

**Оглавление**

[Часть 1. Основные команды Git 4](#_Toc190856257)

[1. Установите и настройте клиент Git на своей рабочей станции. 4](#_Toc190856258)

[2. Настройка Git 4](#_Toc190856259)

[3. Начало работы – создание папки и файлов 5](#_Toc190856260)

[4. Создание репозитория 6](#_Toc190856261)

[5. Добавление файла в репозиторий и добавление первого коммита 6](#_Toc190856262)

[6. Индексация изменений 6](#_Toc190856263)

[7. Коммиты нескольких изменений 7](#_Toc190856264)

[8. Просмотр истории коммитов 8](#_Toc190856265)

[9. Получение старых версий 11](#_Toc190856266)

[10. Отмена локальных изменений (до индексации) 13](#_Toc190856267)

[11. Отмена локальных изменений (после индексации и до коммита) 15](#_Toc190856268)

[12. Отмена коммита. 15](#_Toc190856269)

[Часть 2. Системы управления репозиториями 16](#_Toc190856270)

[1. Создание аккаунта на GitHub 16](#_Toc190856271)

[2. Создание SSH-ключа для авторизации 16](#_Toc190856272)

[3. Создание нового репозитория для своего проекта 17](#_Toc190856273)

[4. Связываем локальный и удаленный репозитории 18](#_Toc190856274)

[5. Создание веток и переключение между ними 18](#_Toc190856275)

[6. Слияние веток 19](#_Toc190856276)

[7. Выполните цепочку действий в репозитории, согласно вариантам. Вариант 3. 20](#_Toc190856277)

[2) Создайте новую ветку и выведите список всех веток 20](#_Toc190856278)

[3) Произведите 5 коммитов в новой ветке в разные файлы 20](#_Toc190856279)

[4) Выгрузите изменения в удаленный репозиторий 21](#_Toc190856280)

[5) Произведите revert 2-го и 4-го коммита в новой ветке](#_Toc190856281)

[6) Выведите в консоли различия между веткой master и новой веткой 22](#_Toc190856282)

[7) Слейте новую ветку с master при помощи merge](#_Toc190856283)

[Часть 3. Работа с ветвлением и оформление кода 23](#_Toc190856284)

[1. Форк репозитория 23](#_Toc190856285)

[2. Клонирование репозитория на локальную машину 23](#_Toc190856286)

[3. Создание двух веток в deprecation-note 23](#_Toc190856287)

[4. 3 коммита в каждую из веток, которые меняют один и тот же кусочек файла 24](#_Toc190856288)

[5. Слияние ветки branch1 в ветку branch2 с разрешением конфликтов 26](#_Toc190856289)

[6. Выгрузка всех изменений во всех ветках в удаленный репозиторий 28](#_Toc190856290)

[7. Выполнение 3-х коммитов в branch1 29](#_Toc190856291)

[8. Повторное клонирование репозитория в другую директорию 29](#_Toc190856292)

[9. Выполнение 3-х коммитов в ветку branch1 30](#_Toc190856293)

[10. Выгрузка всех изменений в удаленный репозиторий 30](#_Toc190856294)

[11. Выгрузка изменения из старого репозитория с опцией –force 31](#_Toc190856295)

[12. Получение всех изменения в новом репозитории 31](#_Toc190856296)

[Ответы на 8 любых вопросов: 32](#_Toc190856297)

[9. Как отменить слияние веток, если произошел конфликт? 32](#_Toc190856298)

[10. Для чего нужен .gitignore? 32](#_Toc190856299)

[11. Что делает команда git status? 32](#_Toc190856300)

[12. Что делает команда git add? 32](#_Toc190856301)

[15. Что делает команда git show? 32](#_Toc190856302)

[16. Что делает команда git stash? 32](#_Toc190856303)

[Вывод 33](#_Toc190856304)

# Часть 1. Основные команды Git

## Установите и настройте клиент Git на своей рабочей станции.

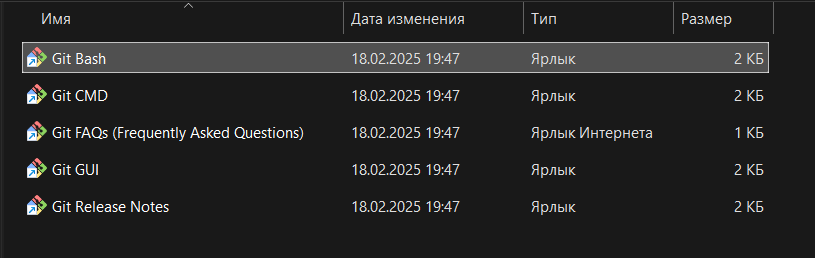


Рисунок 1 – установленный Git на рабочей станции

## Настройка Git

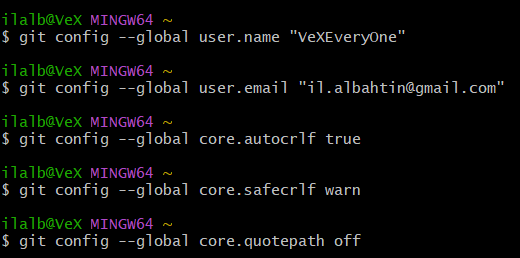


Рисунок 2 – Указание логина и почты для настройки Git

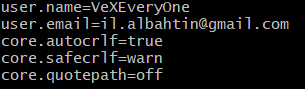


Рисунок 3 – Проверка настройки Git

## Начало работы – создание папки и файлов

## 

## 

## 

## 

Рисунки 4-7 – Создание директории и внесение изменений в файлы директории

## Создание репозитория

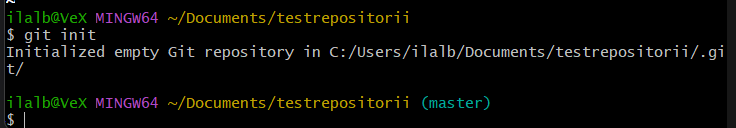


Рисунок 8 – Инициализация репозитория

## Добавление файла в репозиторий и добавление первого коммита

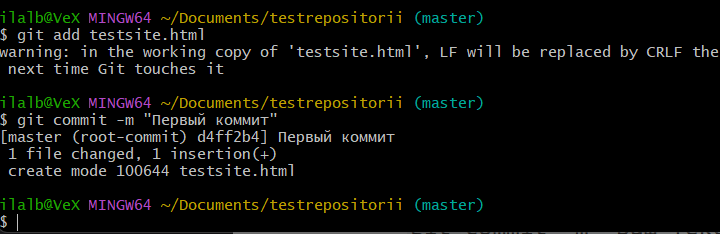


Рисунок 9 – Сделанный коммит с комментарием к нему

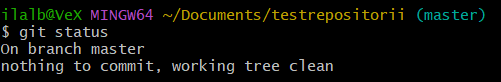
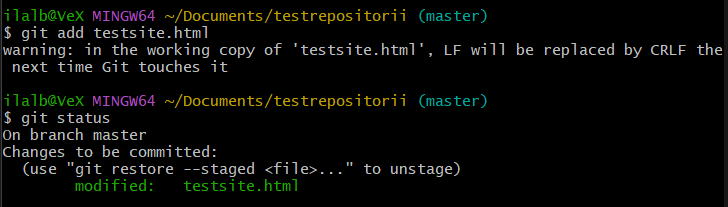


