**Практическая работа**

**Изучение функционала GitHub**

**Цель работы**: Научиться пользоваться системой управления версиями кода **GitHub**.

**Выполнение работы:**

**GitHub** это система управления проектами и версиями кода, а также платформа социальных сетей, созданная для разработчиков. Онпозволяет вам работать совместно с другими людьми по всему миру, планировать свои проекты и отслеживать свою работу. Когда разработчики создают новый проект, они всегда продолжают вносить обновления в код. Даже после запуска проектов им всё равно нужно обновлять версии, исправлять ошибки, добавлять новые функции и т. д.

Система контроля версий помогает отслеживать изменения, внесённые в базу кода. Более того, он записывает, кто внёс изменения и может восстановить стёртый или изменённый код.

**Репозиторий**

Репозиторий или хранилище — это каталог, в котором хранятся файлы вашего проекта. Он может быть расположен в хранилище GitHub или в локальном хранилище на вашем компьютере. Вы можете хранить файлы кодов, изображения, аудио или всё, что связано с проектом, в хранилище.

**Ветка**

Ветка это копия вашего репозитория. Вы можете использовать ветку, когда хотите сделать разработку изолированно.

Работа с веткой не повлияет на центральное хранилище или другие ветки. Если вы сделали эту работу, вы можете объединить свою ветку с другими ветками и центральным репозиторием, используя запрос на извлечение.

**Коммит(фиксация изменений)**

Коммит представляет собой состояние репозитория в определенный момент времени. Это похоже на снапшот, к которому мы можем вернуться и увидеть состояние объектов на определенный момент времени.

**Основные команды:**

git init # *создать новый проект в текущей директории*

git clone https://github.com/cyberspacedk/Git-commands.git

# *клонировать удаленный репозиторий в одноименную директорию*

git status # *показать состояние репозитория (отслеживаемые, изменённые, новые файлы и пр.)*

git add text.txt # *добавить в индекс указанный файл (был изменён, был удалён или это новый файл)*

git reset readme.txt # *убрать из индекса изменения указанного файла (в рабочей директории изменения сохранятся)*

git commit -m "Name of commit" # *зафиксировать в коммите проиндексированные изменения (закоммитить), добавить сообщение*

git revert HEAD --no-edit # *создать новый коммит, отменяющий изменения последнего коммита без запуска редактора сообщения*

git checkout b9533bb # *переключиться на коммит с указанным хешем (переместить HEAD на указанный коммит, рабочую директорию вернуть к состоянию, на момент этого коммита)*

git log master # *показать коммиты в указанной ветке*

git branch # *показать список веток*

git merge feature # *влить в активную ветку изменения из ветки feature*

git stash # *временно сохранить незакоммиченные изменения и убрать их из рабочей директории*

git pull origin master # *влить изменения с удалённого репозитория (только указанная ветка)*

git rebase master # *перенести все коммиты (создать их копии) активной ветки так, будто активная ветка ответвилась от master на нынешней вершине master (часто вызывает конфликты)*

**Ход работы:**

Подготовка и введение. Регистрируемся на GitHub; Устанавливаем продукты Git; Настраиваем пользователя.

