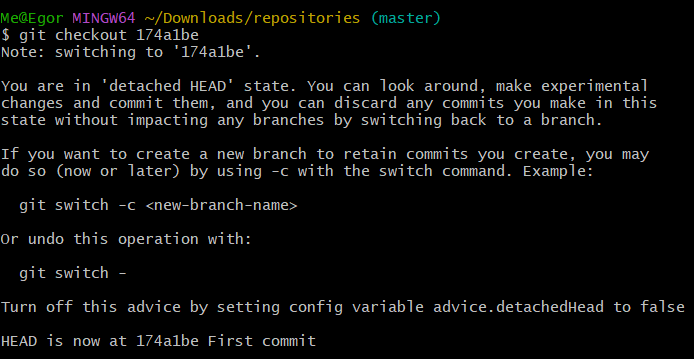


Рисунок 12 – Возвращение состояния к прошлому коммиту

Изучите, как создавать теги для коммитов для использования в будущем.

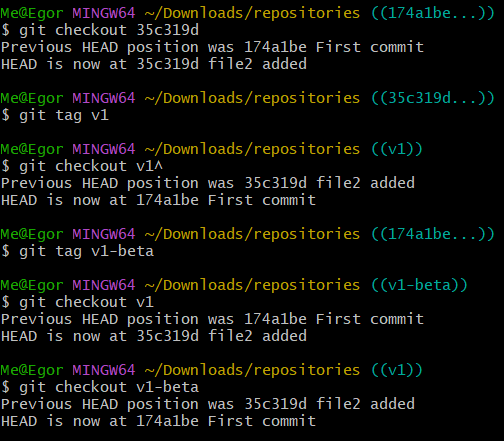


Рисунок 13 – Создание тегов

Отмените некоторые изменения в рабочем каталоге (до и после индексирования).

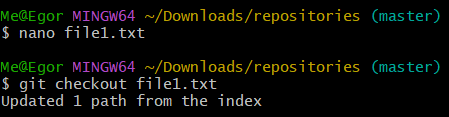


Рисунок 14 – Отмена изменений до индексирования

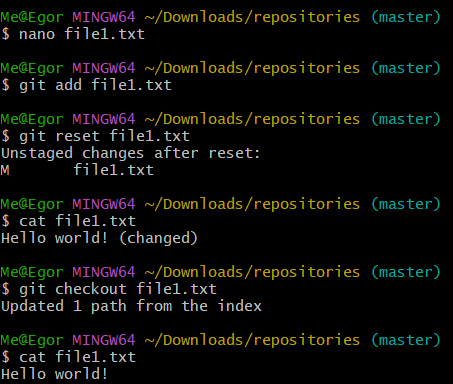


Рисунок 15 – Отмена изменений после индексирования

Отмените один из коммитов в локальном репозитории.

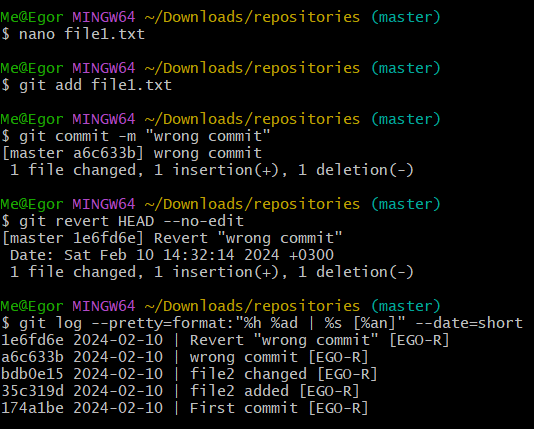


Рисунок 16 – Отмена коммита

Результат прохождения курса:

