* **Вариативные задания 1**
* **Задание 2.1**

Задание выполняется после проведения вебинара «Использование системы контроля версий Git».

На примере основных действий, продемонстрированных на вебинаре по Git, освоить работу (и продемонстрировать выполнение показанных на вебинаре действий) с одним из визуальных клиентов для работы с Git на выбор:

**Вариант 4.** Терминал (Terminal) или Командная строка (или Command Shell Git).

1. cd + путь, где будет находиться репозиторий - Выбор места, где будет находиться проект на диске.

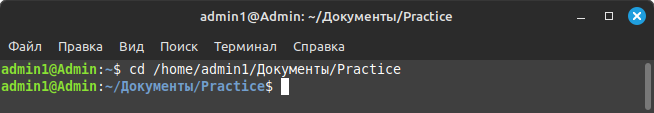


Рисунок 1.

1. git init - Инициализирует пустой репозиторий.

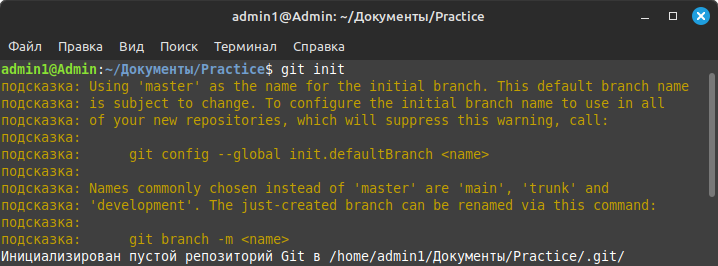


Рисунок 2.

1. ls -la - Позволяет увидеть, что репозиторий создан.

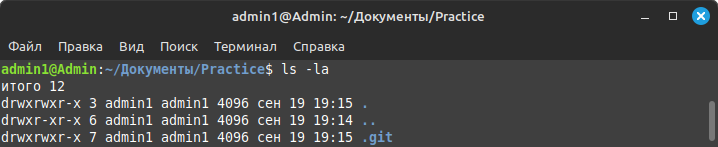


Рисунок 3.

1. git status - Показывает статус репозитория.

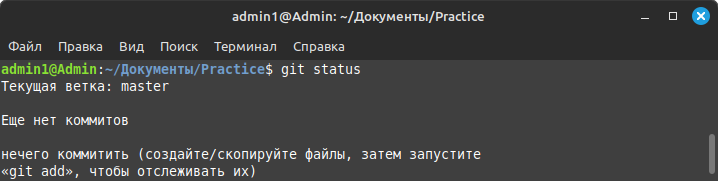


Рисунок 4.

1. touch + название файла - Создаёт файл в репозитории.

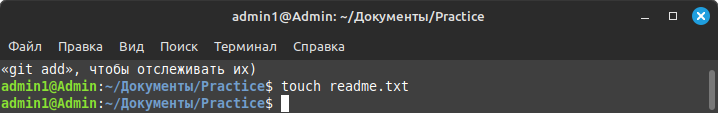


Рисунок 5.

Файл создан.