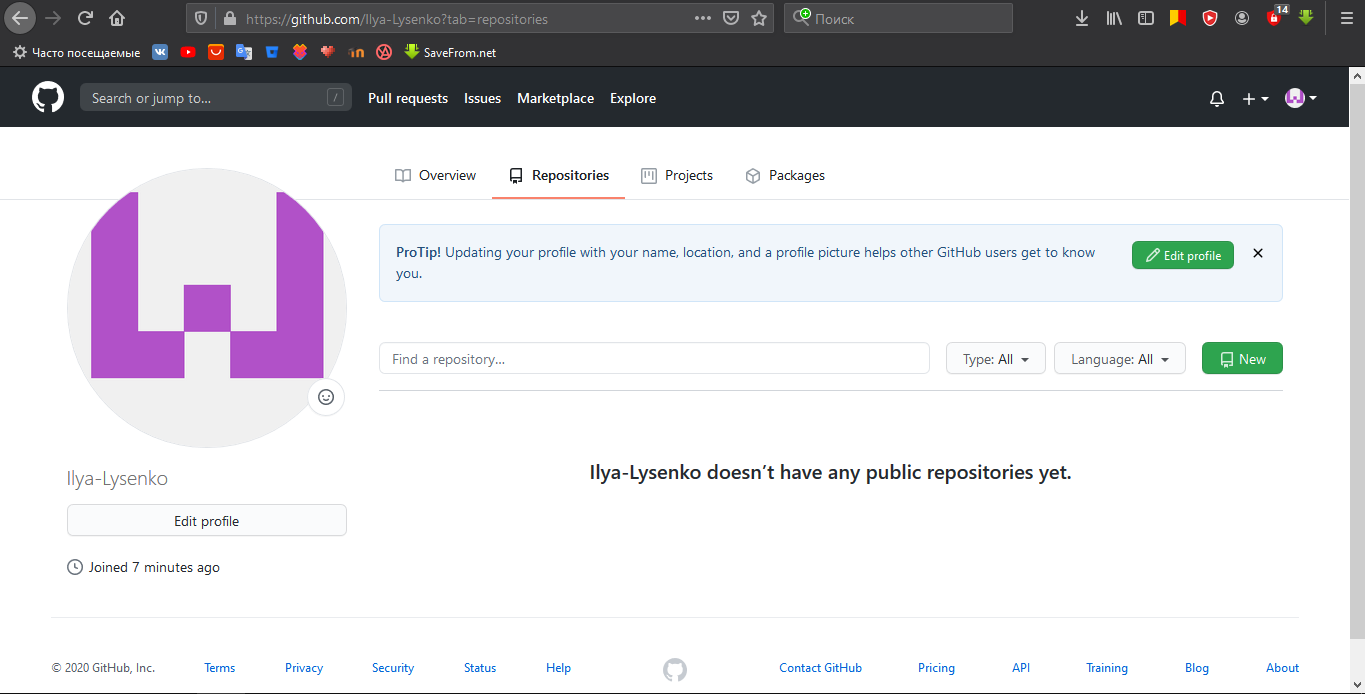


Рис. 1 – Завершение регистрации

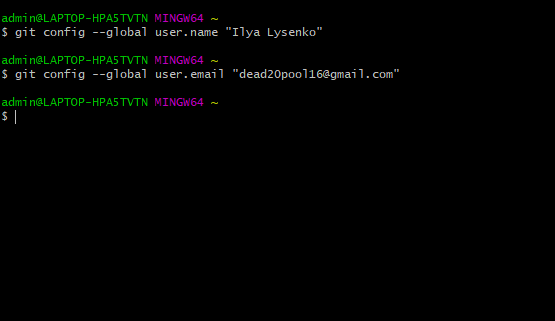


Рис. 2 – Настройка пользователя

Наблюдаем проделанные настройки.

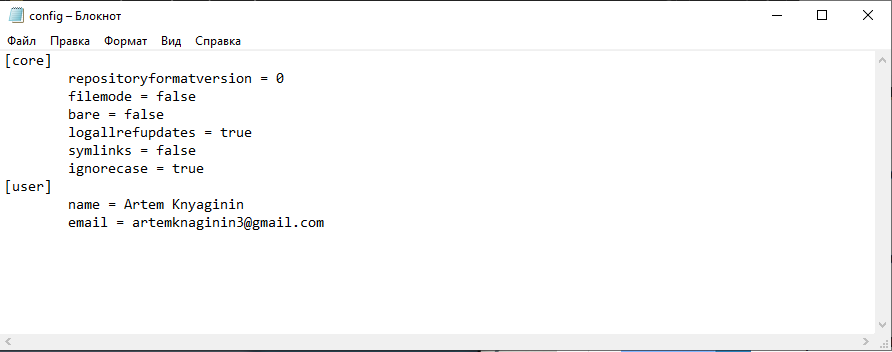


Рис. 3 – Просмотр настроек

Работаем с commit, встроенный редактор gitk

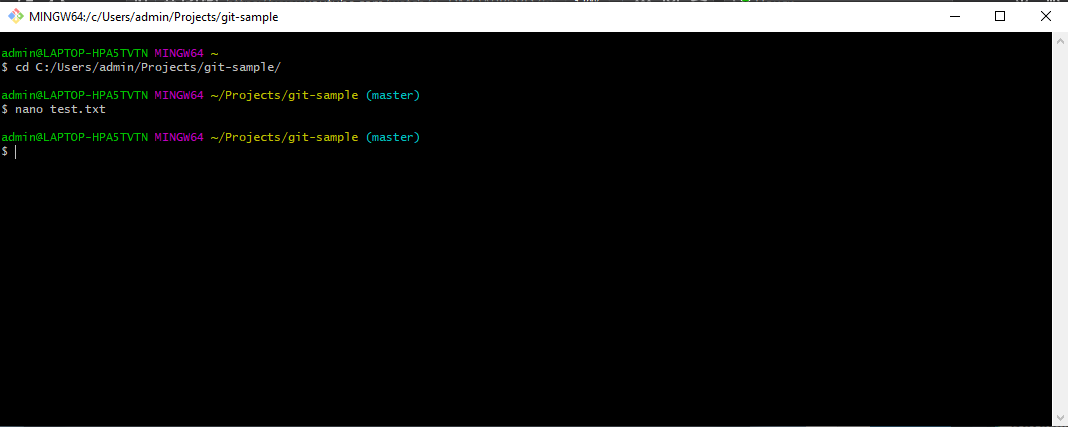


Рис. 4 – Работа с текстовым файлом

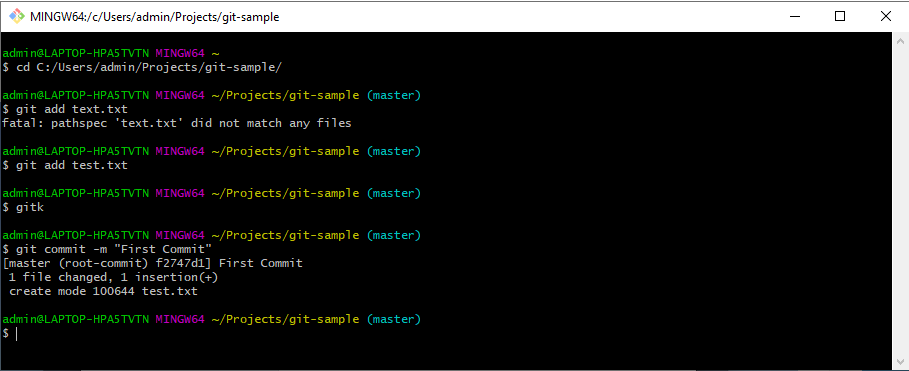


Рис. 5 - Индексируем

После выполнения git commit –m ‘First Commit’ имеем:

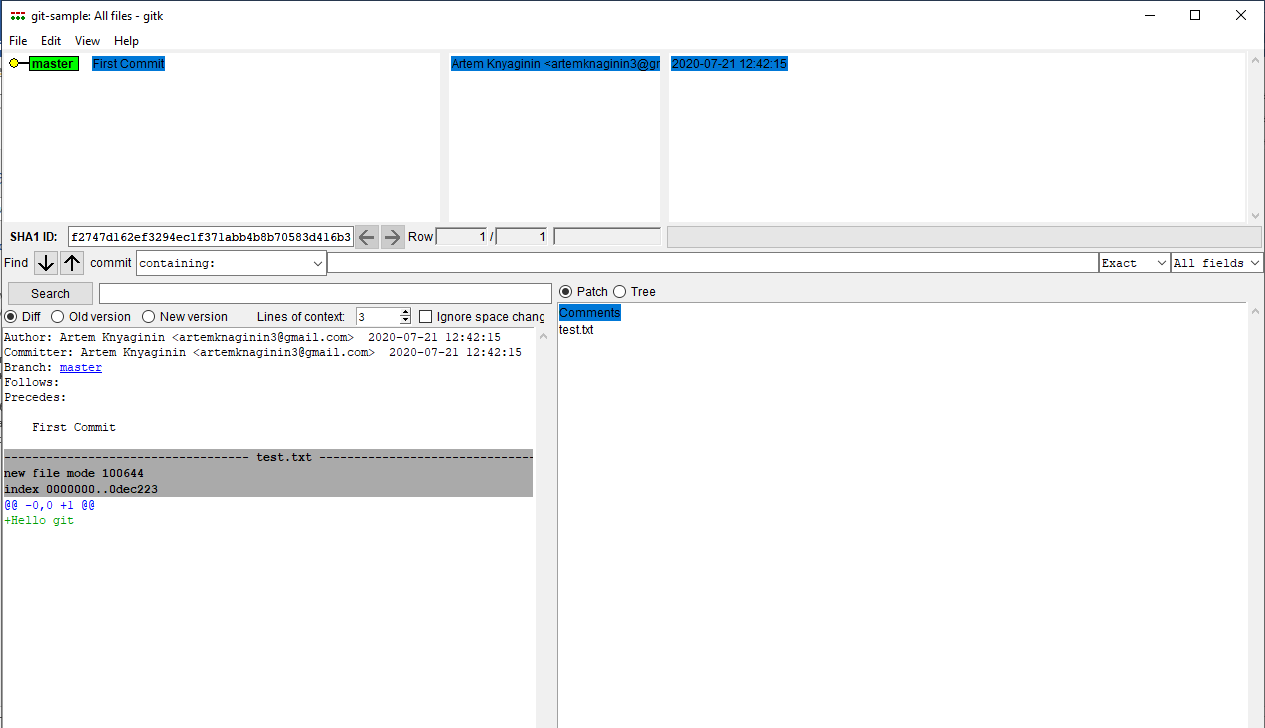


Рис. 6 – Интерфейс gitk

Знакомимся с git status:

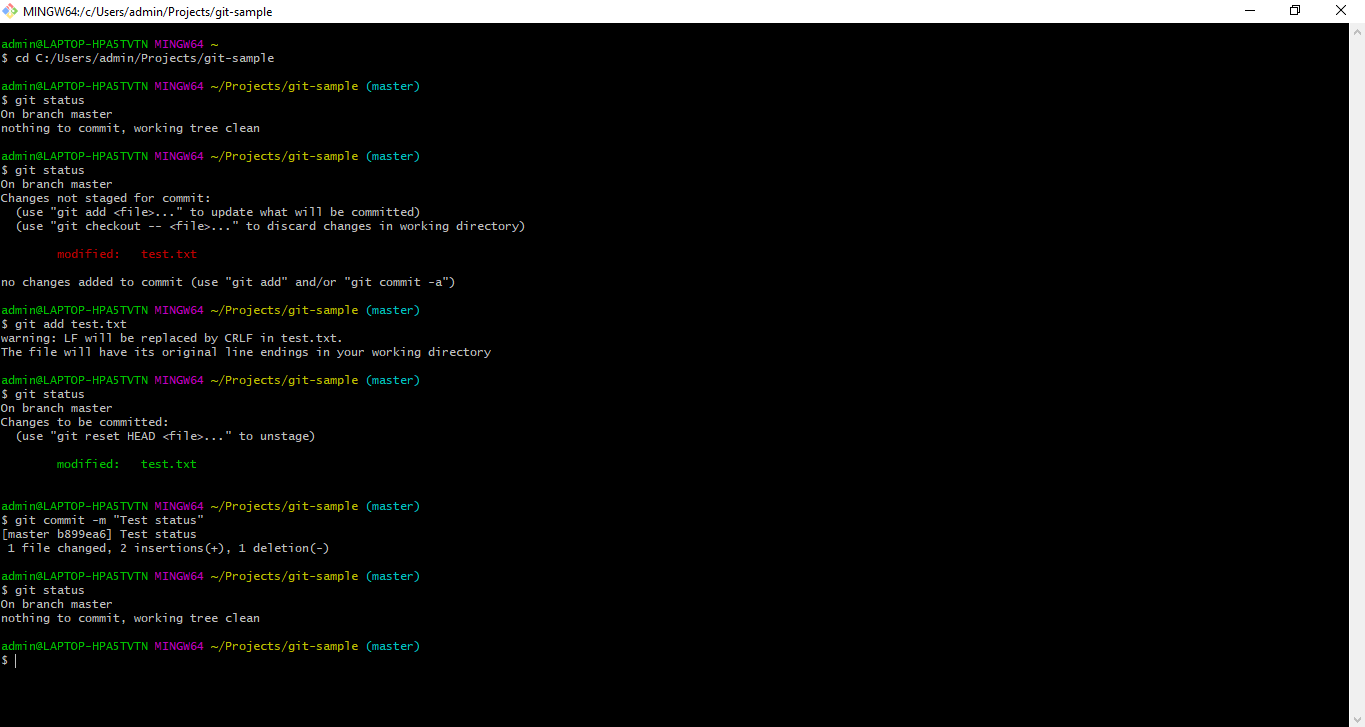


Рис. 7 – git status

Индексация файлов, выполняется через запрос git add:

Был изменен файл text.txt и изменен формат файла index.txt на index.html

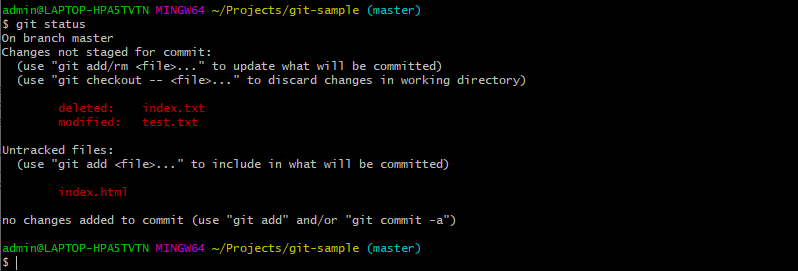


Рис. 8 – Измененные файлы

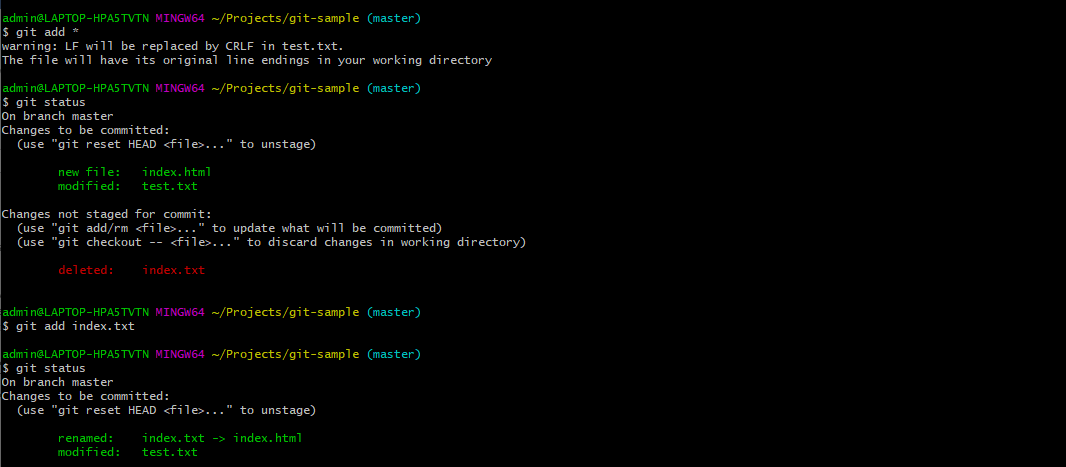


Рис. 9 – Индексация файлов

История «коммитов» git log; Использование флагов.

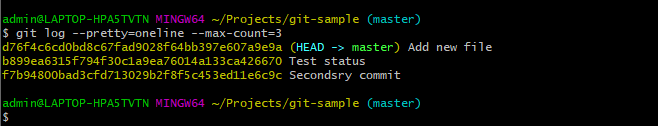


Рис. 10 – Вывод 3 строчек логов

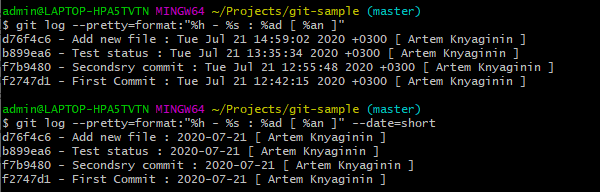


Рис. 11 – Вывод логов с разными флагами

Команда git checkout; переход по хэшам коммитов.