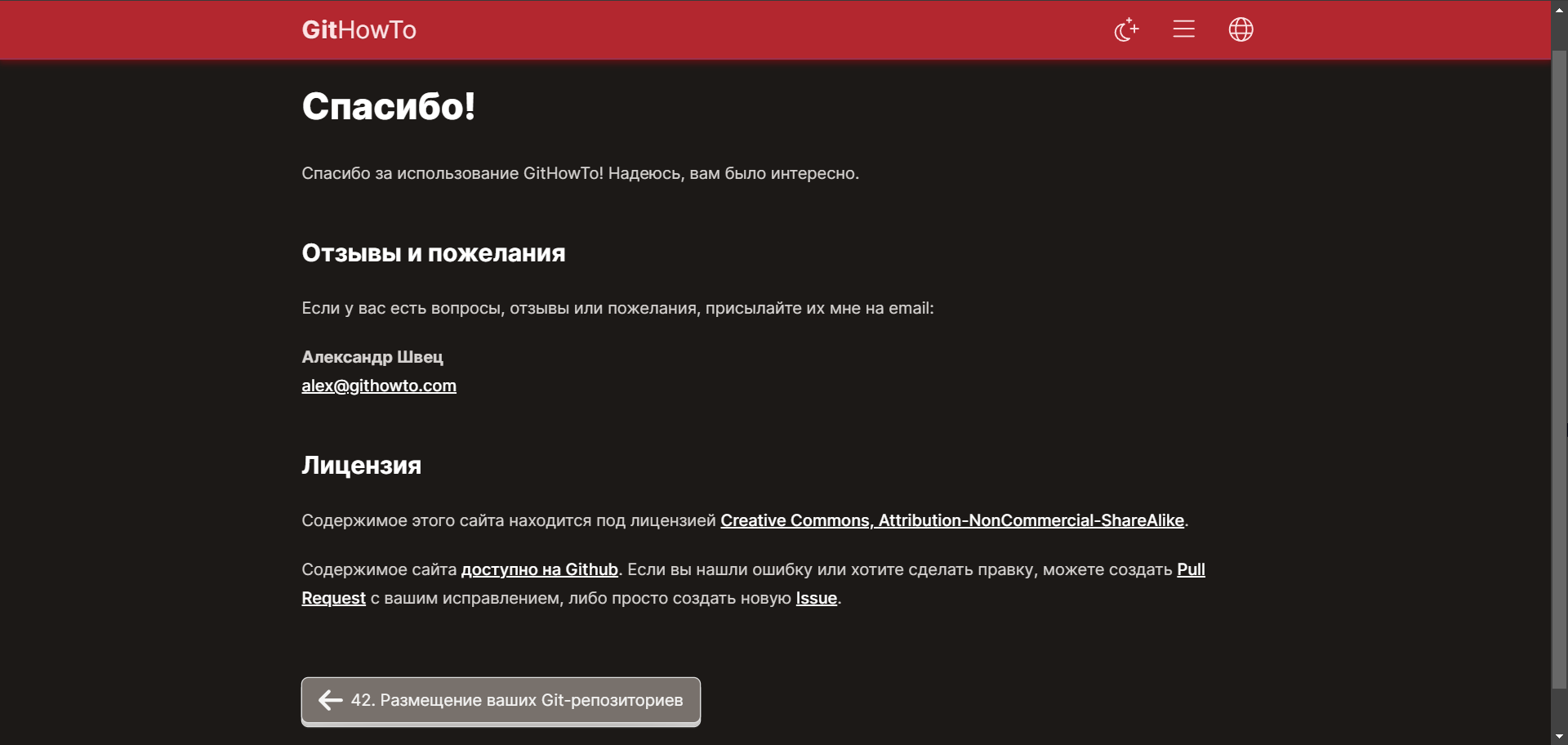


Рисунок 17 – Результат курса

# Части 2 и 3

Создайте аккаунт на GitHub (у кого нет)

Создайте репозиторий на GitHub и на локальной машине, согласно выбранной теме проекта.