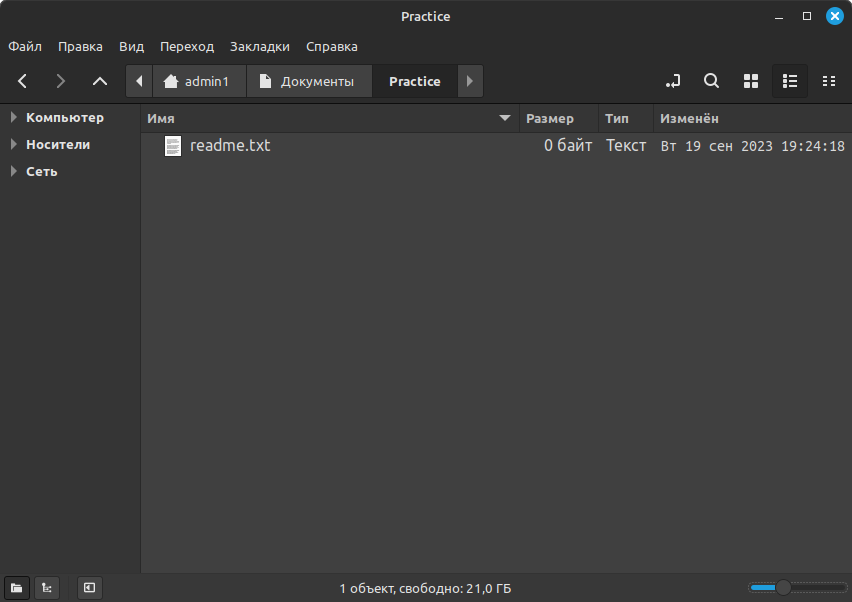


Рисунок 6.

1. nano readme.txt - Позволяет отредактировать файл readme.txt в gnu интерфейсе.

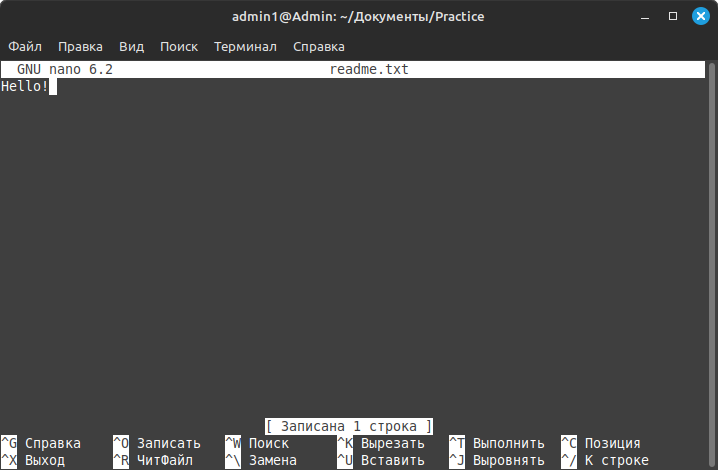


Рисунок 7.

Отредактированный файл в каталоге.

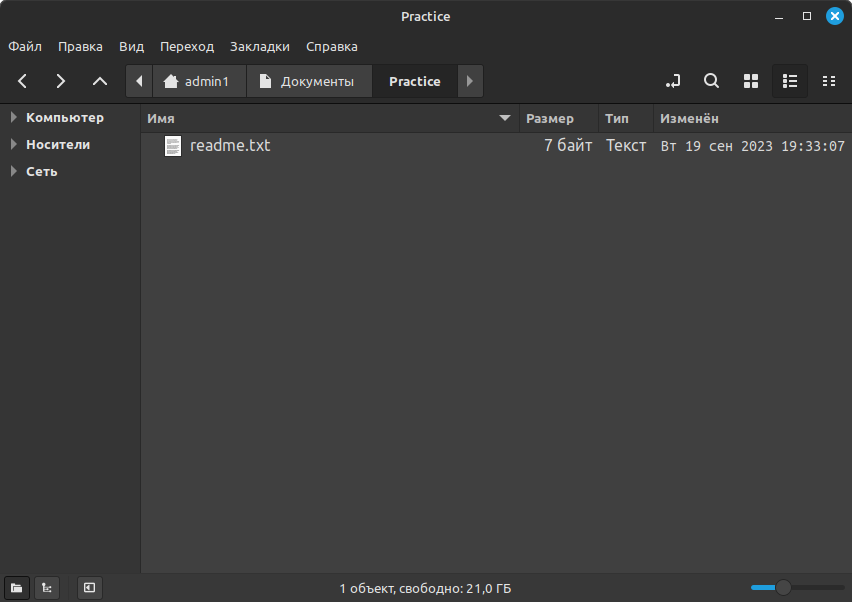


Рисунок 8.

1. git status - Позволяет зафиксировать изменения. Файл создан, но не отслеживается системой.