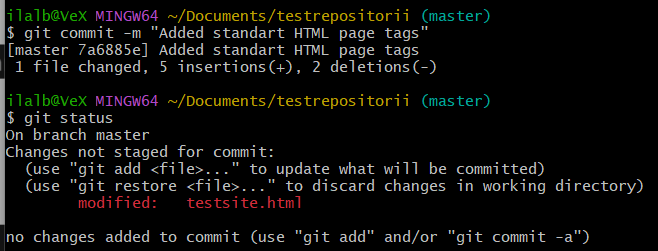
Рисунок 10 - Проверка состояния ветки

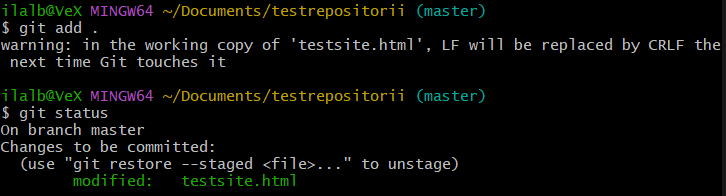
## Индексация изменений

Рисунок 11 – Внесение изменений в файл

## Коммиты нескольких изменений







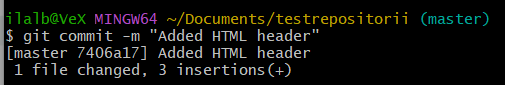


Рисунок 12-15 – создание двух коммитов с изменениями файла proekt.html

## Просмотр истории коммитов

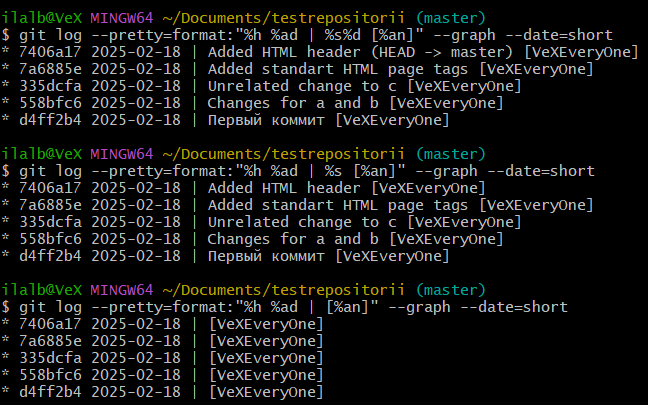
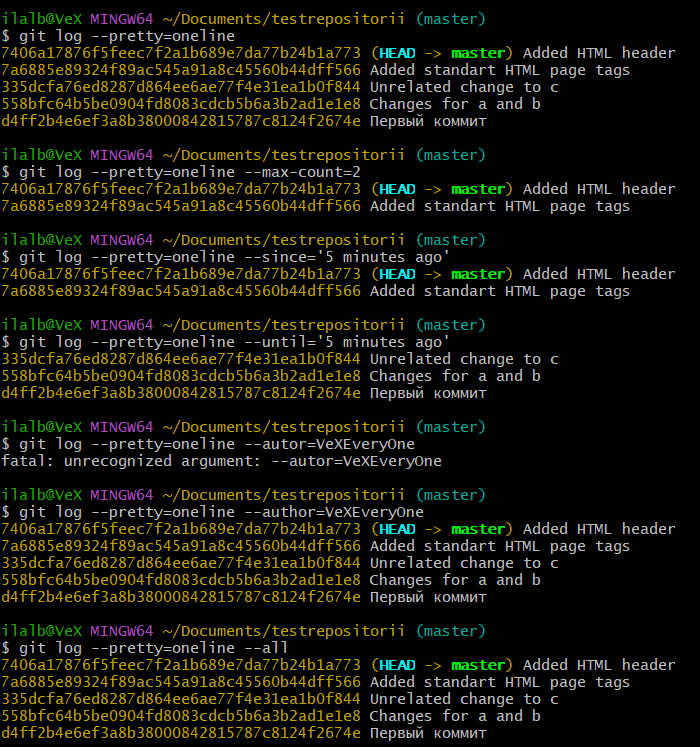
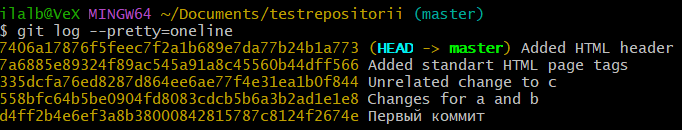
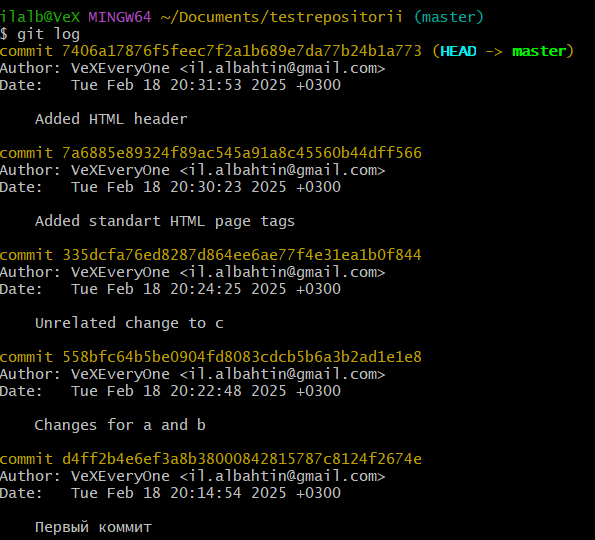
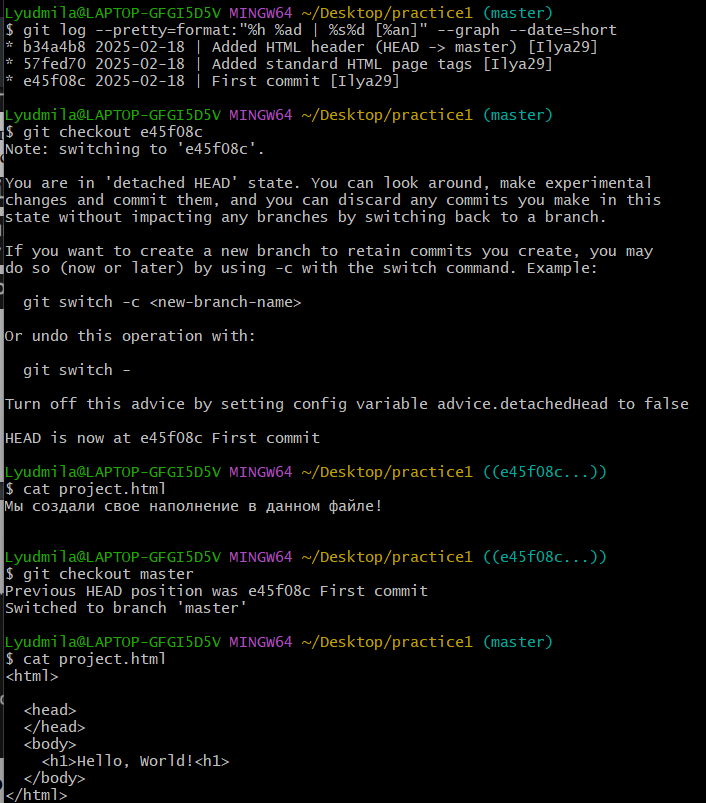
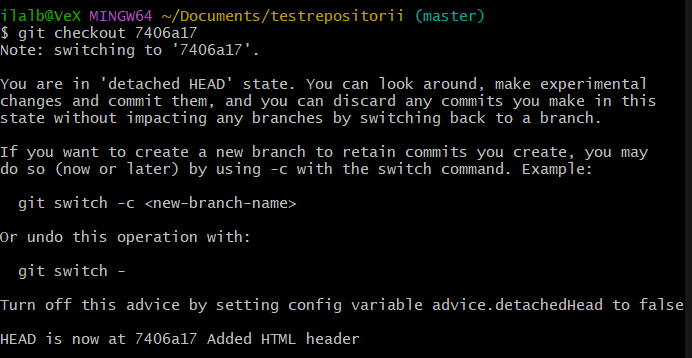