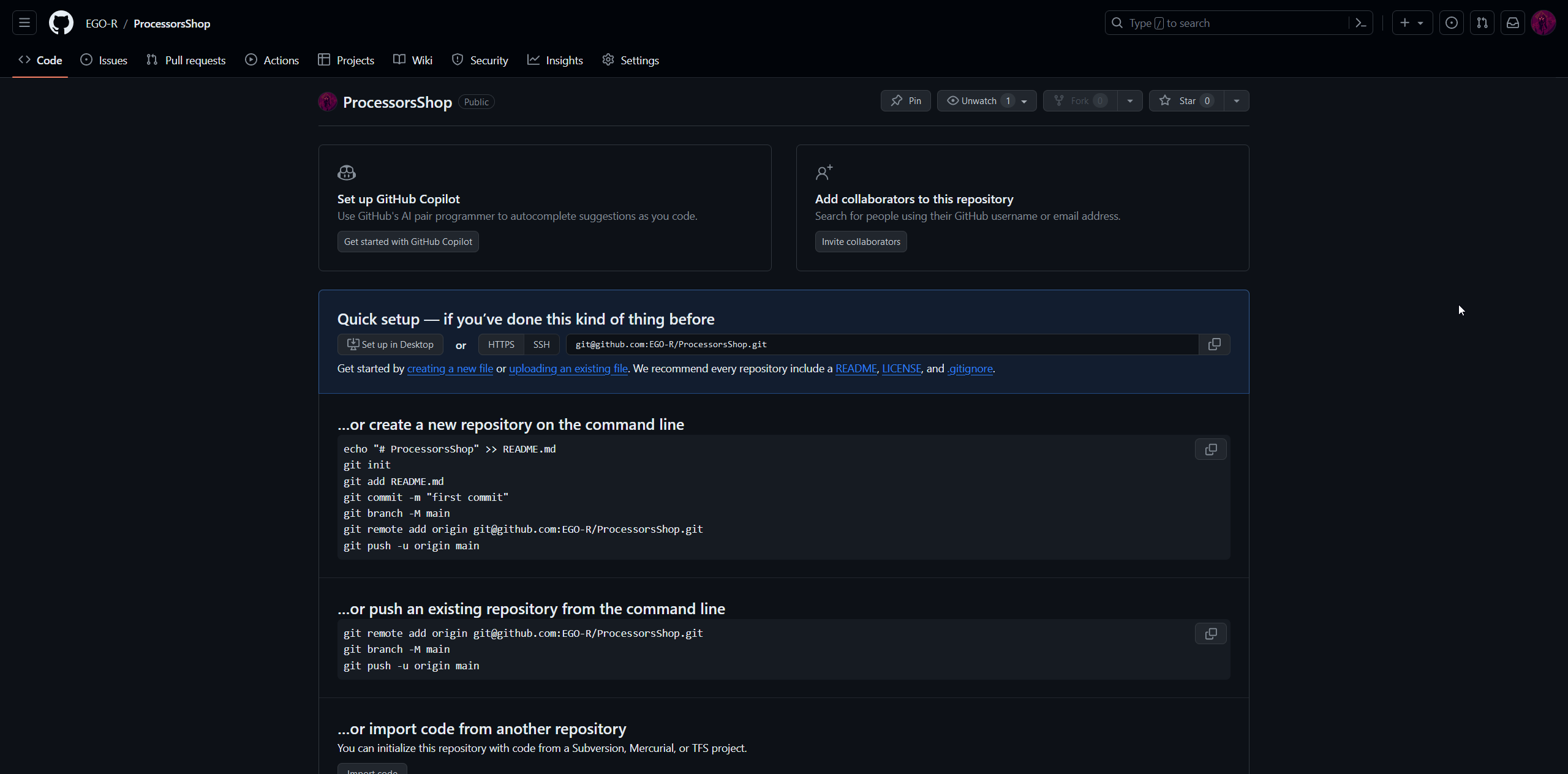


Рисунок 18 – Создание репозитория на GitHub

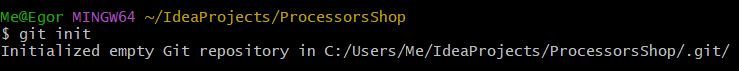


Рисунок 19 – Создание локального репозитория

Создайте несколько файлов на локальной машине при помощи консоли

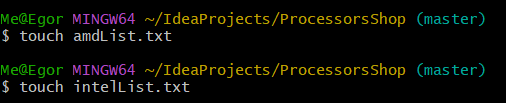


Рисунок 20 – Создание файлов

Создайте SSH-ключ для авторизации

Свяжите репозиторий локальной машины с репозиторием на GitHub при помощи консоли

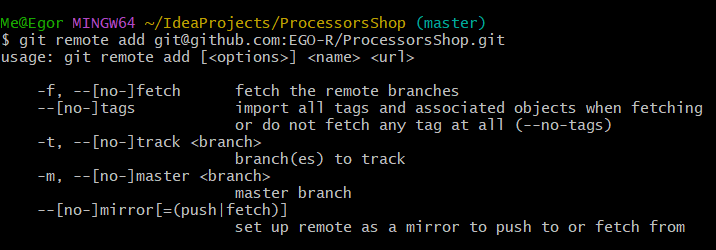


Рисунок 21 – Связывание локального и удалённого репозиториев

Создайте новую ветку в репозитории с помощью команды, произведите в ней какие-нибудь изменения, а после слейте с веткой master