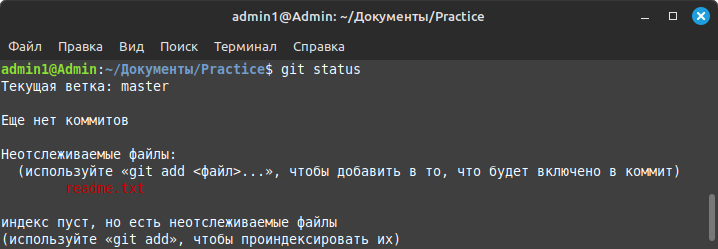


Рисунок 9.

1. git add + имя файла - Делает файл отслеживаемым.

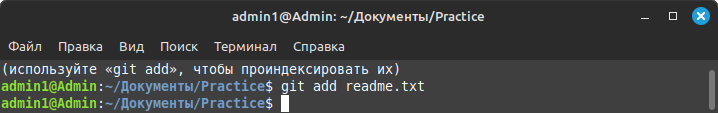


Рисунок 10.

1. git status - Проверяет новый статус файла. Файл становиться отслеживаемым.

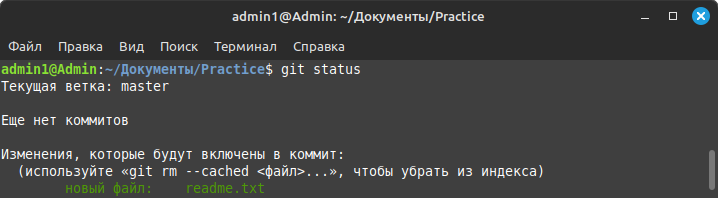


Рисунок 11.

1. git commit -m “First file” - Фиксирует статус файла. Тем самым он хранится в репозитории.

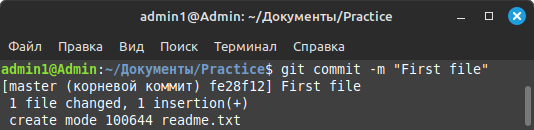


Рисунок 12.

1. git status - Показывает статус файла. Т.к файлов не было добавлено, нечего фиксировать.

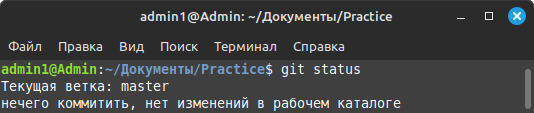


Рисунок 13.

1. nano readme.txt - Ещё раз выполняем команду, чтобы изменить файл.

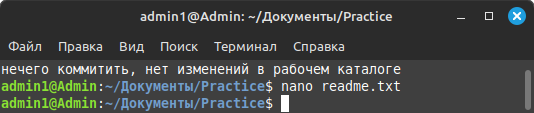


Рисунок 14.

Изменяем файл. Добавляем слово world.

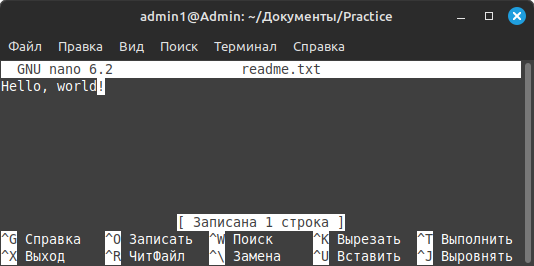


Рисунок 15.

1. git status - Ещё раз проверяем статус файла. Файл отслеживается, но изменения не отмечены для добавления в commit.

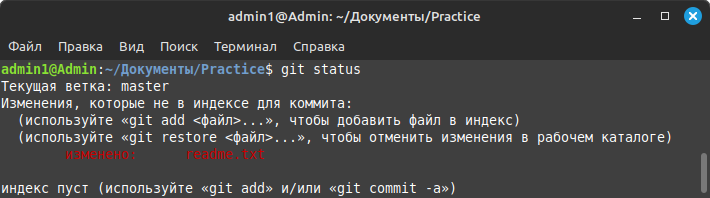


Рисунок 16.

1. git add + имя файла - Добавляем файл в отслеживаемые staged area .

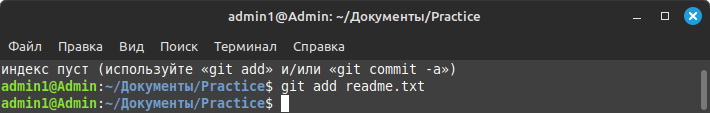


Рисунок 17.