Рисунок 15-18 – Вывод истории коммитов с различными параметрами

## Получение старых версий





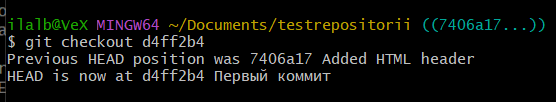
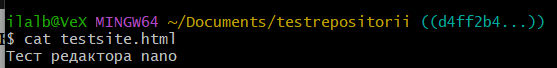
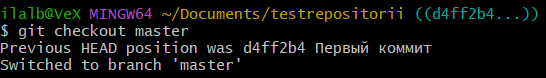
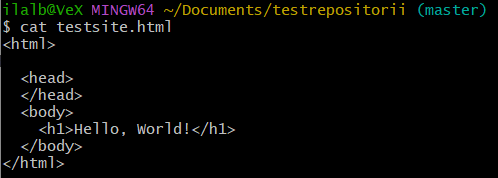
   

Рисунок 19-23 – получение хеша коммита с последующим переключением на него и чтением файла testsite.html. Затем переключение на ветку master и чтение файла из неё.

## Отмена локальных изменений (до индексации)

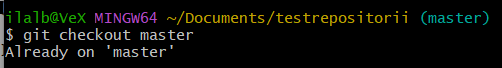
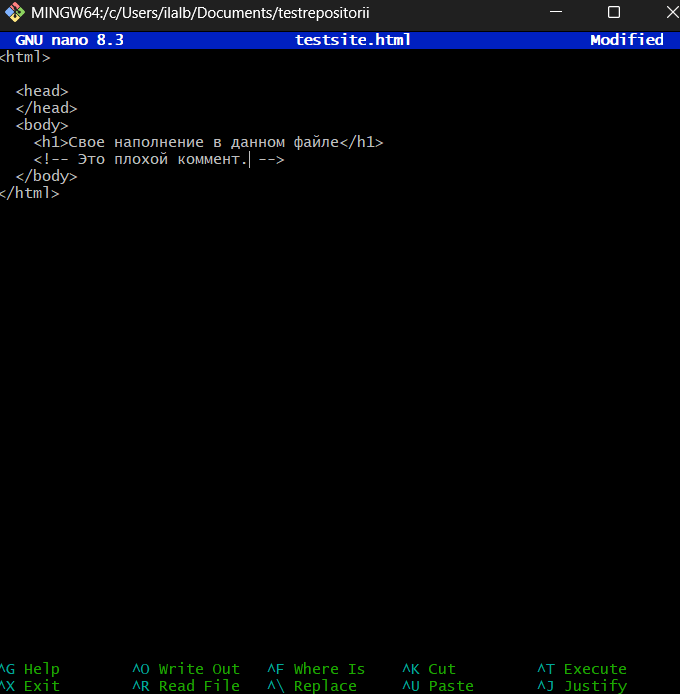
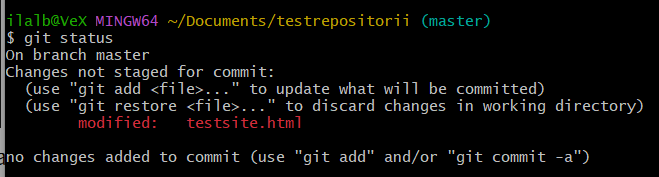
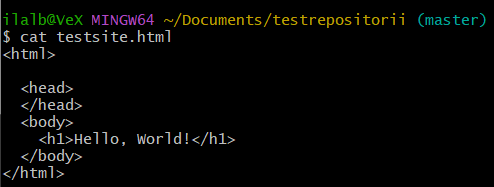
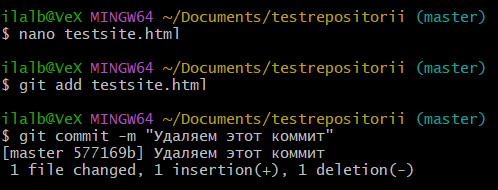
     