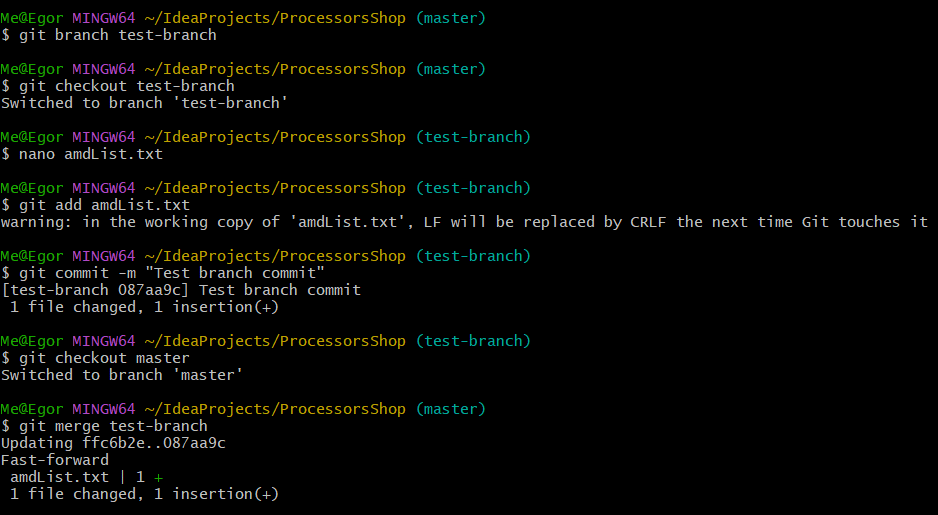


Рисунок 22 – Работа с новой веткой

Выполните цепочку действий в репозитории, согласно вариантам.