1. git add ‘\*.txt’ - добавляет только файлы текстового формата.

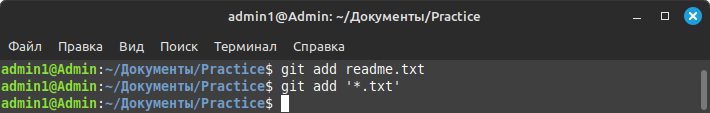


Рисунок 18.

1. Продублируем файлы формата txt.

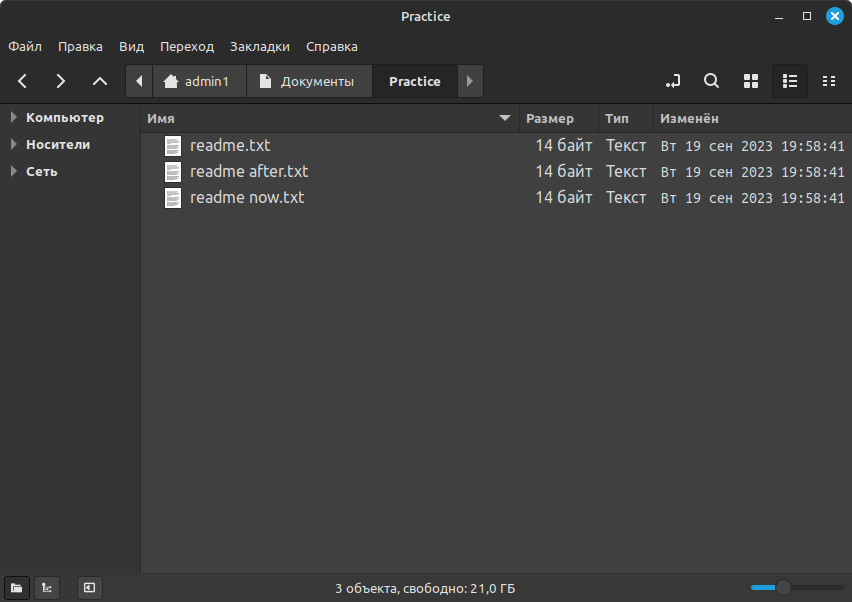


Рисунок 19.

1. git status - Смотрим статус добавленных файлов.

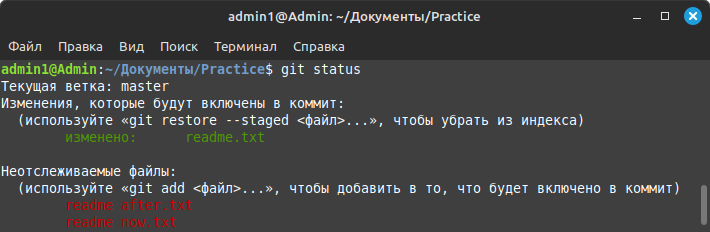


Рисунок 20.

1. git add ‘\*.txt” - Добавляем файлы текстового формата в staged area.

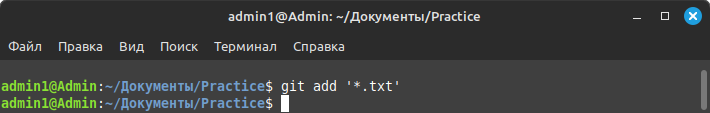


Рисунок 21.

1. git status - Проверяем статус новых файлов.

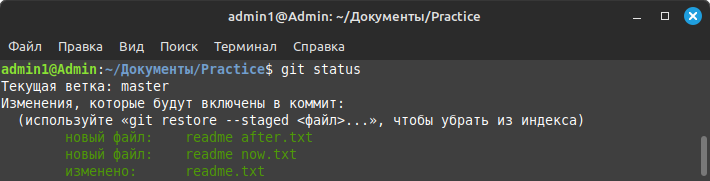


Рисунок 22.

1. git commit -m “All files with txt extension were added” - Фиксируем статус файлов.

