|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  |  |  | | МИНОБРНАУКИ РОССИИ | | | | Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«МИРЭА – Российский технологический университет»**  **РТУ МИРЭА** | | |   Институт Информационных технологий | |  |
|  | |  |
| Кафедра Математического обеспечения и стандартизации информационных технологий | |  |
|  |  | |
|  |  | |

|  |  |
| --- | --- |
| **ОТЧЕТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ № 1** | |
| **по дисциплине** | |
| **«**Технология разработки программных приложений**»**  **Тема: «Система контроля версий.»** | |
|  | |
| Выполнил студент группы ИКБО-33-22 | Шило Ю.С. |
| Принял преподаватель | Петрова А.А. |

|  |  |  |
| --- | --- | --- |
| Лабораторная работа выполнена | «\_\_»\_\_\_\_\_\_\_202\_\_ г. | *(подпись студента)* |
|  |  |  |
| «Зачтено» | «\_\_»\_\_\_\_\_\_\_202\_\_ г. | *(подпись руководителя)* |

Москва 2024

**ЗАДАНИЕ**

**Цель работы:** получить навыки по работе с командной строкой и git’ом.

**Часть 1. Основные команды Git.**

*Задание 1.* Установите и настройте клиент git на своей рабочей станции.

После установки git на рабочее устройство нам потребуется его настроить для этого добавим имя и почту для добавления имени используем команду: git config --global user.name “Shilo Yuriy”, а для изменения почты git config --global user.email “shilo.y.s@edu.mirea.ru”, а для проверки наших изменений введем git config --list. Выполнение данных команд предоставлено на рисунке 1.1.

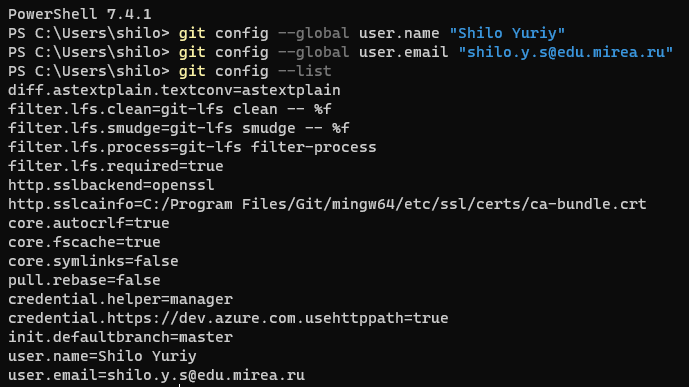


Рисунок 1.1 – Изменение глобальных настроек информации о пользователе

*Задание 2.* Создайте локальный репозиторий и добавьте в него несколько файлов

Для создание локальной репозитории мы должны ввести команду git init, находясь в папке, которую мы хотим сделать нашим репозиторием. После ввода данной команды консоль вывела следующее рисунок 1.2.



Рисунок 1.2 – Создание локального репозитория

*Задание 3.* Внесите изменения в один из файлов.

В пустой текстовый файл была написана фраза “Какой-то текст, который был добавлен. После создания репозитория.” рисунок 1.3.

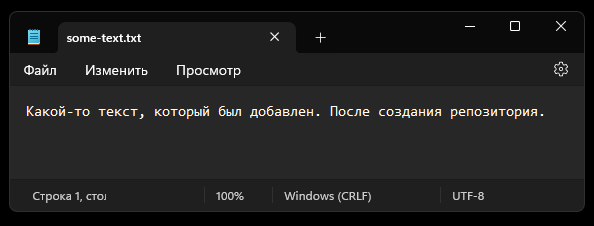


Рисунок 1.3 – Содержание текстового файла

*Задание 4.* Проиндексируйте изменения и проверьте состояние.

Для того что бы добавить изменный индекс в Git мы должны воспользоваться командой git add ., а чтобы проверить состояние репозитории мы должны использовать команду git status. После выполнения данных команд консоль выведет следующие рисунок 1.4.

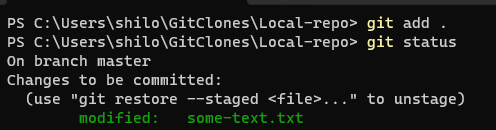


Рисунок 1.4 – Добавление индекса в Git и получение статуса локального репозитория

*Задание 5.* Сделайте коммит того, что было проиндексировано в репозиторий. Добавьте к коммиту комментарий.

Для того чтобы за коммитить изменения мы воспользуемся командой git commit -m “…” (за место трех точек мы пишем наш комментарий) выполнение данной команды показано на рисунке 1.5.

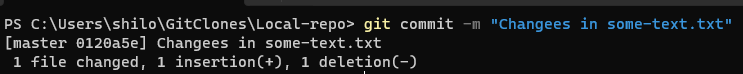


Рисунок 1.5 – Выполнение команды для коммита с комментарием

*Задание 6.* Измените еще один файл. Добавьте это изменение в индекс git. Измените файл еще раз. Проверьте состояние и произведите коммит проиндексированного изменения. Теперь добавьте второе изменение в индекс, а затем проверьте состояние с помощью команды git status. Сделайте коммит второго изменения.

Для выполнения данного задания поочередно введем команды, которые мы разбирали в предыдущих заданиях. Все введённые команды и их результат предоставлены на рисунке 1.6.