Рисунок 24-28 – отмена локальных изменений до индексации

## Отмена локальных изменений (после индексации и до коммита)



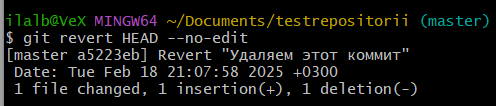


Рисунок 29-30 – Отмена локальных изменений после индексации и до коммита

## Отмена коммита.

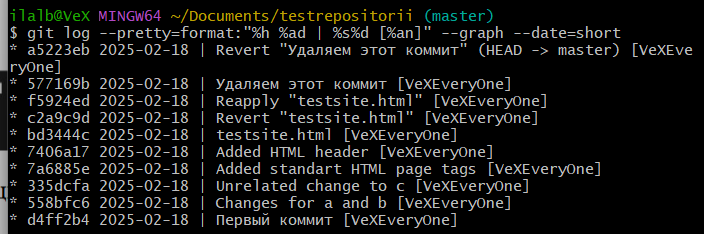


Рисунок 31 – Отмена коммита

# Часть 2. Системы управления репозиториями

# Расчет варианта: V = Ост((3-1)/11)+1 = 3

## Создание аккаунта на GitHub

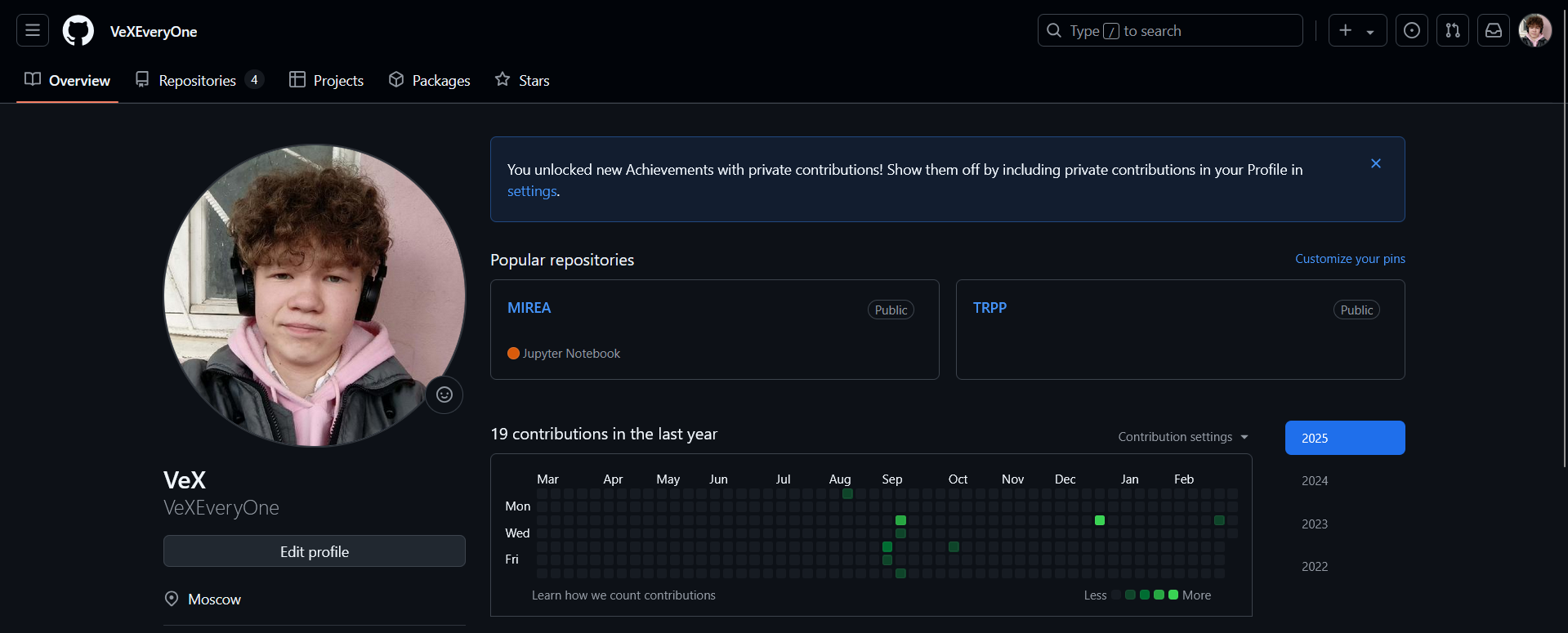


Рисунок 32 – Созданный профиль на GitHub

## Создание SSH-ключа для авторизации



Рисунок 33 – генерация SSH-ключа на рабочей машине

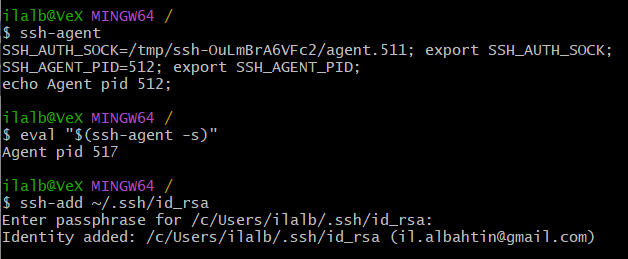


Рисунок 34 – добавление ключа в ssh-agent

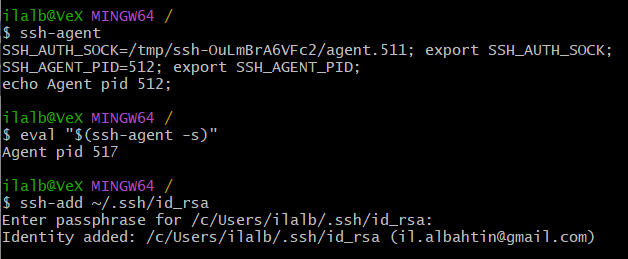


Рисунок 35 – Проверка доступности ключа и добавление ключа в систему

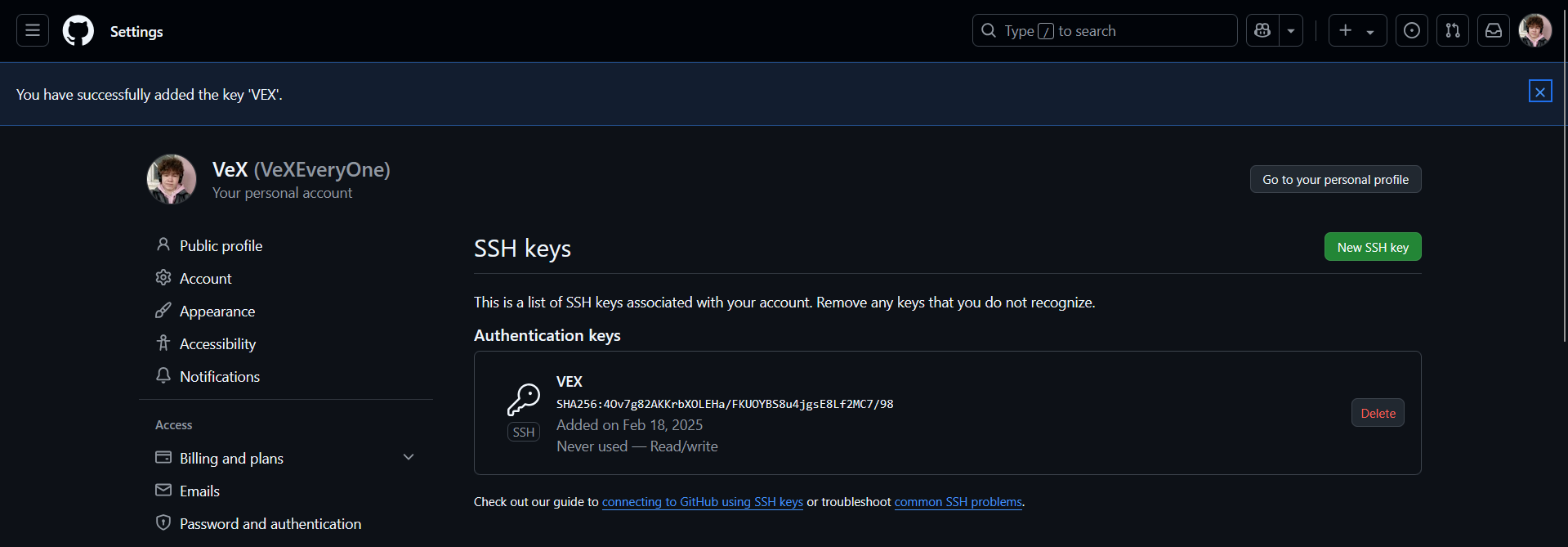


