|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Студент:** | Еселидзе Д.В. | | **Допуск:** | Муравьев К.А. |
| **Отчет по лабораторной работе №3**  «Выпуск TLS-сертификатов и работа с git-репозиториями» | | | | |
| Дата выполнения | 2.12.2022 | Дата защиты | | 19.12.2022 |
| Оценка |  | Подпись | |  |

Цель работы:

|  |
| --- |
| Получить навыки по выпуску сертификатов, работе с git-репозиториями |

Задачи работы:

|  |
| --- |
| Подключить безопасный протокол https, научиться работать с системами контроля версий, создать локальные и открытые репозитории. |

Краткий конспект теоретической части:

|  |
| --- |
| HTTPS и TLS: (Transport Layer Security - безопасность транспортного уровня) - Криптографический протокол для более безопасной передачи информации, созданный в 1999м году на основе протокола SSL.  (HyperText Transfer Protocol Secure) ― Защищённый протокол передачи данных в интернете. Это тот же протокол HTTP, но защищенный SSL-сертификатом.  Инфраструктура PKI: PKI — (Public Key Infrastructure). В PKI существует Удостоверяющий Центр, отдающий публичные части своих ключей клиентам, которым выдают подписанные ими сертификаты.  Алгоритм Диффи-Хеллмана: Алгоритм, который используется для безопасного обмена секретным ключом между двумя сторонами в режиме реального времени в незащищенной сети.  Репозиторий: – это место, где хранятся и поддерживаются какие-либо данные.  Git: это распределённая система управления версиями. Система, которая позволяет сразу нескольким разработчикам сохранять и отслеживать изменения в файлах вашего проекта.  GitHub: это веб-сервис основанный на системе контроля версий Git. |

**1 Настройка TLS**

В рамках данного блока необходимо обеспечить безопасное подключение к веб-серверу по протоколу HTTPS.

В начале работы была создана директория pki и был получен закрытый ключ удостоверяющего центра, находящийся в файле ca.key, сгенерированный по алгоритму RSA (рисунок 1.1).

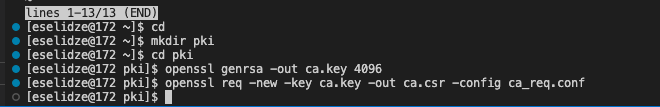


Рисунок 1.1 - Создание директории и генерация закрытого ключа УЦ

Далее был написан конфигурационный файл ca\_req.conf, необходимый для создания запроса на сертификат на его основе (рисунок 1.2).

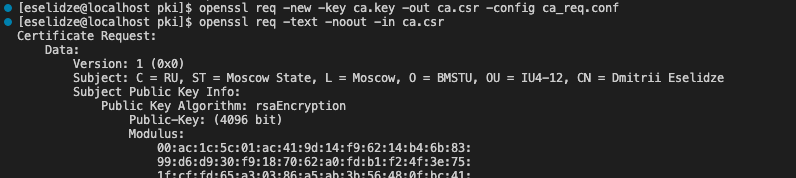


Рисунок 1.2 - Запрос на сертификат

Далее была выполнена команда для создания запроса на сертификат было проверено поле Requested Extensions (Рисунок 1.3).

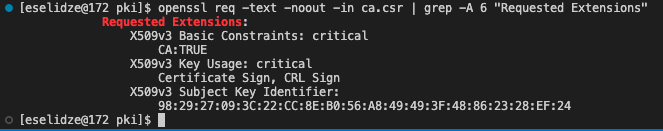


Рисунок 1.3 - Проверка поля Requested Extensions

Далее был создан еще один конфигурационный файл ca.conf, необходимый для описания параметров работы УЦ. Содержание файла показано на рисунке 1.4.

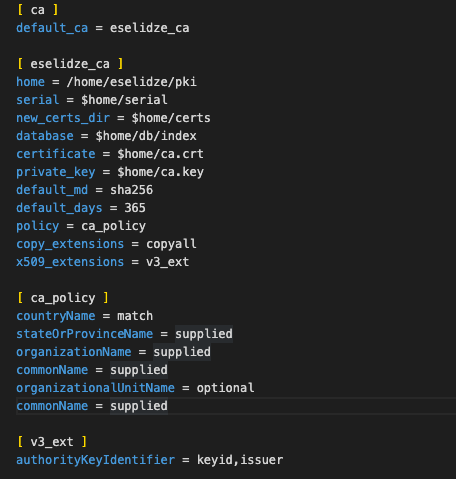


Рисунок 1.4 - Файл ca.conf

Далее были созданы необходимые служебные файлы и каталоги (рисунок 1.5).