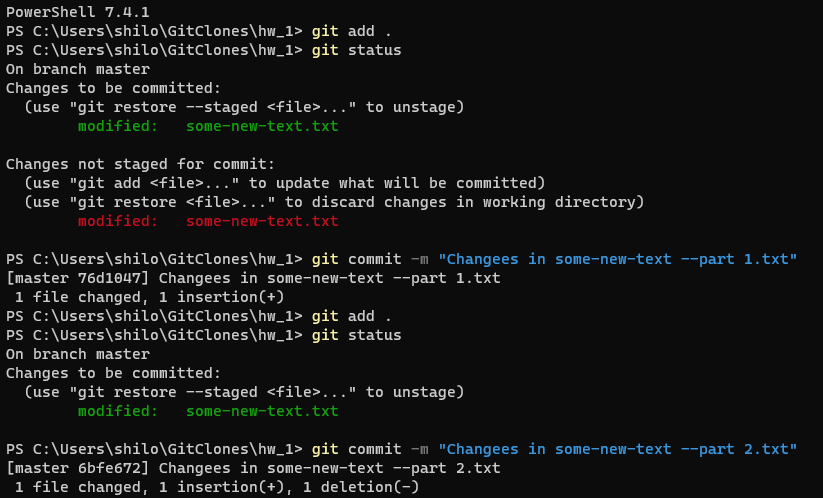


Рисунок 1.6 – Все операции произведенные в задание 6

*Задание 7.* Просмотрите историю коммитов с помощью команды git log. Ознакомьтесь с параметрами команды и используйте некоторые из них для различного формата отображения истории коммитов.

На рисунке 1.7 предоставлены три различных формата вывода коммитов: git log, git log --inline, git log --graph. Первый это вывод без форматирования, второй выводит каждый коммит в виде одной строчки и третий выводит список в виде графов.

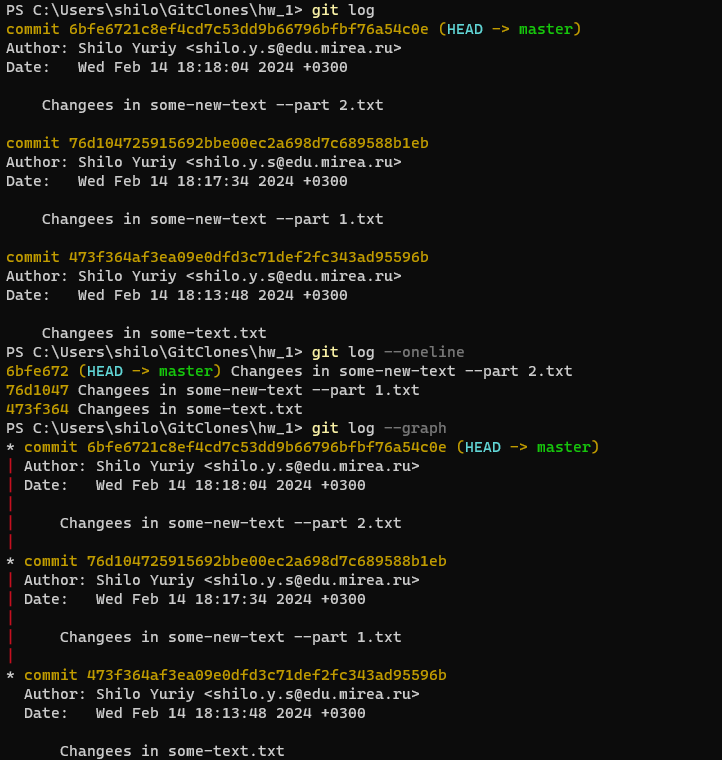


Рисунок 1.7 – Три варианта форматирования вывода коммитов

*Задание 8.* Верните рабочий каталог к одному из предыдущих состояний.

Для возврата к предыдущему коммиту мы воспользуемся командой git checkout HEAD~, также за место HEAD~ мы можем написать хэш код коммита, имя ветки или имя ветки. Показ работы команды по возврату на один коммит назад предоставлена на рисунке 1.8.

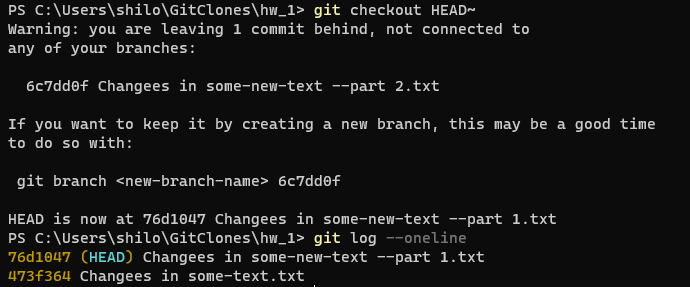


Рисунок 1.8 – Действия по возврату к предыдущему коммиту

*Задание 9.* Изучите, как создавать теги для коммитов для использования в будущем.

Для создания простого тега используется команда git tag …, за место троеточия нужно написать имя нашего тега. А для просмотра списка существующих тегов используется команда git tag. После добавления тега он будет отображаться в истории коммитов для удобного возврата. Отработка данных команд предоставлена на рисунке 1.9.

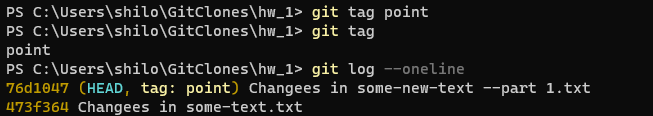


Рисунок 9 – Создание тега point

*Задание 10.* Отмените некоторые изменения в рабочем каталоге (до и после индексирования).

Для отката изменения в рабочей папки можно воспользоваться командой git checkout point, после выполнения данной команды я возвращусь на момент с тегом point, а для возврата к другой точке я воспользовался командой git checkout tag\_02. Выполнение данных команд предоставлено на рисунке 1.10.