Рисунок 36 – Добавление ключа на Github

## Создание нового репозитория для своего проекта

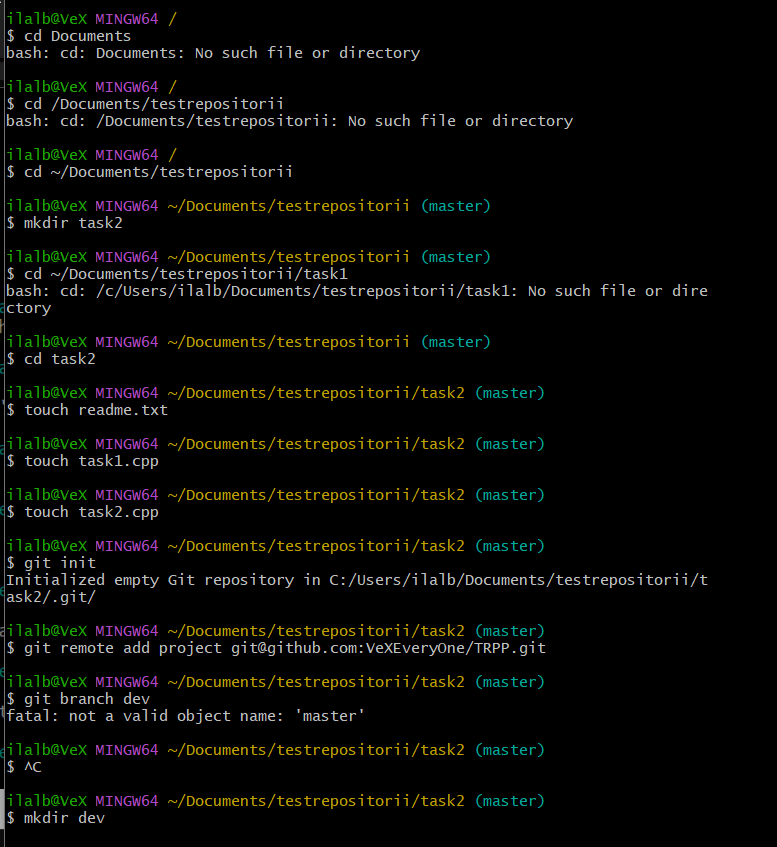


Рисунок 37 – создание репозитория с несколькими файлами

## Связываем локальный и удаленный репозитории

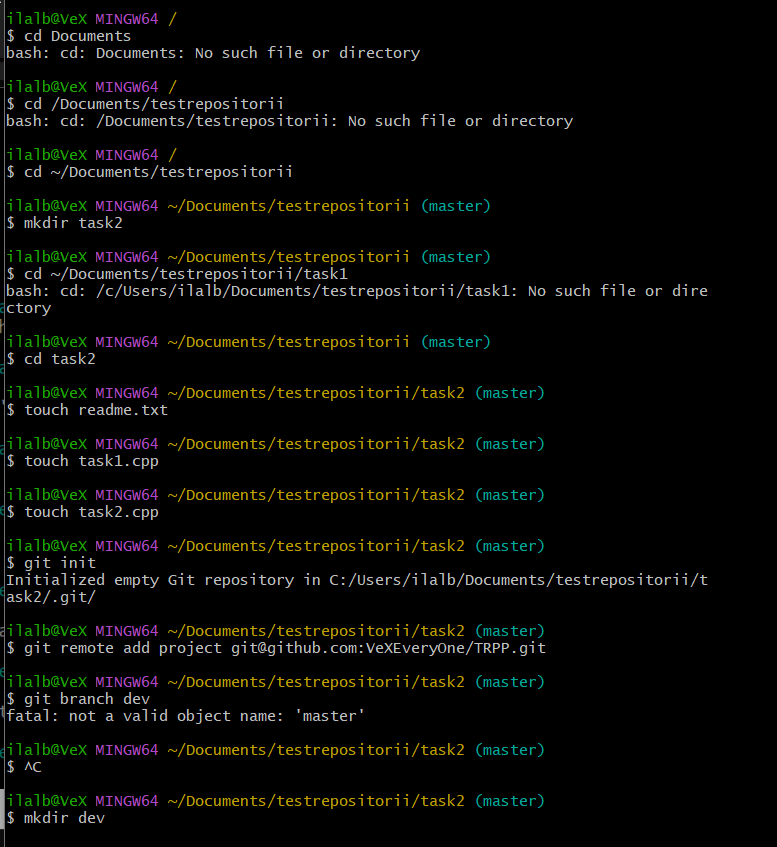


Рисунок 38 – связывание локального репозитория с удалённым

* 1. Создание веток и переключение между ними



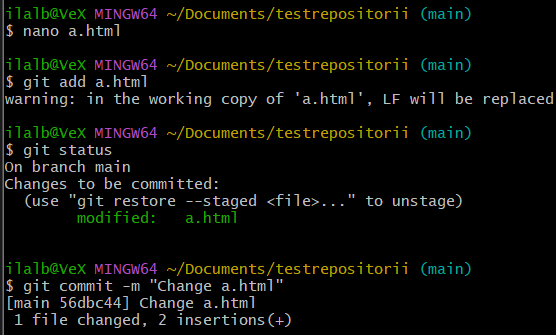


Рисунок 39-40 –Создание новой ветки и работа в ней

## Слияние веток

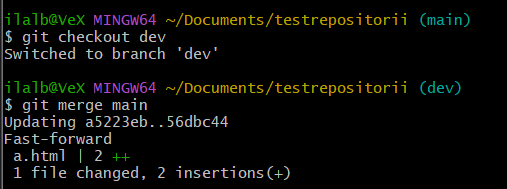


Рисунок 41 – слияние веток main и dev

## Выполните цепочку действий в репозитории, согласно вариантам. Вариант 3.

1. **Клонируйте непустой удаленный репозиторий на локальную машину**

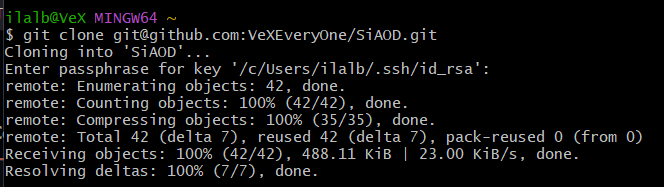


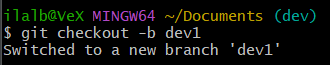
Рисунок 42 – клонирование непустого удаленного репозитория