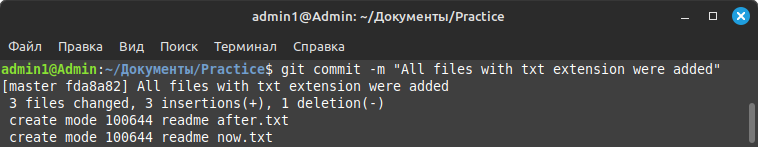


Рисунок 23.

1. git log - Показывает историю всех commit в обратном хронологическом порядке.

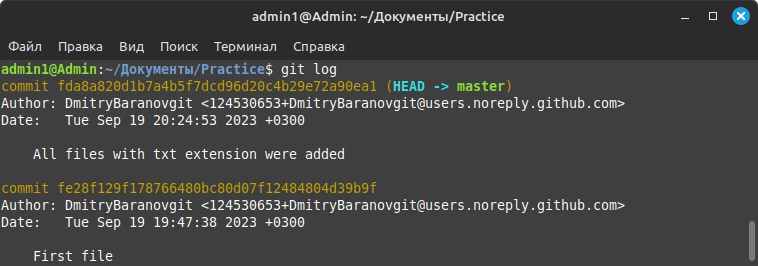


Рисунок 24.

1. git log –summary - Показывает подробную историю всех изменений.

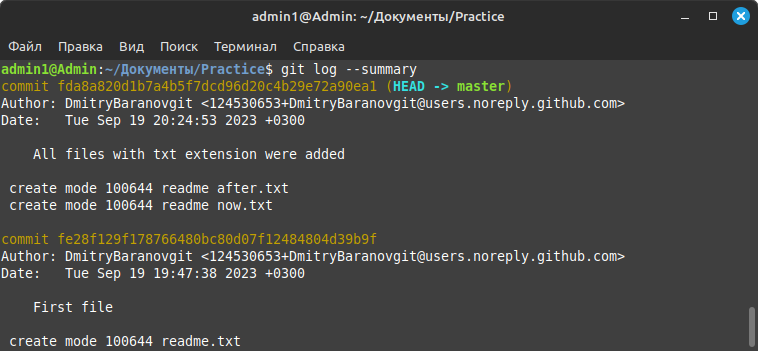


Рисунок 25.

1. git remote add origin https://github.com/DmitryBaranovgit/VSR2.1.git - Добавляет удалённый репозиторий.

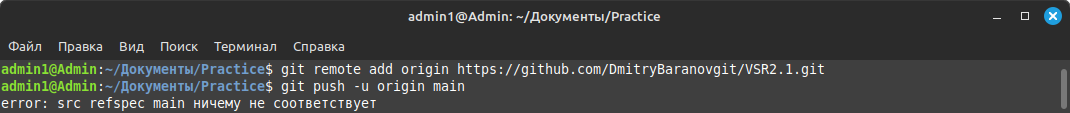


Рисунок 26.

1. Кликнув по аватарке в правом верхнем углу зайти в настройки GitHub.