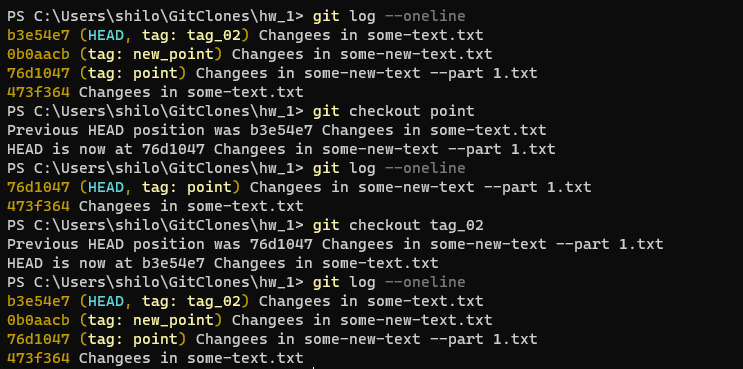


Рисунок 1.10 – Отмена изменений в рабочем каталоге

*Задание 11.* Отмените один из коммитов в локальном репозитории.

Для выполнения данного задания существует два способа отмены коммита первый способ использовать git revert … или git reset … где за место троеточия мы пишем идентификатор коммита. Первая команда создает новый коммит, который отменяет изменения, но при этом сохраняет историю, второй же удаляет историю. Действие одной из команд предоставлено на рисунке 1.11.

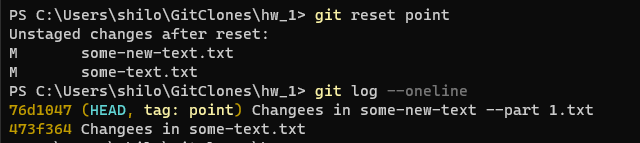


Рисунок 1.11 – Отмена коммита с удаление истории

**Часть 2. Системы управления репозиториями**

*Задание 1.* Создайте аккаунт на GitHub (у кого нет)

Для решения данного задания я вошел в свой профиль GitHub его скриншот преоставлен на рисунке 2.1.

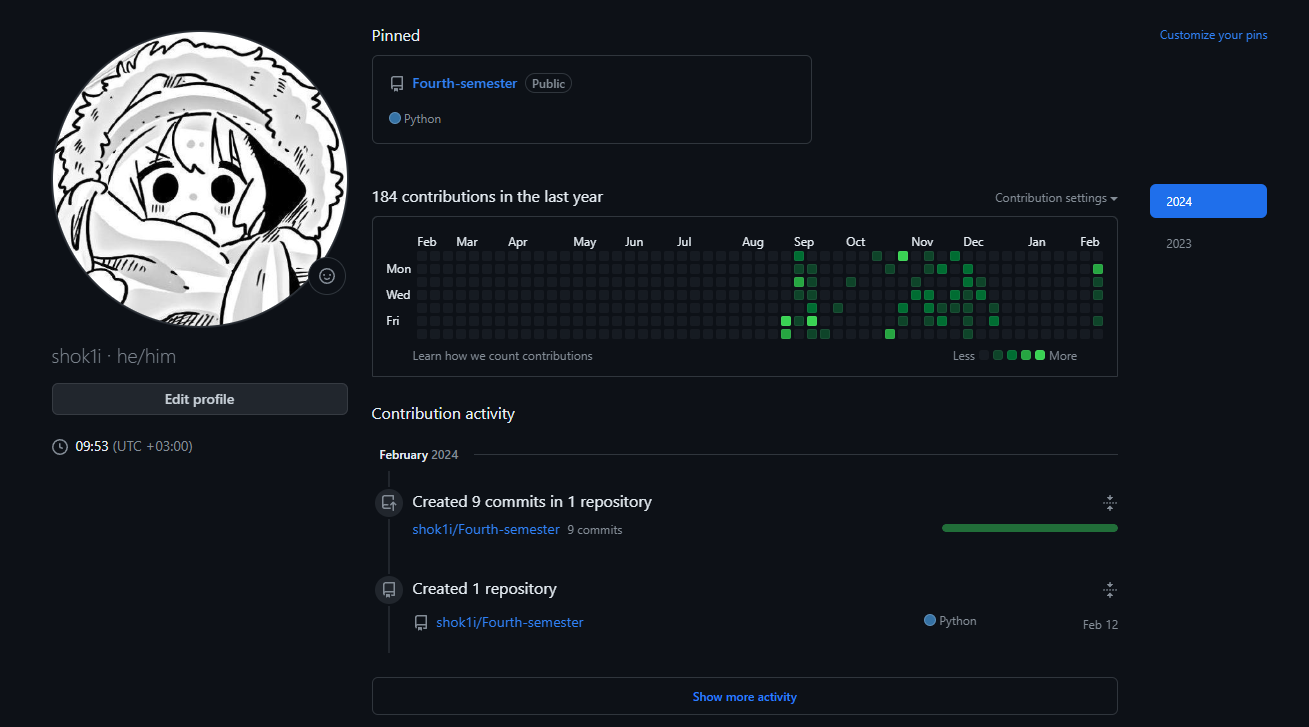


Рисунок 2.1 – Скриншот профиля на GitHub

*Задание 2.* Создайте репозиторий на GitHub и на локальной машине, согласно выбранной теме проекта

На рисунке 2.2 показан созданный репозиторий на GitHub.