Клонируйте непустой удаленный репозиторий на локальную машину

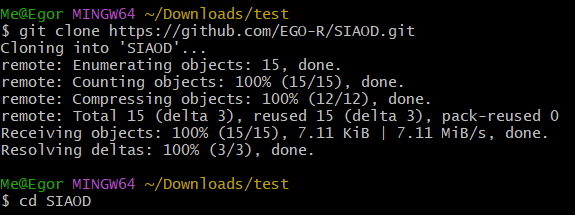


Рисунок 23 – Клонирование удалённого репозитория

Создайте новую ветку и выведите список всех веток

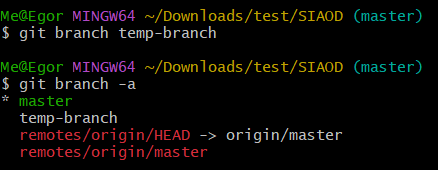


Рисунок 24 – Вывод всех веток

Произведите 3 коммита в новой ветке в разные файлы

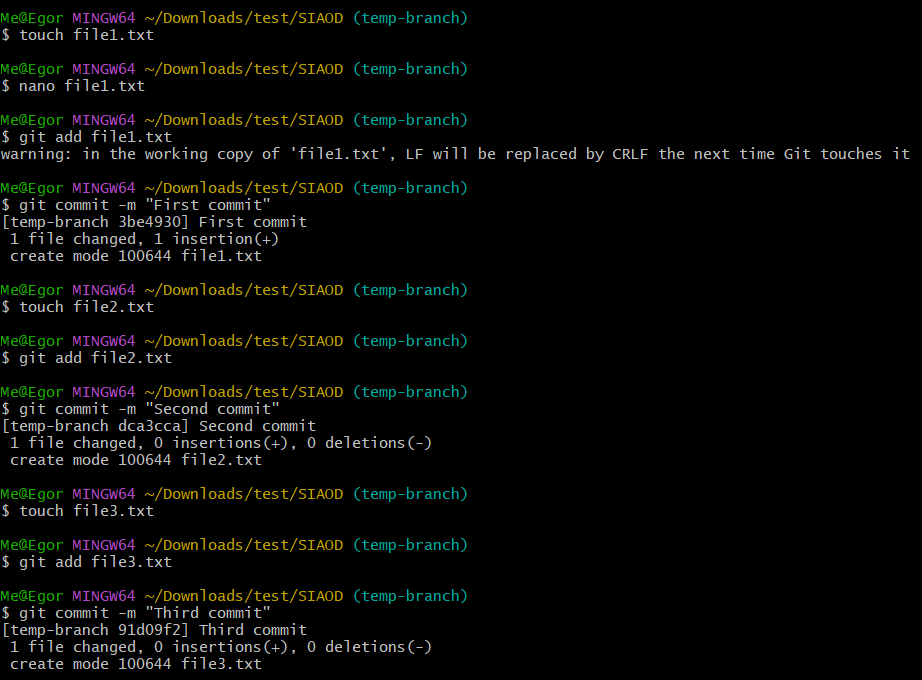


Рисунок 25 – Создание 3 коммитов в разные файлы

Выгрузите изменения в удаленный репозиторий

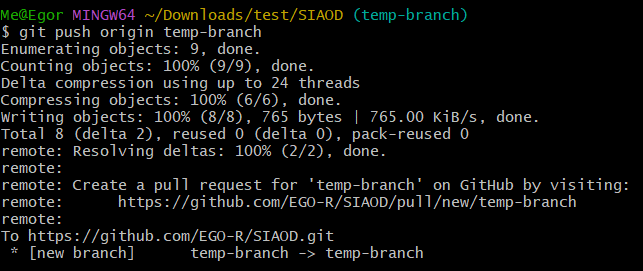


Рисунок 26 – Выгрузка изменений в удалённый репозиторий

Произведите revert предпоследнего коммита в новой ветке

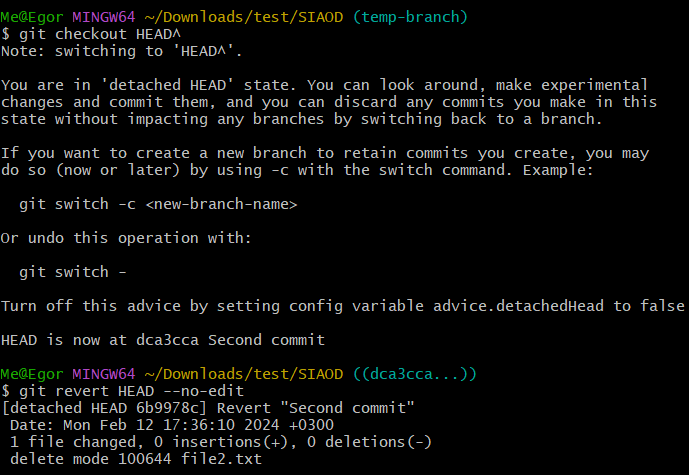


Рисунок 27 – Revert предпоследнего коммита