## Создайте тег указывающий на последний коммит в ветке master

## 

Рисунок 43 – Создание тега на последний коммит ветки master

## Создайте новую ветку и выведите список всех веток



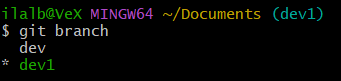


Рисунок 44-45 – создание новой ветки и вывод списка всех веток

## Произведите 3 коммита в новой ветке в разные файлы

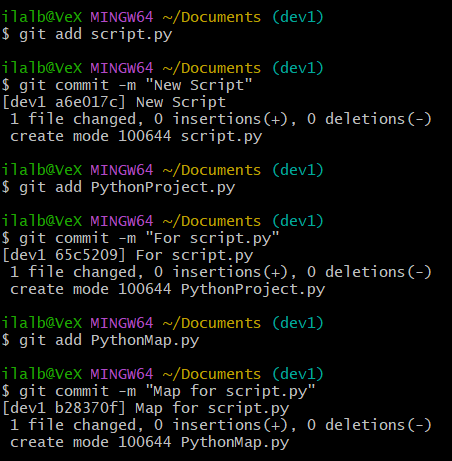


Рисунок 46 – 3 коммита в новой ветке в разные файлы

## Выгрузите изменения в удаленный репозиторий



Рисунок 47 – выгрузка изменений в удалённый репозиторий

## Откатите ветку к созданному тегу (в том числе в удаленном репозитории)

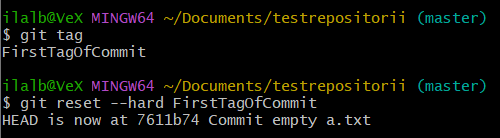


Рисунок 48 – Откат к тегу

## Выведите в консоли различия между веткой master и новой веткой

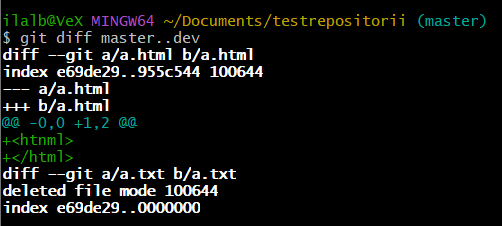


Рисунок 49 – вывод различий между ветками master и dev