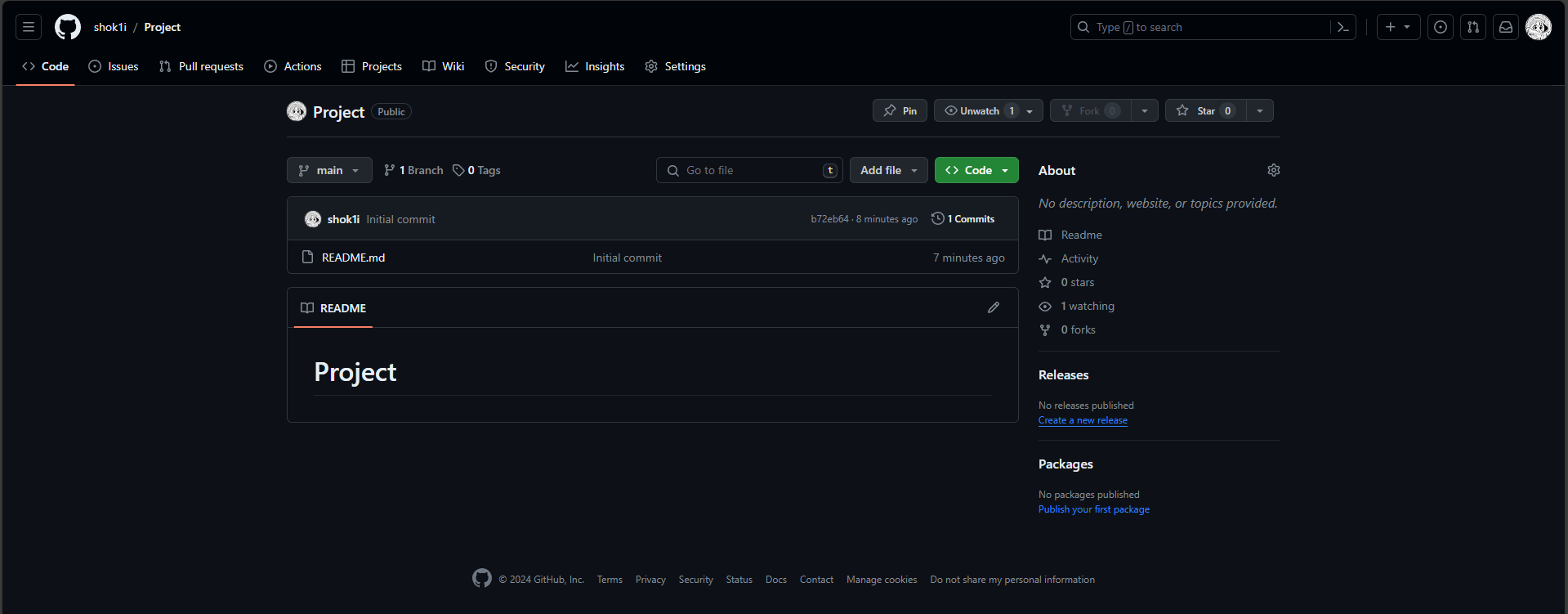


Рисунок 2.2 – Создан репозиторий на гитхабе

*Задание 3.* Создайте несколько файлов на локальной машине при помощи консоли

На локальной машине были созданы три текстовых файла done--project git--push и path скриншот репозитория показан на рисунке 2.3.

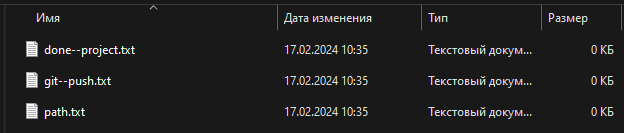


Рисунок 2.3 – Созданные файлы в папке проекта

*Задание 4.* Создайте SSH-ключ для авторизации

Для создания SSH-ключа введем команды ssh-keygen -t rsa -b 4096 -C “shilo152004@mail.ru” после ввода данный команды был следущий вывод в консоль рисунок 2.4.

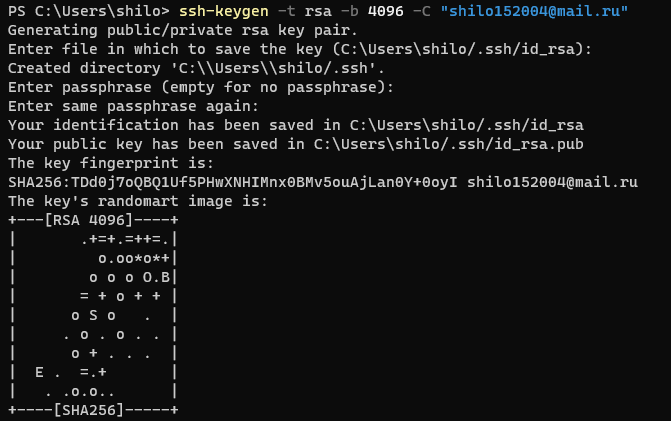


Рисунок 2.4 – Создание SSH ключа

*Задание 5.* Свяжите репозиторий локальной машины с репозиторием на GitHub при помощи консоли

Для связывания локального репозитория с репозиторием GitHub в находясь в папке репозитория в консоль введем команду git remote add project git@github.com:shok1i/Project .git. Отработка данной команды предоставлена на рисунке 2.5.



Рисунок 2.5 – Связывание репозиториев

*Задание 6.* Создайте новую ветку в репозитории с помощью команды, произведите в ней какие-нибудь изменения, а после слейте с веткой master

Для перемещения в другую ветку используем команду git checkout … (за место троеточия название ветки). Для создания новой ветки используется команда git branch … (за место троеточия название ветки).

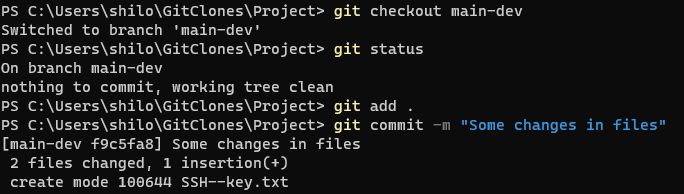


Рисунок 2.6 – Создание новой ветки и коммит в нее изменений

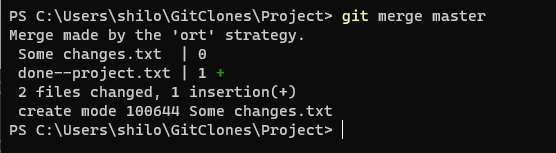


Рисунок 2.7 – Слияние ветки с веткой master

*Задание 7.*

**ВАРИАНТ 5**

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

Для клонирования репозитория мы должны использовать команду git clone git@github.com:… (за место троеточия пишем название репозитория). После выполнени данной команды команды в консоль выведиться следующие рисунок 2.8.