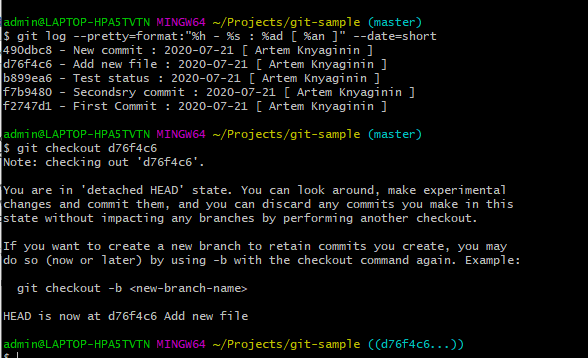


Рис. 12 – Работа с checkout

Отмена индексированных файлов; git reset и git checkout

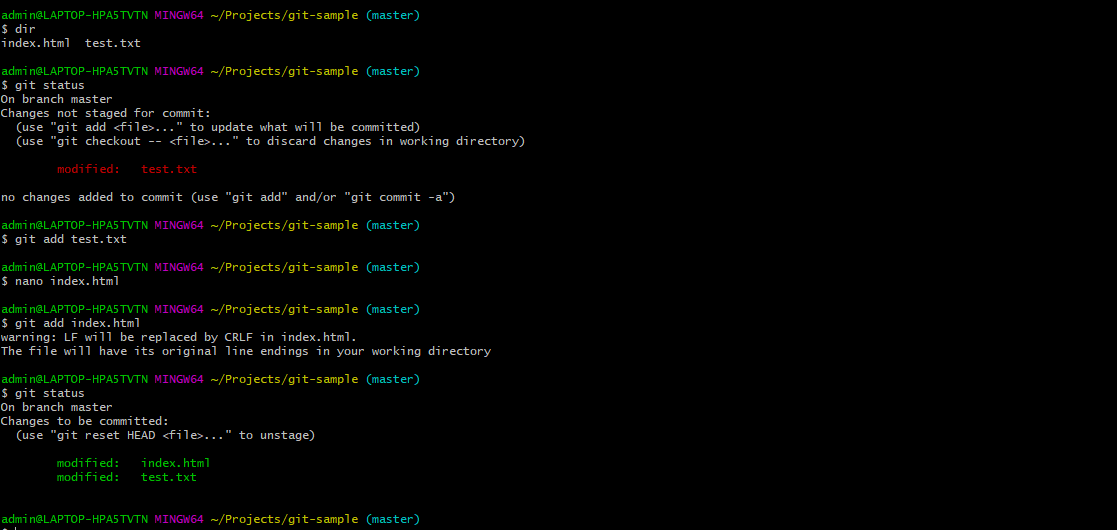


Рис. 13 – Индексируем файлы

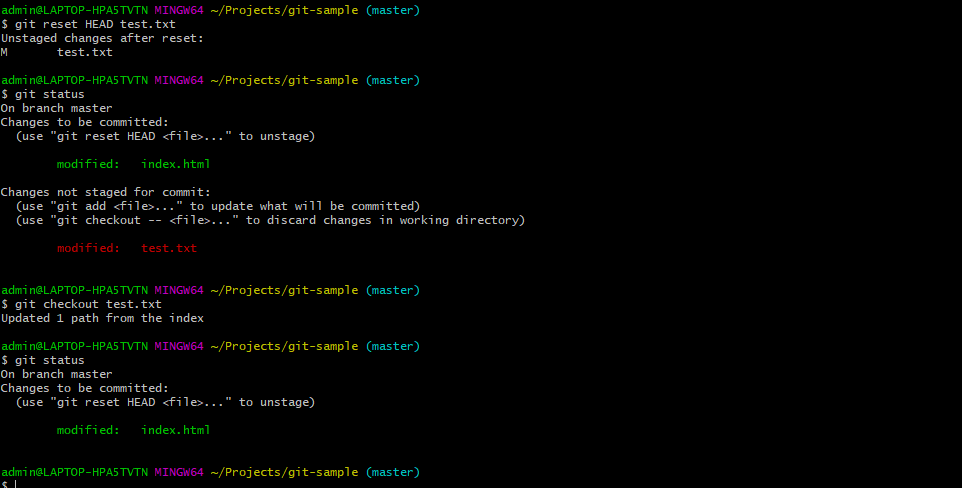


Рис. 14 – Отменяем индексацию

Revert - отмена коммита. Revert по логам.