# Часть 3. Работа с ветвлением и оформление кода

# Расчет варианта: V = Ост((3-1)/11)+1 = 3

## Форк репозитория



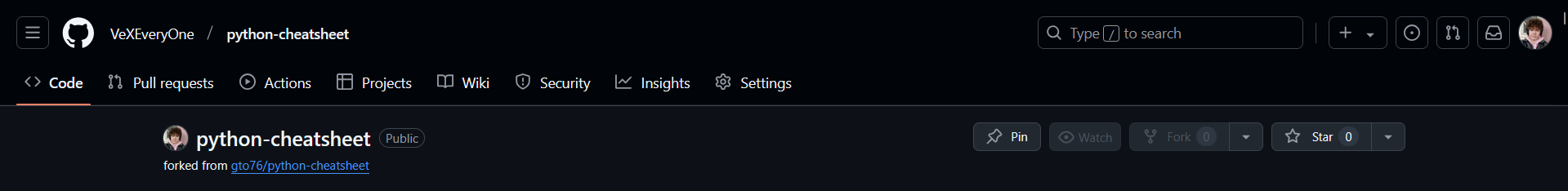
Рисунок 50 – персональный вариант 

Рисунок 51 – сделанный форк репозитория

## Клонирование репозитория на локальную машину

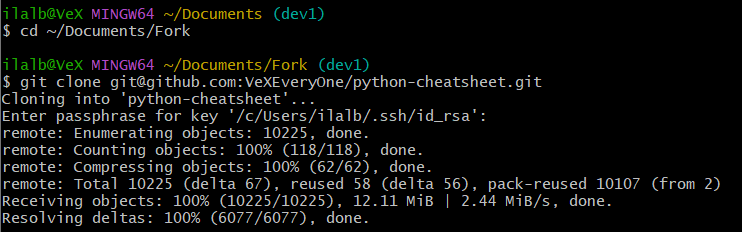


Рисунок 52 – клонирование репозитория на локальную машину

## Создание двух веток в deprecation-note

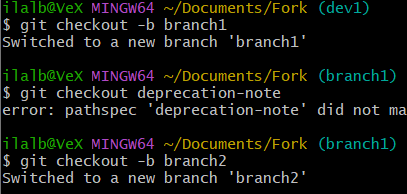


Рисунок 53 – создание двух веток branch1 и branch2

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

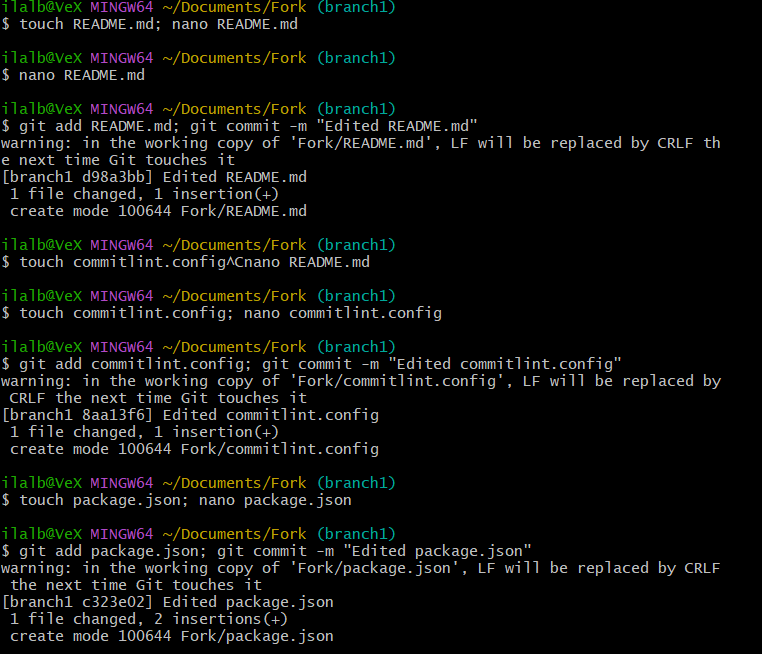
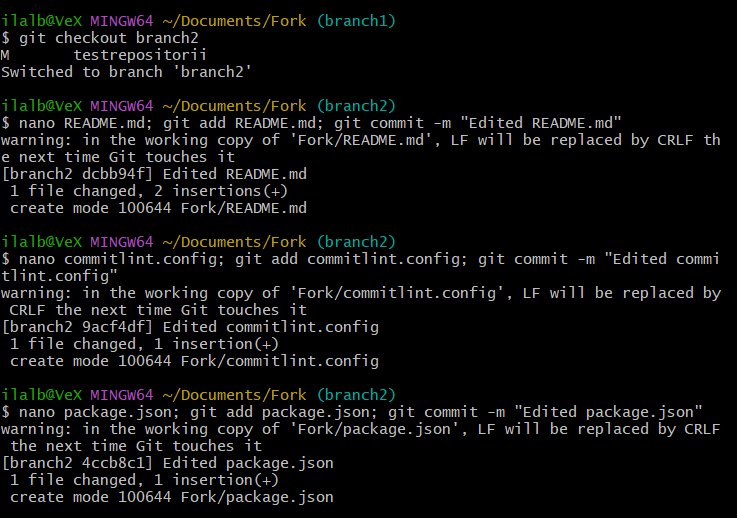
 

Рисунок 54-55 – создание 3-х коммитов в каждую из веток

## Слияние ветки branch1 в ветку branch2 с разрешением конфликтов

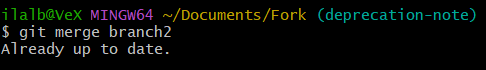
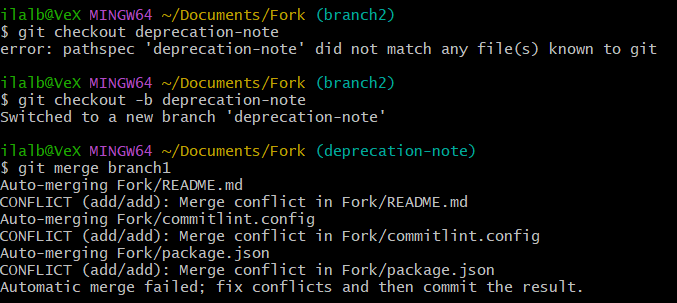
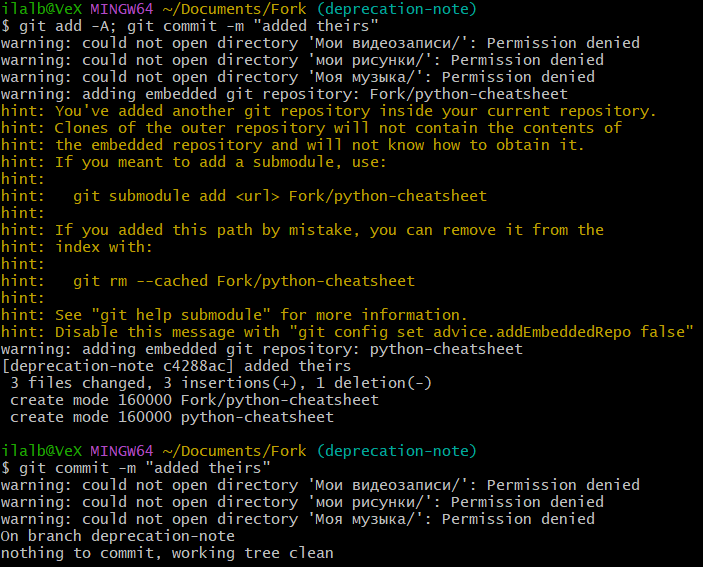
 