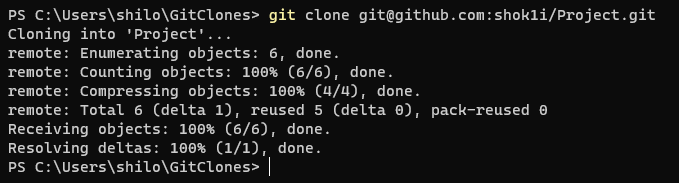


Рисунок 2.8 – Выполнение команды

*Задание 2.* Создайте новую ветку и выведите список всех веток

Для создания ветки используем git branch … (за место троеточия нужно написать ее имя). Для того что бы узнать какие ветки присутствуют в репозитории воспользуемся командой git branch. На рисунке 2.9 показано выполнение данных команд.

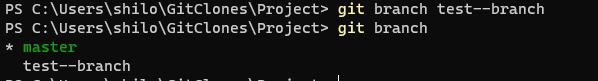


Рисунок 2.9 – Создание ветки и проверка на ее наличие

*Задание 3.* Произведите 3 коммита в новой ветке в разные файлы

Для того чтобы переместиться в другую ветку нужно использовать команду git checkout. На рисунке 2.10 показано создание трех коммитов в новую ветку.

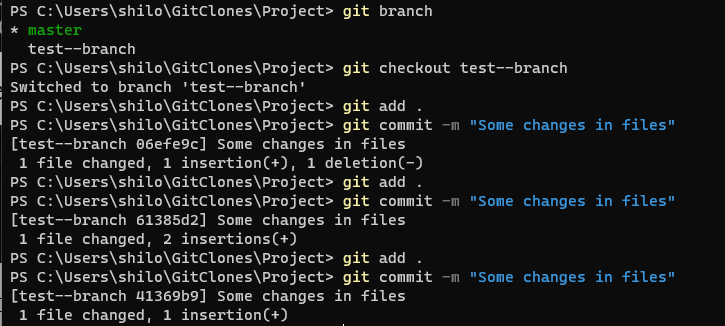


Рисунок 2.10 – Создание трех коммитов на новой ветке

*Задание 4.* Выгрузите изменения в удаленный репозиторий

Для того чтобы выгрузить изменения в удаленный репозиторий нужно воспользоваться команду git push. На рисунке 2.11 показан процесс выгрузки репозитория в удаленный репозиторий.

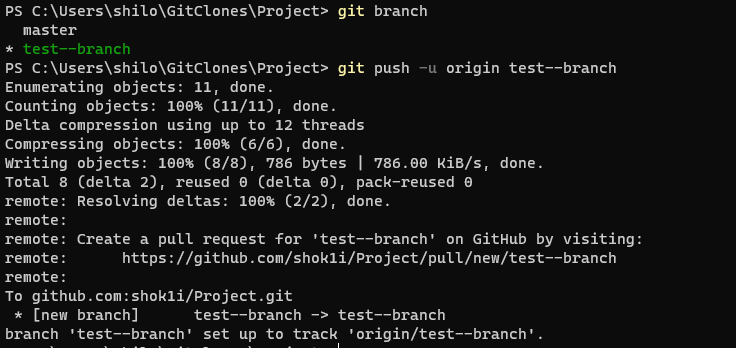


Рисунок 2.11 – Выгрузка репозитория на GitHub

*Задание 5.* Произведите revert предпоследнего коммита в новой ветке

Для выполнения данного задания нужно воспользоваться команндой git revert HEAD^. Выполнение данной команды показано на рисунке 2.12.

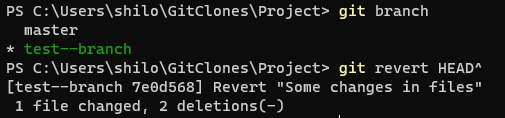


Рисунок 2.12 – Откат изменений

*Задание 6.* Выведите в консоли различия между веткой master и новой веткой

Для вывода в консоль различия между ветками нужно воспользоваться командой git diff (и название двух веток). Выполнение данной команды предоставлено на рисунке 2.13.

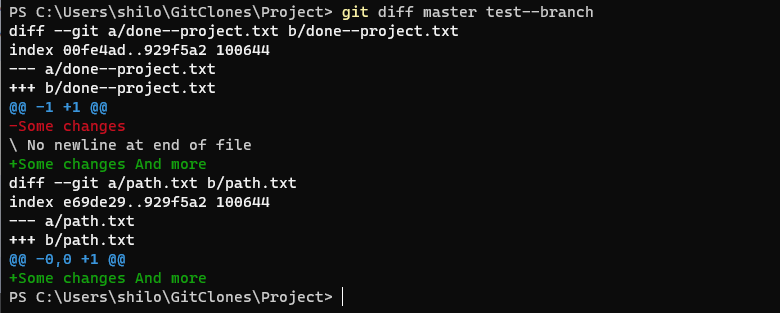


Рисунок 2.13 – Различия между веткой maser и test--branch

*Задание 7.* Слейте новую ветку с master при помощи merge

Для слияния двух веток используется команда git merge (и название двух веток или название ветки, с которой вы хотите слить текущую). Выполнение данной команды показано на рисунке 2.14.

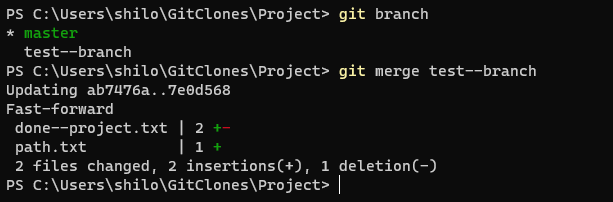


Рисунок 2.14 – Слияние текущей ветки с веткой test--branch

**Часть 3. Ветвление и оформление кода**

**ВАРИАНТ 2.** *github.com/google/model\_search*

*Задание 1.* Сделайте форк репозитория в соответствии с вашим вариантом

Для создания форка на странице нужного нам репозитория мы должны нажать создать форк и выбрать нужное нам название. Созданный форк предоставлен на рисунке 3.1.