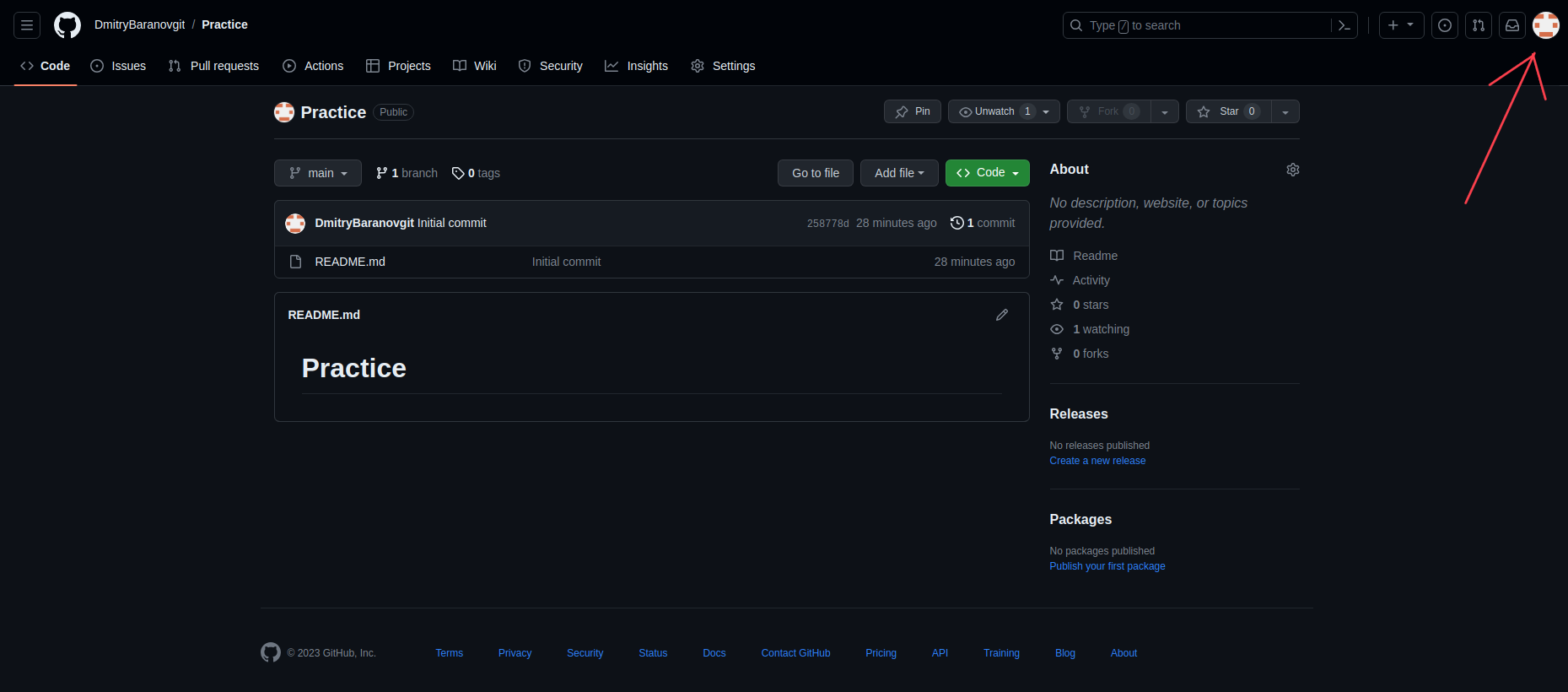


Рисунок 27.

Настройки.