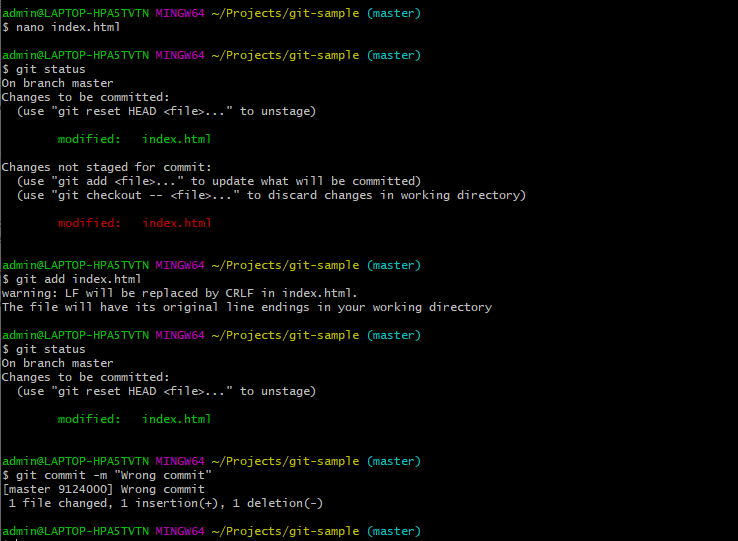


Рис. 15 – Делаем commit для изменения



Рис. 16 – Отменяем commit командой revert



Рис. 17 – Результат по логам

Решение простого конфликта

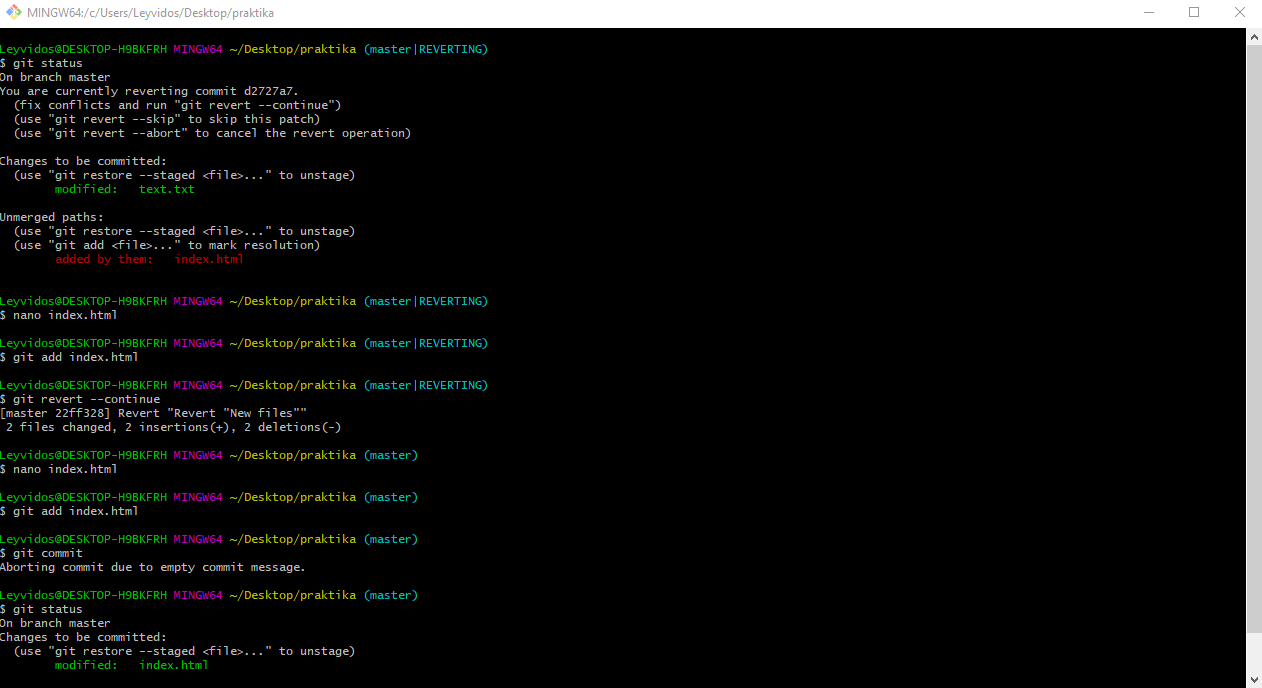


Рис. 18 – Решение конфликта

Ветки и их применение

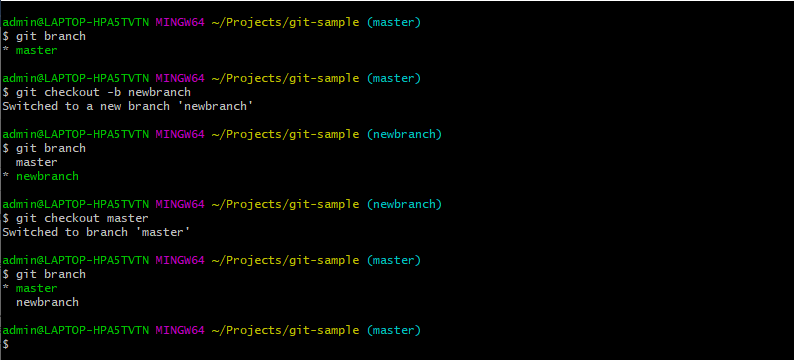


Рис. 19 – Переключение между ветвями

Слияние веток и решение конфликтов слияния

Создаем файл devfile.txt в ветке newbranch. Попробуем слить ветку master и ветку newbranch