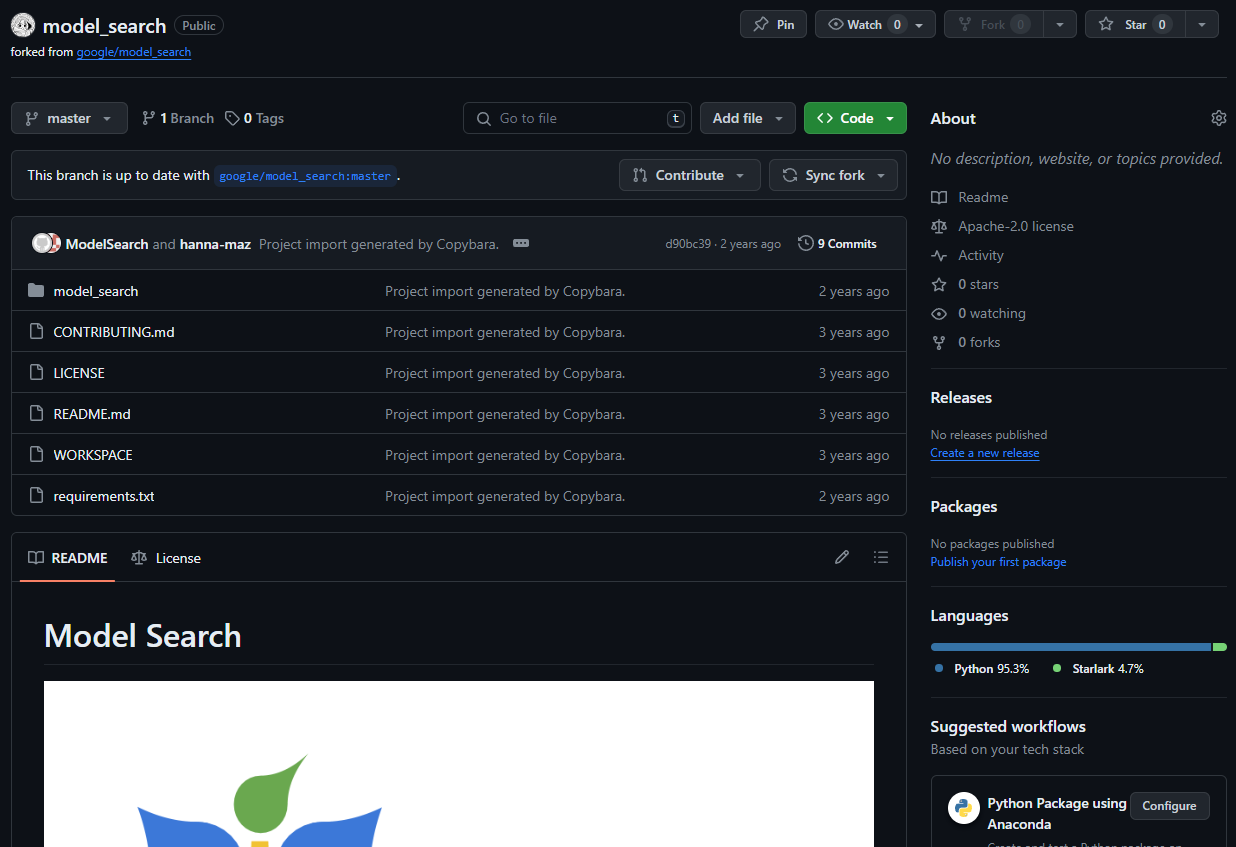


Рисунок 3.1 – Форк репозитория Model-Search

*Задание 2.* Склонируйте его на локальную машину

Для клонирования репозитория мы должны использовать команду git clone git@github.com:… (за место троеточия пишем название репозитория). После выполнени данной команды команды в консоль выведиться следующие рисунок 3.2.



Рисунок 3.2 – Клонирование нужного нам репозитория

*Задание 3.* Создайте две ветки branch1 и branch2 от последнего коммита в master'е

Для создания новой ветки используется команда git branch … (за место троеточия название ветки). Создание веток и выполнение данного задания предоставлена на рисунке 3.3.

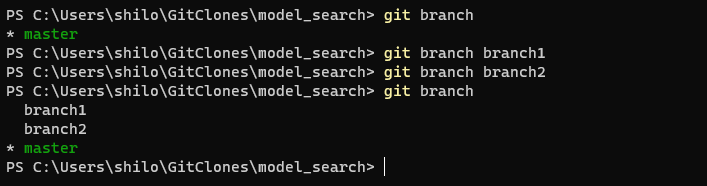


Рисунок 3.3 – Создание Двух веток branch1 и branch2

*Задание 4.* Проведите по 3 коммита в каждую из веток, которые меняют один и тот же кусочек файла

Создадим две ветки командой git branch branch1 и branch2 соответственно, затем перейдя в ветки banch1 командой git checkout branch1. Три раза выполним изменения файла каждый раз коммитя это изменение, тоже самое проделаем и в другой ветке. Весь процесс выполнения команд предоставлен на рисунке 3.4.