Рисунок 1.5 - Создание служебных файлов и каталогов

Далее был сгенерирован самоподписанный сертификат УЦ, содержание которого показано на рисунке 1.6.

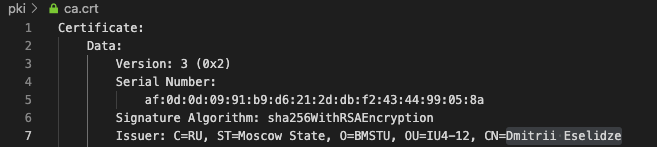


Рисунок 1.6 - Сертификат УЦ

Полученный сертификат был скачан на хост (рисунок 1.7).

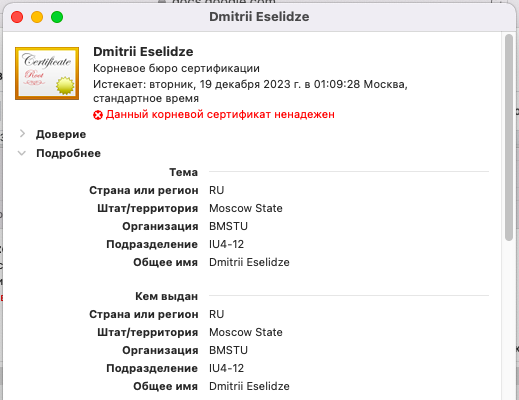


Рисунок 1.7 - Сведения о сертификате

После установки сертификата в хранилище доверенных корневых сертификатов было установлено, что ОС доверяет всем сертификатам, подписанным закрытым ключом (рисунок 1.8).

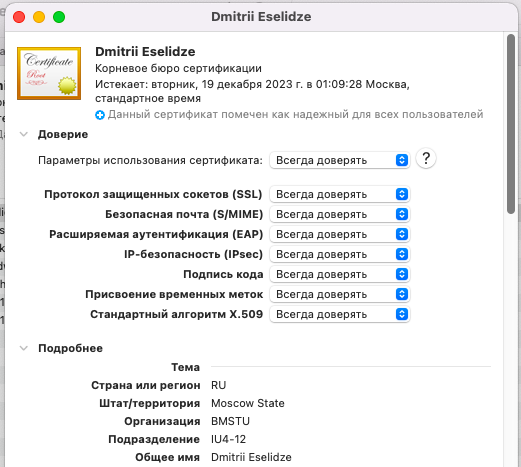


Рисунок 1.8 - Сведения о сертификате (с доверием ОС)

Далее был создан конфигурационный файл webserver\_req.conf, который будет отвечать за сертификат веб-сервера. Содержимое файла представлено на рисунке 1.9.

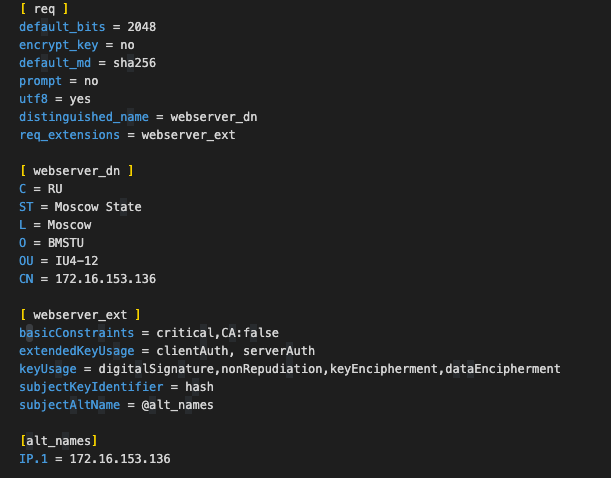


Рисунок 1.9 - Файл webserver\_req.conf

Далее был создан закрытый ключ и запрос на сертификат веб-сервера (Рисунок 1.10).



Рисунок 1.10 - Команды для генераций и сам процесс

Далее был выпущен сертификат веб-сервера, подписанный закрытым ключом УЦ (Рисунок 1.11).