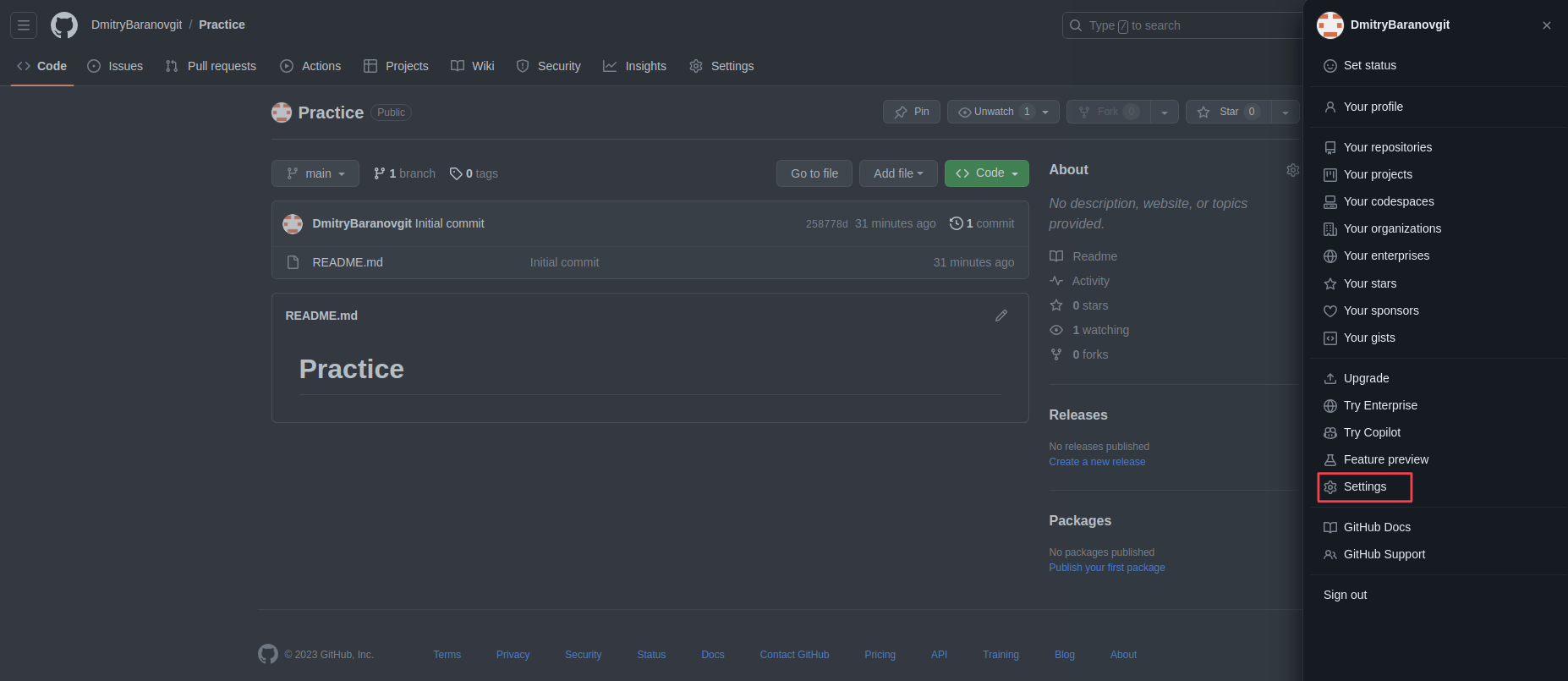


Рисунок 28.

1. Затем открыть настройки для разработчиков.

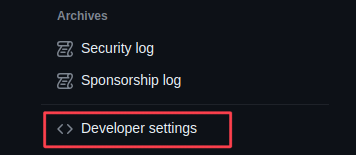


Рисунок 29.

Сгенерировать ключ.

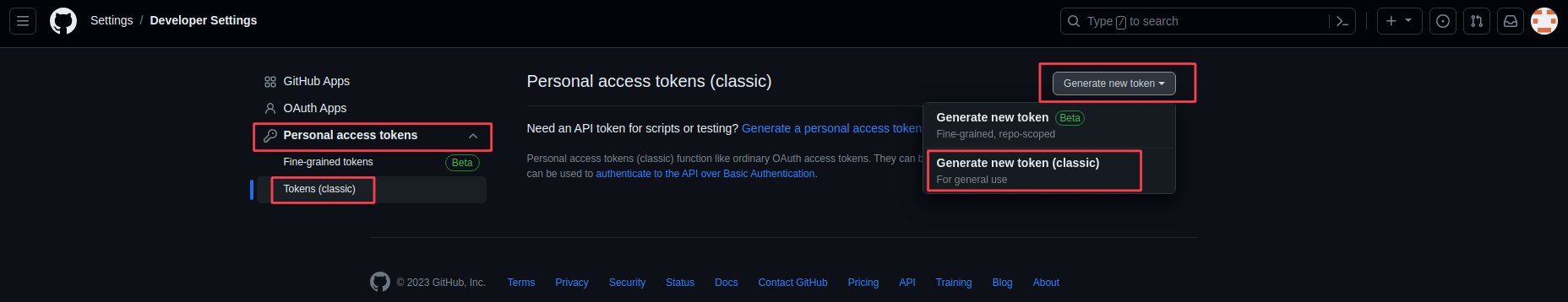


Рисунок 30.

Задать параметры.

Отметить операции с репозиторием.