Выведите в консоли различия между веткой master и новой веткой

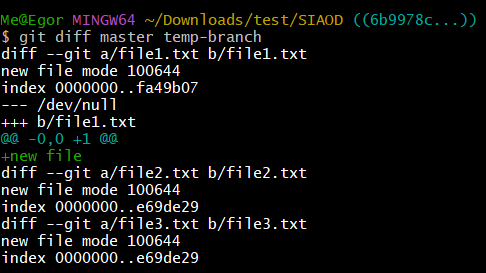


Рисунок 28 – Различия между ветками

Слейте новую ветку с master при помощи merge

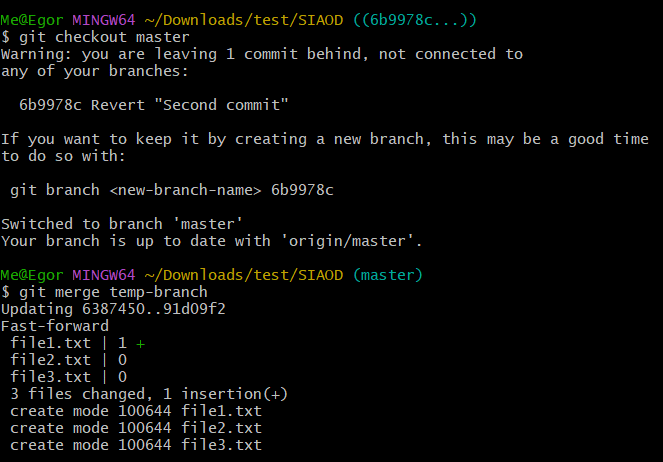


Рисунок 29 – Слияние

Вариант 1 - <https://github.com/liamg/traitor>

Сделайте форк репозитория в соответствии с вашим вариантом



Рисунок 30 – Форк репозитория

Склонируйте его на локальную машину

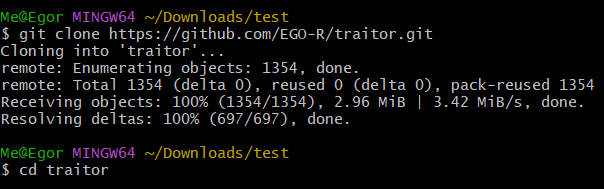


Рисунок 31 – Клонирование удалённого репозитория в локальный

Создайте две ветки branch1 и branch2 от последнего коммита в master'е

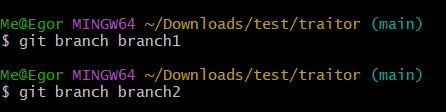


Рисунок 32 – Создание 2 веток

Проведите по 3 коммита в каждую из веток, которые меняют один и тот же кусочек файла