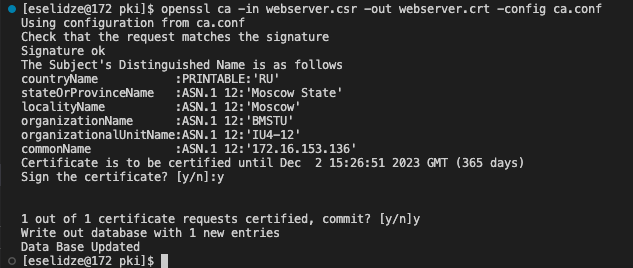


Рисунок 1.11 - Генерация сертификата веб-сервера

Далее было проведено сравнение значения атрибута х509v3 Authority Key Identifier и идентификатор ключа УЦ, которым подписан сертификат (рисунок 1.12).

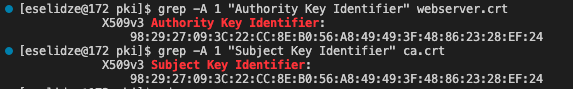


Рисунок 1.12 - Сравнение необходимых значений

После копирования сертификата и закрытого ключа веб-сервера в папку ssl, находящуюся в /etc/nginx, в этой же папке был создан файл dhparam.prem, в котором хранится ключевая информация для поддержки технологии Forward Secrecy (рисунок 1.13).

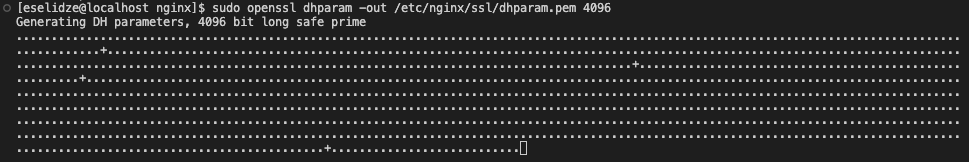


Рисунок 1.13 - Создание файла dhparam.prem

Далее был создан файл ssl\_params, в который были помещены настройки TSL (рисунок 1.14).

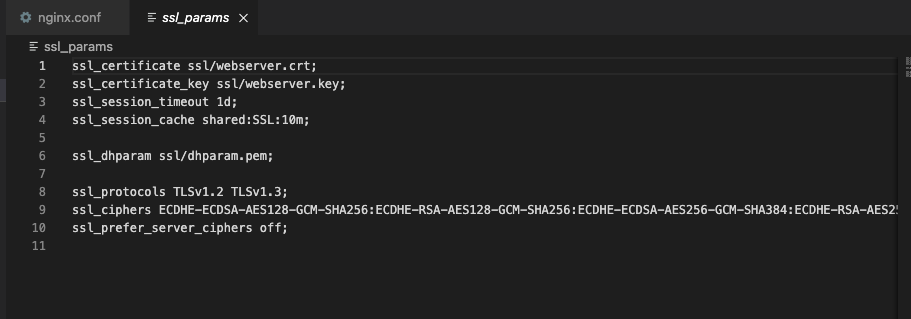


Рисунок 1.14 - Содержимое файла ssl\_params

Далее были внесены изменения в конфиг nginx (рисунок 1.15).

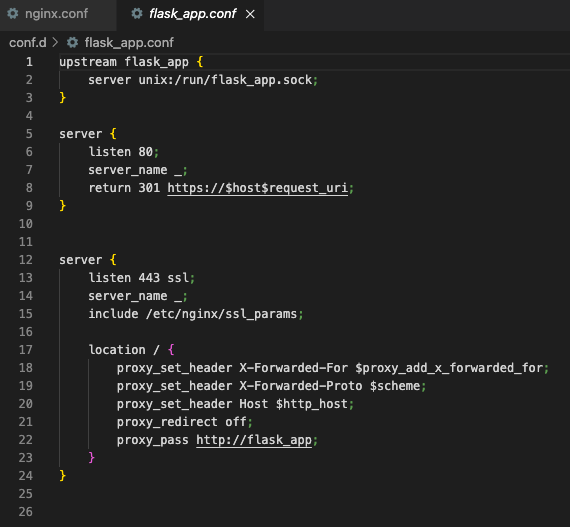


Рисунок 1.15 - Содержимое файла flask\_app.conf

После перезагрузки nginx было выявлено, что протокол HTTPS работает корректно (рисунок 1.16).