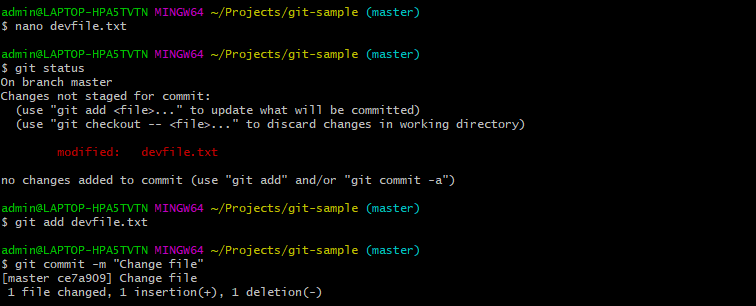


Рис. 20 – Изменяем файл devfile.txt

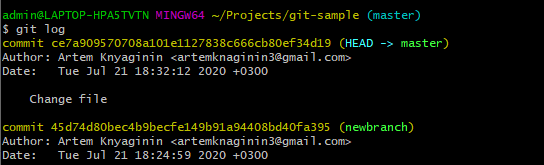


Рис. 21 – Проверяем логи



Рис. 22 – Проверяем файл

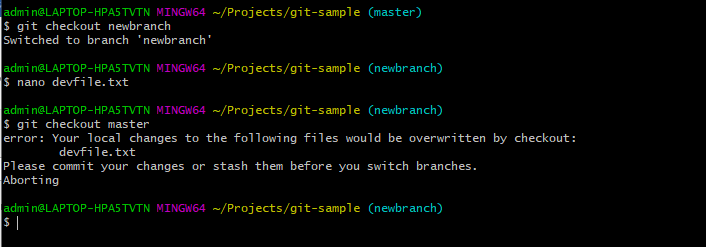


Рис. 23 – Предупреждение об изменениях

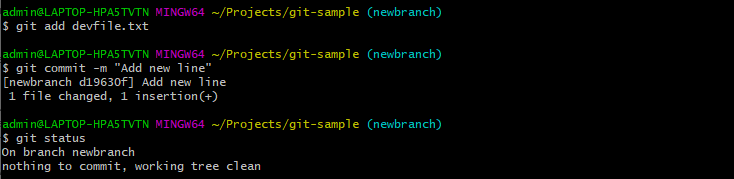


Рис. 24 – Коммитим изменения

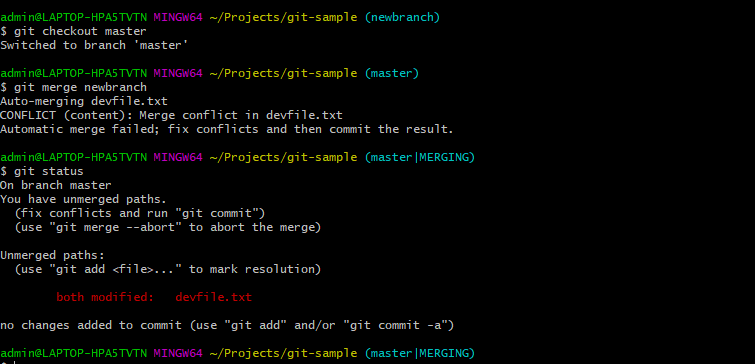


Рис. 25 – Имеем конфликт

Решаем с помощью merge

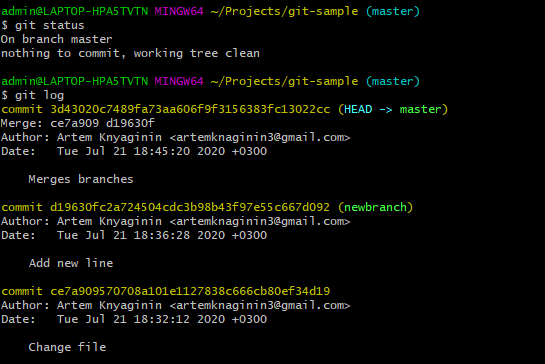


Рис. 26 – Конечный результат

Работа с удаленным репозиторием.

Имеем файл на ГитХабе, который был удален на локальном репозитории.

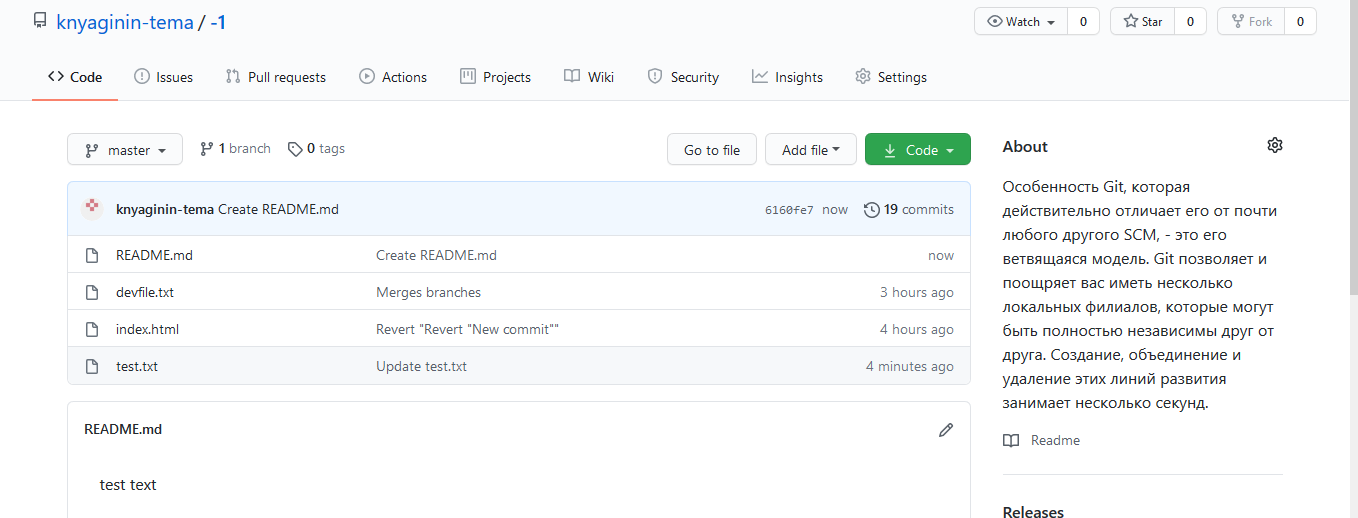


Рис. 27 – Файл test.txt

Пользуемся командой git pull, чтобы получить изменения с удаленного репозитория



Рис. 28 – Обновляем локальный репозиторий

Работа со stash

Имеем измененный файл на ГитХабе:

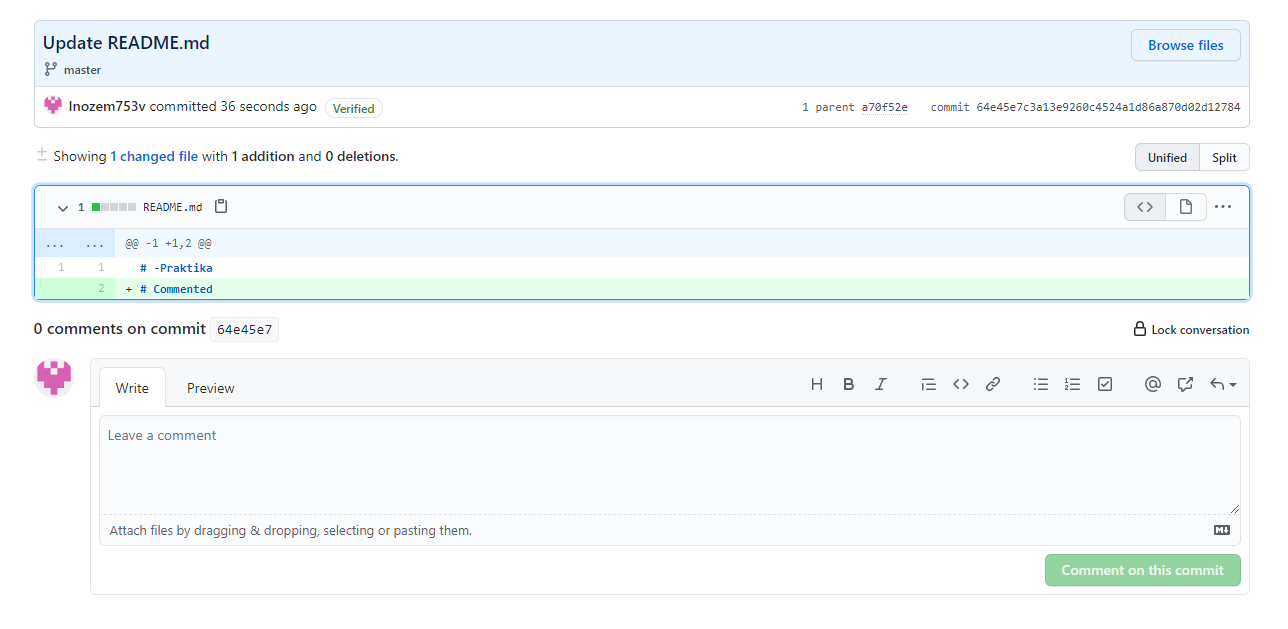


Рис. 29 – Файл README изменен

Имеем неиндексированное изменение в локальном репозитории, помести его в stash:

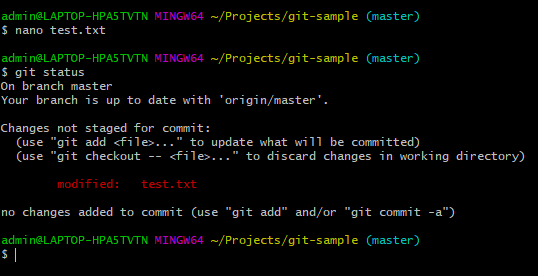


Рис. 30 – Локальные изменения

Получаем изменения из удаленного репозитория

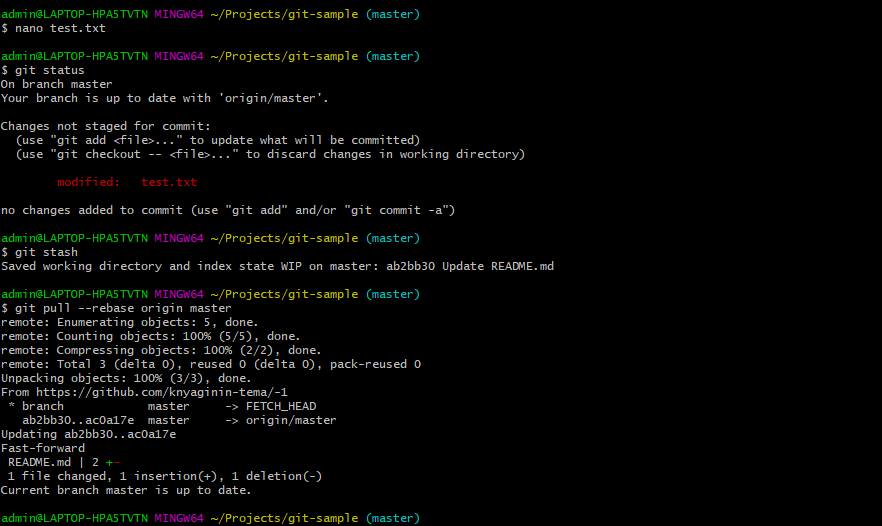


Рис. 31 – Работа с командами

Вытаскиваем из stash изменения файла

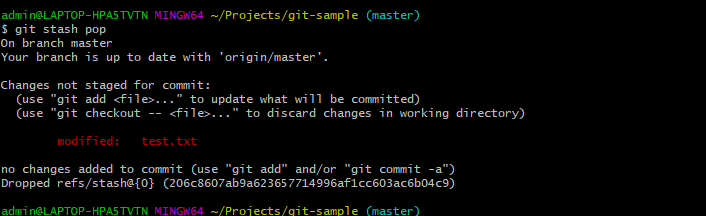


Рис. 32 – Измененный файл

Завершаем работу с локальным изменением и отправляем на удаленный репозиторий

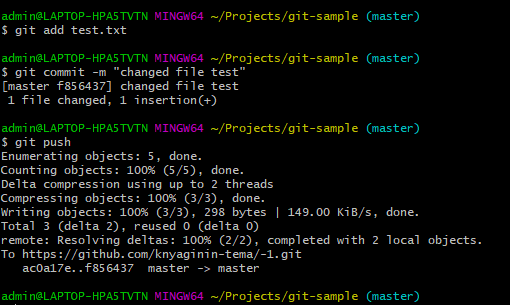


Рис. 33 – Завершение работы