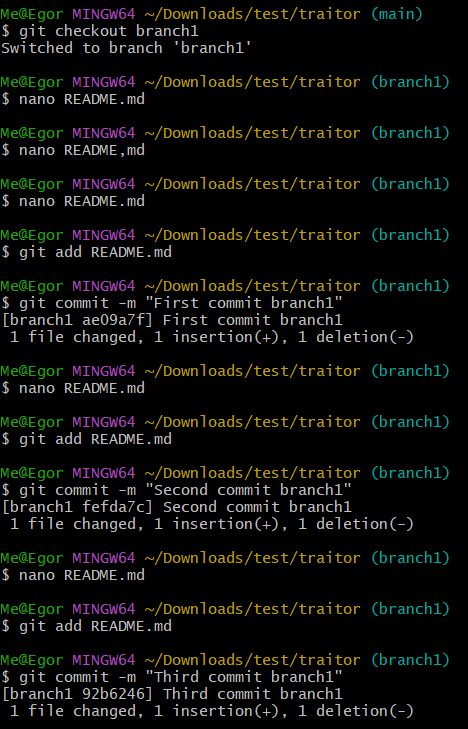


Рисунок 33 – Создание 3 коммитов часть 1

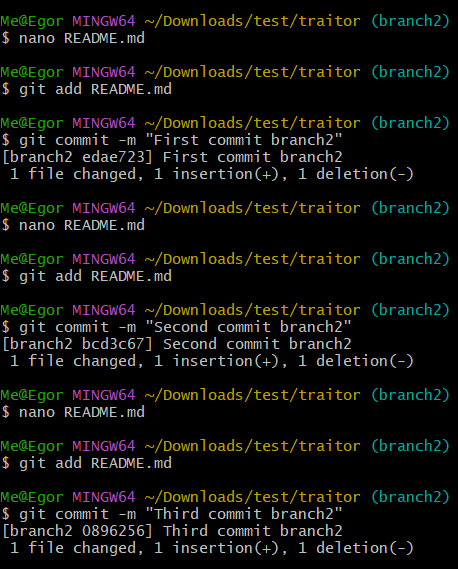


Рисунок 34 – Создание 3 коммитов часть 2

Выполните слияние ветки branch1 в ветку branch2, разрешив конфликты при этом

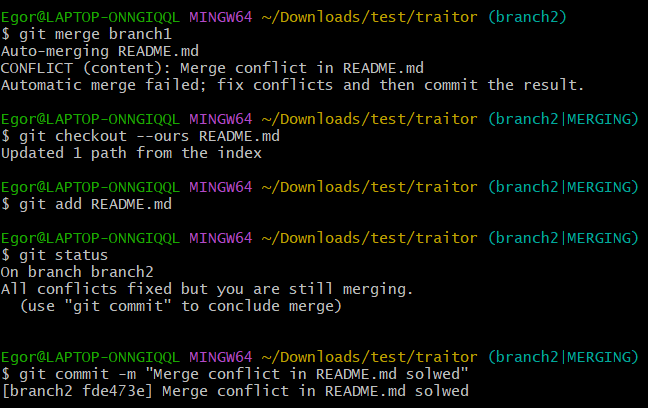


Рисунок 35 – Слияние и разрешение конфликтов

Выгрузите все изменения во всех ветках в удаленный репозиторий

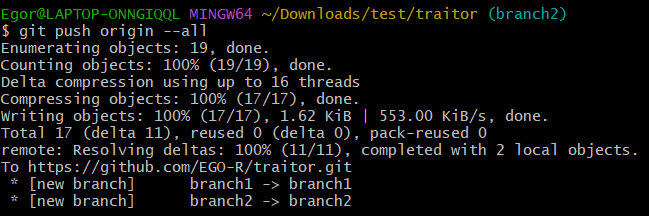


Рисунок 35 – Выгрузка всех изменений в удаленный репозиторий

Проведите еще 3 коммита в ветку branch1

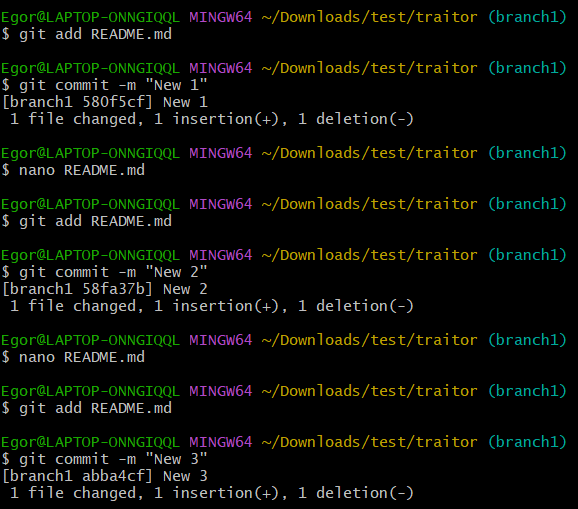


Рисунок 36 – Создание 3 коммитов

Склонируйте репозиторий еще раз в другую директорию

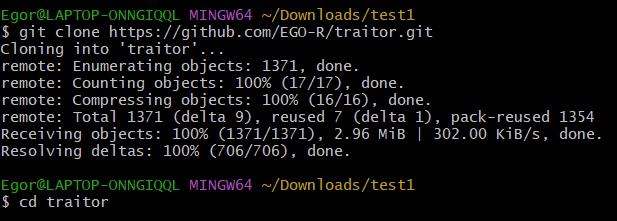


Рисунок 37 – Клонирование удалённого репозитория в другую папку

В новом клоне репозитории сделайте 3 коммита в ветку branch1

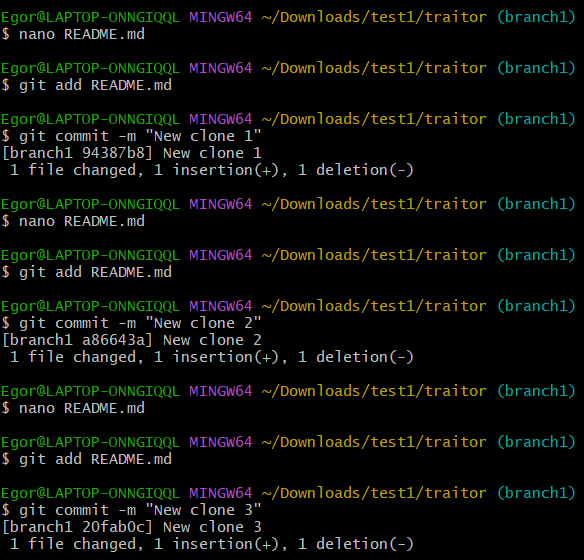


Рисунок 38 – Создание 3 коммитов

Выгрузите все изменения из нового репозитория в удаленный репозиторий

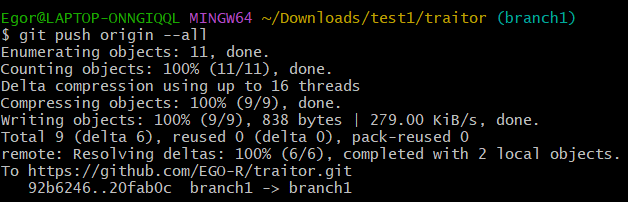


Рисунок 39 – Выгрузка изменений в удалённый репозиторий

Вернитесь в старый клон с репозиторием, выгрузите изменения с опцией –force

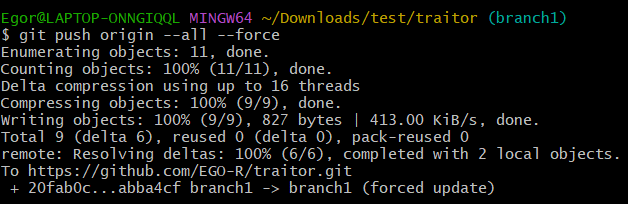


Рисунок 40 – Принудительная выгрузка изменений

Получите все изменения в новом репозитории

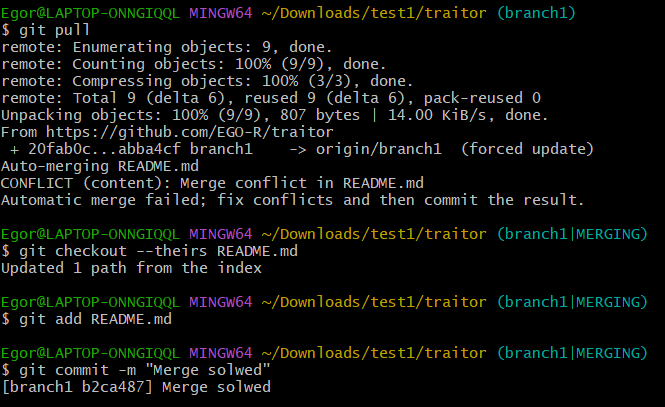


Рисунок 41 – получение всех изменений в новом репозитории

# Ответы на контрольные вопросы

1. Какие существуют типы систем контроля версий? Приведите примеры к каждому типу.

Различают следующие системы контроля версий:

а) локальная система контроля версий, представляющая из себя простейшую базу данных, хранящую записи об изменениях во всех файлах. Её невозможно использовать в команде. Пример: Revision Control System (RCS);

б) централизованная система контроля версий, в которой используется единый сервер, содержащий все версии файлов. Разработчики получают все файлы из этого хранилища. Он позволяет работать над проектом нескольким людям, однако, при выходе сервера из строя доступ к файлам будет полностью утрачен. Пример: Concurrent Versions System (CVS), Subversion;