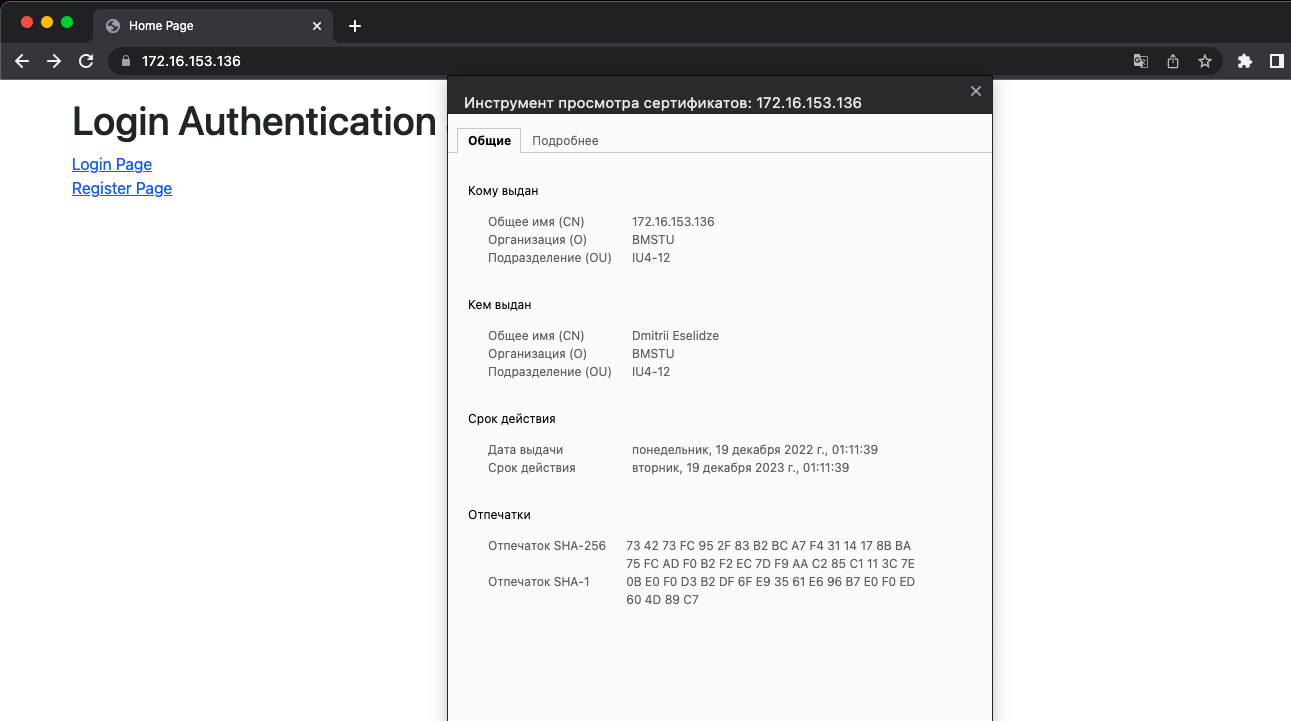


Рисунок 1.16 - Проверка безопасности соединения

На данном этапе настроена инфраструктура открытых ключей, созданы сертификаты УЦ и веб-сервера, а сам веб-сервер настроен на работу по протоколу https.

**2 Работа с системами контроля версий**

В данном блоке необходимо произвести операции с коммитами через терминал.

Для начала работы были указаны сведения об авторе (рисунок 2.1).



Рисунок 2.1 - Данные автора

Далее была создана директория нового проекта, а в ней - новый репозиторий (рисунок 2.2).

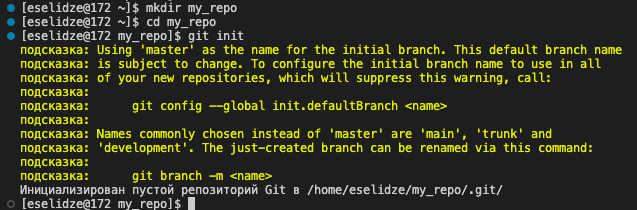


Рисунок 2.2 - Создание директории и репозитория

После этого было выявлено отсутствие коммитов (рисунок 2.3).

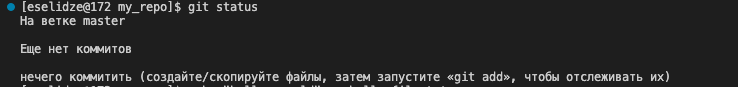


Рисунок 2.3 - Отсутствие коммитов

Далее был создан и добавлен в репозиторий новый файл (рисунок 2.4).