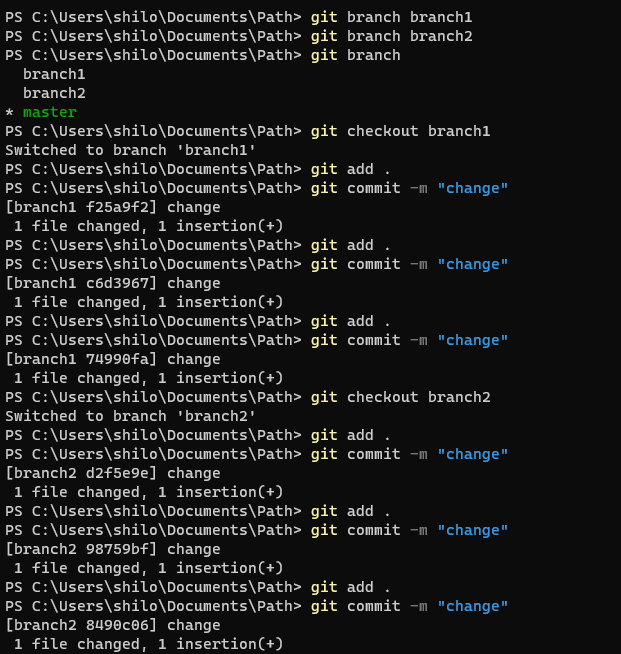


Рисунок 3.4 – Действия выполняемые при решении задания

*Задание 5.* Выполните слияние ветки branch1 в ветку branch2, разрешив конфликты при этом

Для слияния текущей ветки с другой воспользуемся git merge branch1 как мы можем заметить у нас возник конфликт слияния. В моем случае я разрешил его вручную. Затем проиндексировал данный файл и закомител слияние рисунок 3.5.



Рисунок 3.5 – Слияние двух веток с разрешением конфликта

*Задание 6.* Выгрузите все изменения во всех ветках в удаленный репозиторий

Для выгрузки изменений воспользуемся командой git push -u origin название ветки. Выполнение данных команд предоставлено на рисунке 3.6.

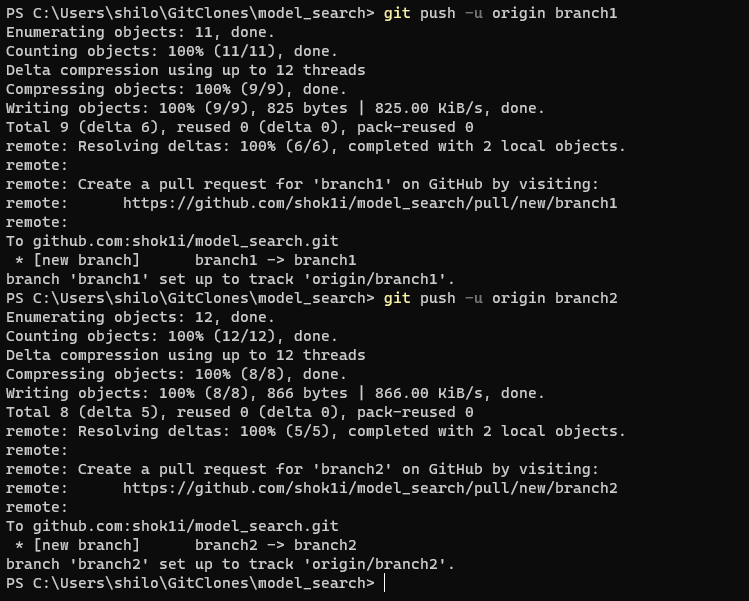


Рисунок 3.6 – Выгрузка двух веток в удаленный репозиторий

*Задание 7.* Проведите еще 3 коммита в ветку branch1

Выберем ветку командой git checkout branch1 и сделаем три коммита в данной ветке. Выполнение данных команд предоставлено на рисунке 3.7.

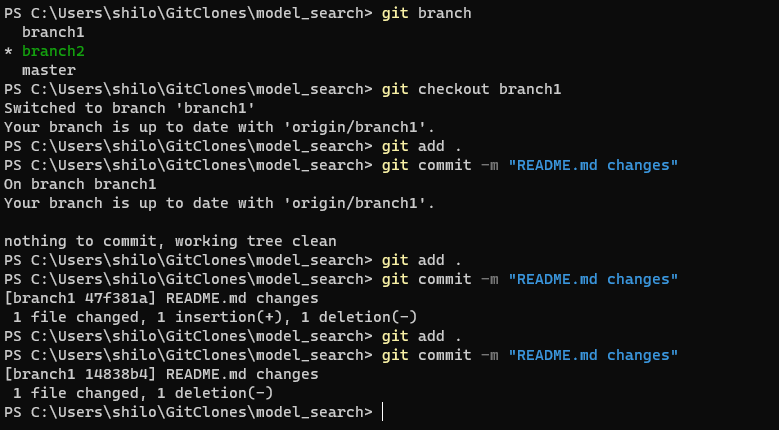


Рисунок 3.7 – Добавление трех коммитов в ветку branch1

*Задание 8.* Склонируйте репозиторий еще раз в другую директорию

Для клонирования удаленного репозитория воспользуемся командой git clone ссылка на репозиторий. На рисунке 3.8 показано выполнение данной команды

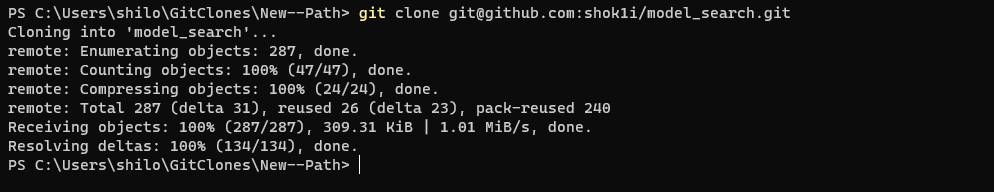


Рисунок 3.9 – Клонирование удалённого репозитория на локальное устройство

*Задание 9.* В новом клоне репозитории сделайте 3 коммита в ветку branch

Выберем ветку командой git checkout branch1 и сделаем три коммита в данной ветке. Выполнение данных команд предоставлено на рисунке 3.9.