в) распределённая система контроля версий. Разработчики выкачивают репозиторий целиком, работают с ним, и обновляют существующий. Такой тип позволяет и работать нескольким людям одновременно, и отказоустойчив так как у каждого разработчика есть копия репозитория. Пример: Git, Mercurial.

2. К какому типу систем контроля версий относится Git?

Git – распределённая система контроля версий.

3. Что такое репозиторий Git?

Репозиторий - место, где хранятся и поддерживаются какие-либо данные. Git способен отслеживать все изменения, которые происходили в нём.

4. Что такое коммит?

Коммит – это операция, в которой содержатся записи об изменениях в файлах репозитория и другая дополнительная информация. С помощью этого, можно определить, какие изменения были сделаны, отменить их или восстановить.

5. Что такое ветка в репозитории Git?

Ветка в Git – указатель на последний коммит в определённой ветке. Веток может быть множество. Главная ветка – master (как правило). С помощью ветвления разработчики могут работать независимо над разными версиями кода, не мешая при этом друг другу.

6. Что такое тег в репозитории Git?

Тег – специальная ссылка на конкретный коммит. Он используется для указания состояние кода в определённый момент. С помощью тегов можно быстро вернутся к нужному коммиту в любой момент. Как правило, в теге указывается версия проекта.

7. Что такое слияние двух веток?

Слияние двух веток – внесение изменений, сделанных в одной ветке в другую. Например, в ветке feature было создано обновление программы. Чтобы это обновление добавилось в основную ветку, её необходимо слить с веткой master.

8. Что такое конфликт в Git? Как его решить и почему они бывают?

Конфликт – ошибка, возникающая при слиянии веток, когда обе ветки меняли одну и ту же строку в каком-либо файле. Для решения конфликта, разработчику необходимо вручную выбрать нужную версию.

# Вывод

Выводы: в ходе практической работы были получены навыки по работе с командной строкой и git’ом. Все задачи практической работы были выполнены, ошибок получено не было.