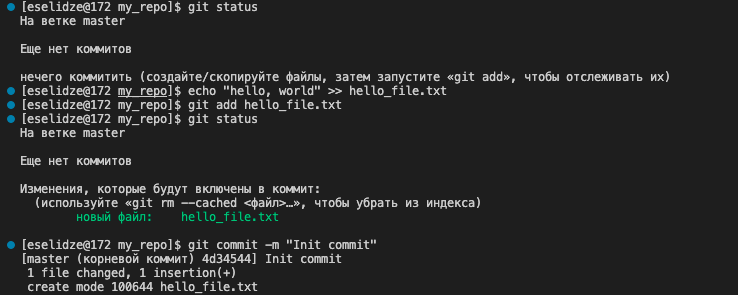


Рисунок 2.4 - Создание файла и добавление его в репозиторий

Далее был произведен коммит файла (рисунок 2.5).

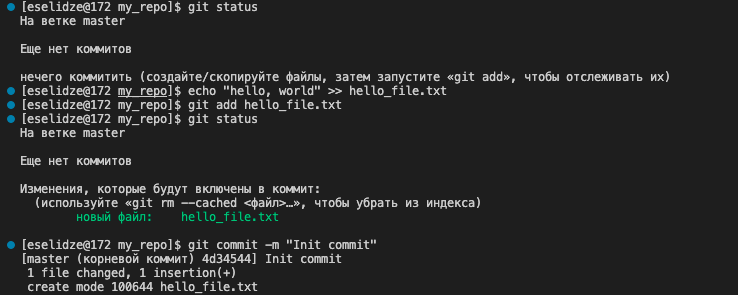


Рисунок 2.5 - Коммит файла

Далее были внесены изменения в файл hello\_file.txt (рисунок 2.6).

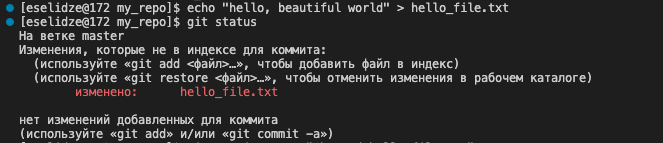


Рисунок 2.6 - Изменение содержания файла

После чего были зафиксированы все изменения (рисунок 2.7).



Рисунок 2.7 - Фиксирование изменений

Далее были просмотрены изменения конкретного коммита по его номеру (рисунок 2.8).

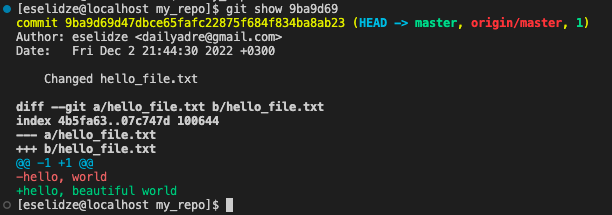


Рисунок 2.8 - Изменения конкретного коммита

Далее репозиторий был возвращен в состояние до версии fe39a9a (рисунок 2.9).



Рисунок 2.9 - Файл до и после изменений

Далее была просмотрена история коммитов (рисунок 2.10).

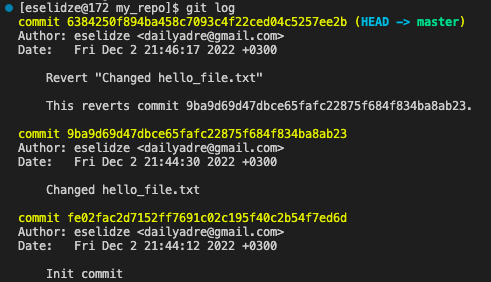


Рисунок 2.10 - История коммитов

После этого был удален последний коммит (рисунок 2.11).



Рисунок 2.11 - Удаление коммита

После чего история коммитов была просмотрена еще раз с целью проверки прошлого действия (рисунок 2.12).

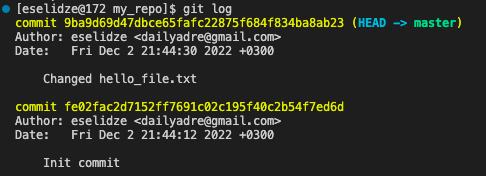


Рисунок 2.12 - Повторная проверка истории коммитов

На данном этапе были произведены операции по откатыванию до определенного коммита и удалению коммитов.

**3 Работа с GitHub**

В данном блоке необходимо будет предоставить доступ к проекту другим пользователям с помощью GitHub.

Для начала был создан публичный репозиторий доступный по ссылке: https://github.com/dima-code/lab\_project (рисунок 3.1).