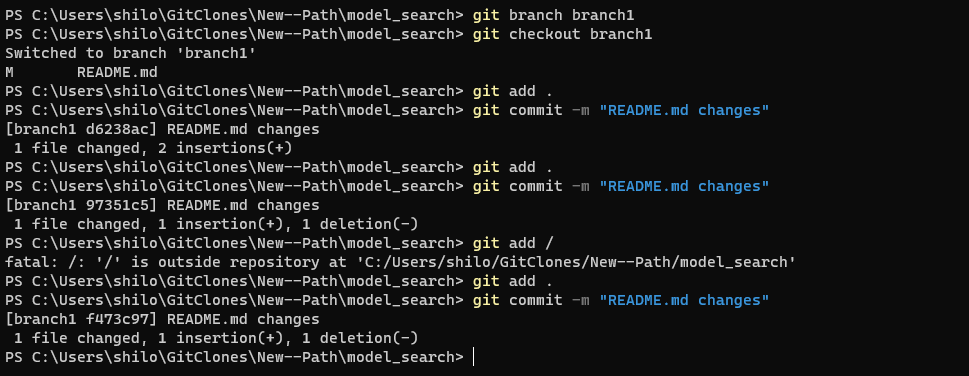


Рисунок 3.9 – Создание трех коммитов в новом репозитории в ветке branch1

*Задание 10.* Выгрузите все изменения из нового репозитория в удаленный репозиторий.

Для выгрузки изменений используется команда git push. Выполнение данной команды предоставлена на рисунке 3.10.

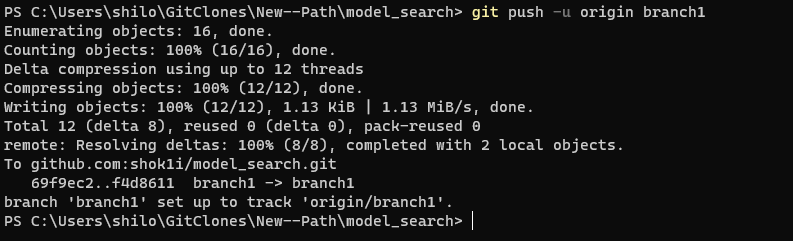


Рисунок 3.10 – Выгрузка изменений на удаленный репозиторий

*Задание 11.* Вернитесь в старый клон с репозиторием, выгрузите изменения с опцией –force

Для выполнения данного задания нужно воспользоваться командой git push --forse. Выполнение данной команды предоставлена на рисунке 3.11.

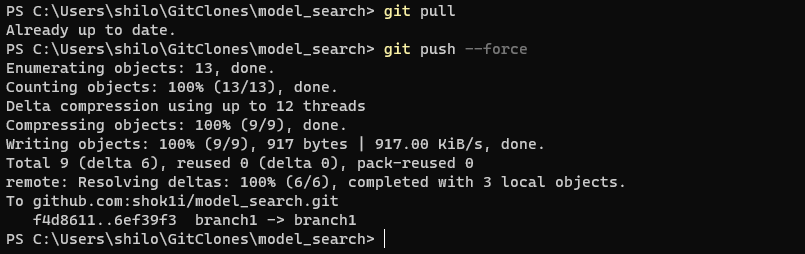


Рисунок 3.11 – Выполнение команды git push –forse

*Задание 12.* Получите все изменения в новом репозитории

Для получения изменений в локальном репозитории нужно воспользоваться командой git pull. Ее выполнение предоставлено на рисунке 3.12

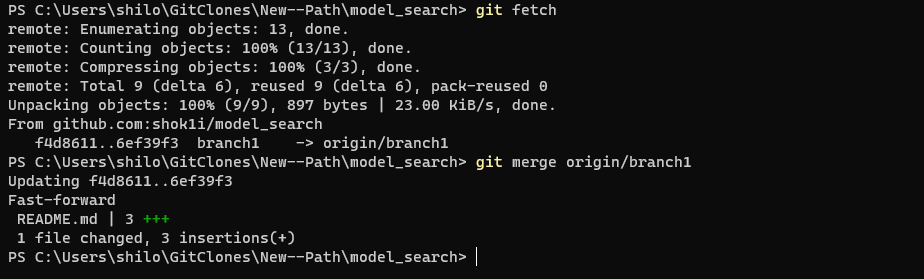


Рисунок 3.12 – Получение изменений с удаленого репозитория

**Контрольные вопросы**

1. Что делает команда git status?

Команда `git status` показывает текущее состояние рабочего каталога и индекса в репозитории Git, показывая измененные, добавленные и удаленные файлы.

1. Что делает команда git add?

Команда `git add` добавляет изменения в рабочем каталоге к индексу, подготавливая их к коммиту.

1. Что делает команда git log?

Команда `git log` отображает историю коммитов в репозитории Git, показывая информацию о каждом коммите, такую как хэш коммита, автор, дата и сообщение коммита.

1. Что делает команда git diff?

Команда `git diff` показывает различия между рабочим каталогом и индексом, между индексом и последним коммитом, или между любыми двумя коммитами в репозитории Git.

1. Что делает команда git show?

Команда `git show` отображает информацию о конкретном коммите или объекте Git, показывая изменения, внесенные в рабочий каталог в результате этого коммита или объекта.

1. Как сделать ветку с названием my\_branch?

Для создания новой ветки с названием `my\_branch` в Git, вы можете использовать команду `git branch my\_branch`. Эта команда создаст новую ветку с заданным названием и переключит вас на нее.

1. Что такое форк репозитория?

Форк репозитория — это создание копии чужого репозитория на GitHub в своем аккаунте. Форк позволяет внести изменения в проект без влияния на исходный репозиторий.

1. Что такое pull request

Pull request - это механизм на GitHub, который позволяет разработчикам предложить свои изменения из форка репозитория в исходный репозиторий.

**Вывод**

В ходе данной практической работы были получены навыки работы с GitHub и с git. Был создан репозиторий для своего проекта и форк чужого. Создан аккаунт на GitHub.