Рисунок 56-58 – слияние веток branch1 и branch2 с разрешением конфликтов

## Выгрузка всех изменений во всех ветках в удаленный репозиторий

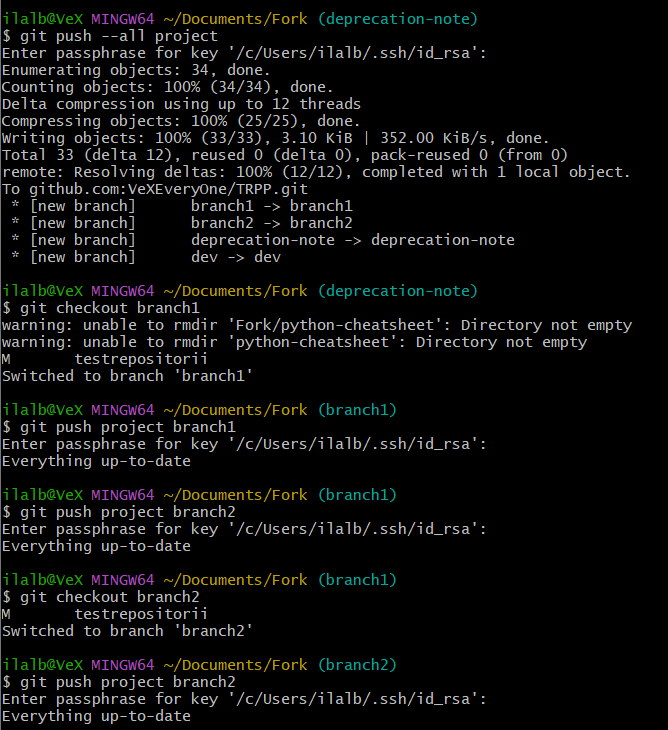


Рисунок 59 – выгрузка изменений во всех ветках в удаленный репозиторий

## Выполнение 3-х коммитов в branch1

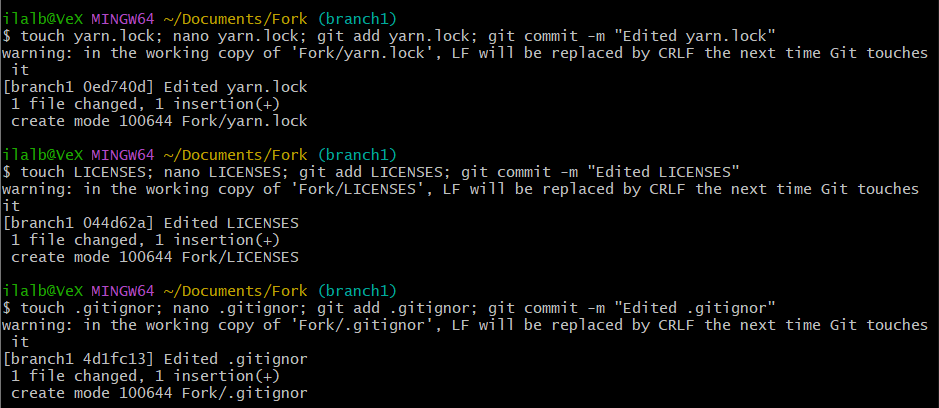


Рисунок 60 – Выполнение 3-х коммитов в branch1

## Повторное клонирование репозитория в другую директорию

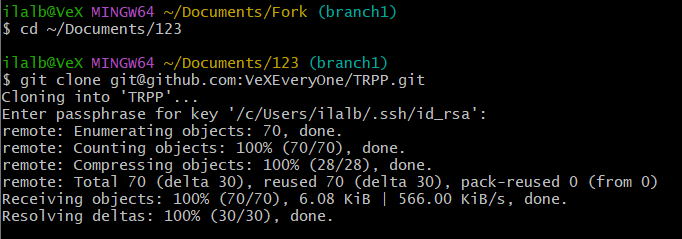


Рисунок 61 – повторное клонирование репозитория в другую директорию

## Выполнение 3-х коммитов в ветку branch1

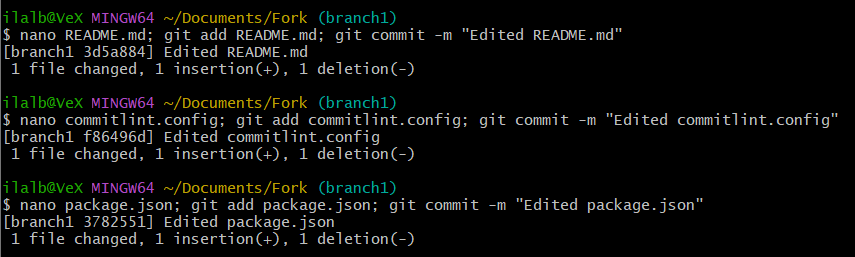


Рисунок 62 – три коммита в ветке branch1

## Выгрузка всех изменений в удаленный репозиторий

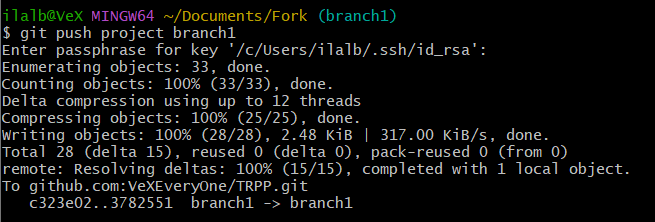


Рисунок 63 – выгрузка всех изменений из локального репозитория в удаленный

## Выгрузка изменения из старого репозитория с опцией –force

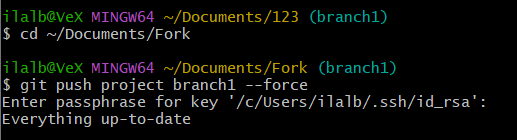


Рисунок 64 – вернулись в старый клон с репозиторием и выгрузили изменения с опцией –force

## Получение всех изменения в новом репозитории

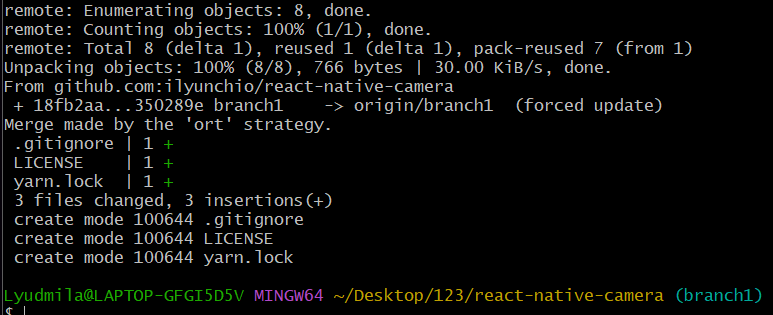


Рисунок 65 – перешли в новый репозиторий и получили в нём все изменения

# Ответы на 8 любых вопросов:

## Как отменить слияние веток, если произошел конфликт?

Отменить слияние веток при появлении конфликта можно командой **git merge --abort**.

## Для чего нужен .gitignore?

**.gitignore** используется для указания файлов и директорий, которые не должны учитываться системой контроля версий Git. Это позволяет игнорировать временные файлы, конфигурационные файлы, и прочее, чтобы они не попадали в репозиторий.

## Что делает команда git status?

Команда **git status** показывает текущий статус репозитория, включая измененные, добавленные и неотслеживаемые файлы.

## Что делает команда git add?

Команда **git add** добавляет изменения файлов в индекс, подготавливая их к следующему коммиту.

1. **Что делает команда git log?**

Команда **git log** показывает историю коммитов. Она перечисляет

коммиты, которые можно достичь, следуя связям родительских коммитов от указанных коммитов.

1. **Что делает команда git diff?**

Команда **git diff** показывает различия между рабочим деревом и индексом, а также между индексом и последним коммитом.

## Что делает команда git show?

Команда **git show** отображает информацию о конкретном коммите, включая изменения, внесенные этим коммитом.

## Что делает команда git stash?

Команда **git stash** временно сохраняет изменения в рабочей директории, чтобы можно было переключиться на другую ветку или выполнить другие операции.

# Вывод

В ходе работы были изучены основные команды Git и принципы работы с GitHub. Выполнены задачи по настройке Git, созданию и управлению локальным и удаленным репозиториями, работе с ветками, слиянию и разрешению конфликтов. Также рассмотрены команды для отмены изменений, просмотра истории коммитов и управления изменениями. Полученные навыки позволяют эффективно управлять версиями проектов и работать в команде.