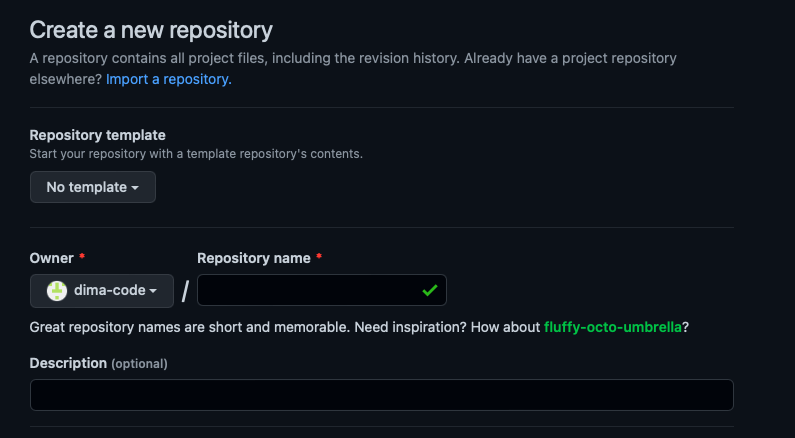


Рисунок 3.1 - Создание публичного репозитория

Далее было произведено подключение к публичному репозиторию (рисунок 3.2).

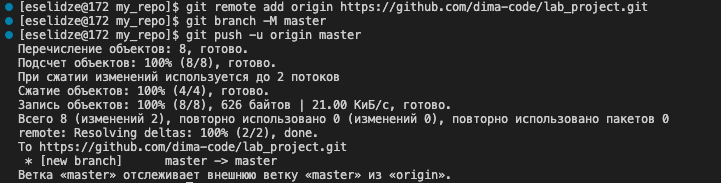


Рисунок 3.2 - Подключение к публичному репозиторию

На данном этапе на сайте можно просматривать загруженные файлы (рисунок 3.3).

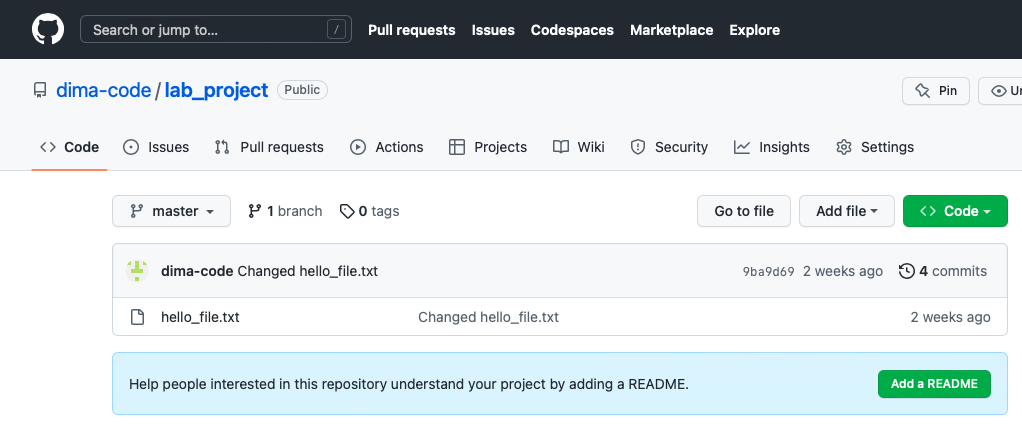


Рисунок 3.3 - Веб-страница репозитория

Также есть возможность просматривать историю коммитов (рисунок 3.4).

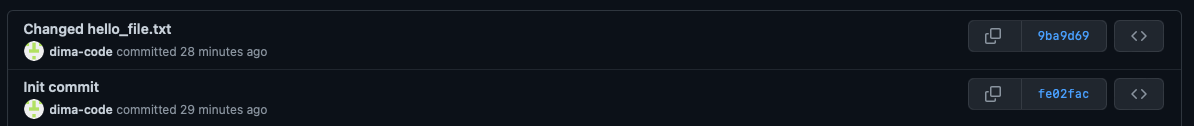


Рисунок 3.4 - История коммитов

По завершении работы мы научились работать с протоколом безопасного подключения HTTPS, что поможет в будущих проектах с использованием веб-серверов. Также мы научились создавать репозитории как локальные, так и открытые на GitHub. С помощью него можно будет работать в команде или большой компании.