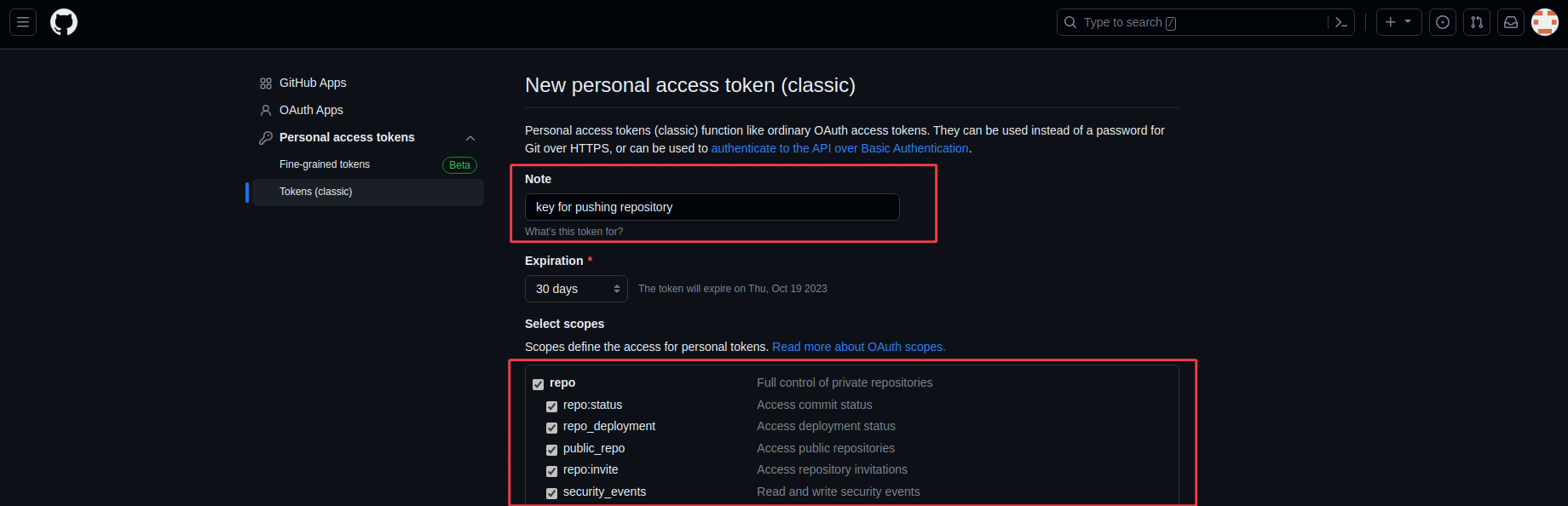


Рисунок 31.

Отметить операции администратора.

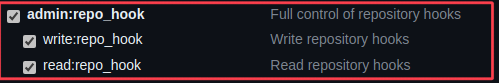


Рисунок 32.

Отметить удаление репозитория.



Рисунок 33.

Нажать создать ключ и скопировать сам ключ.



Рисунок 34.

1. git push -u origin + название ветки - Выгрузка в удалённый репозиторий.

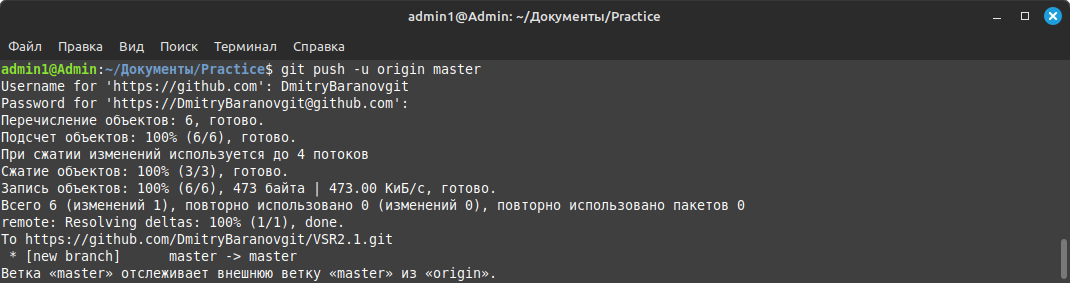


Рисунок 35.

1. Смотрим выгруженные файлы в GitHub.

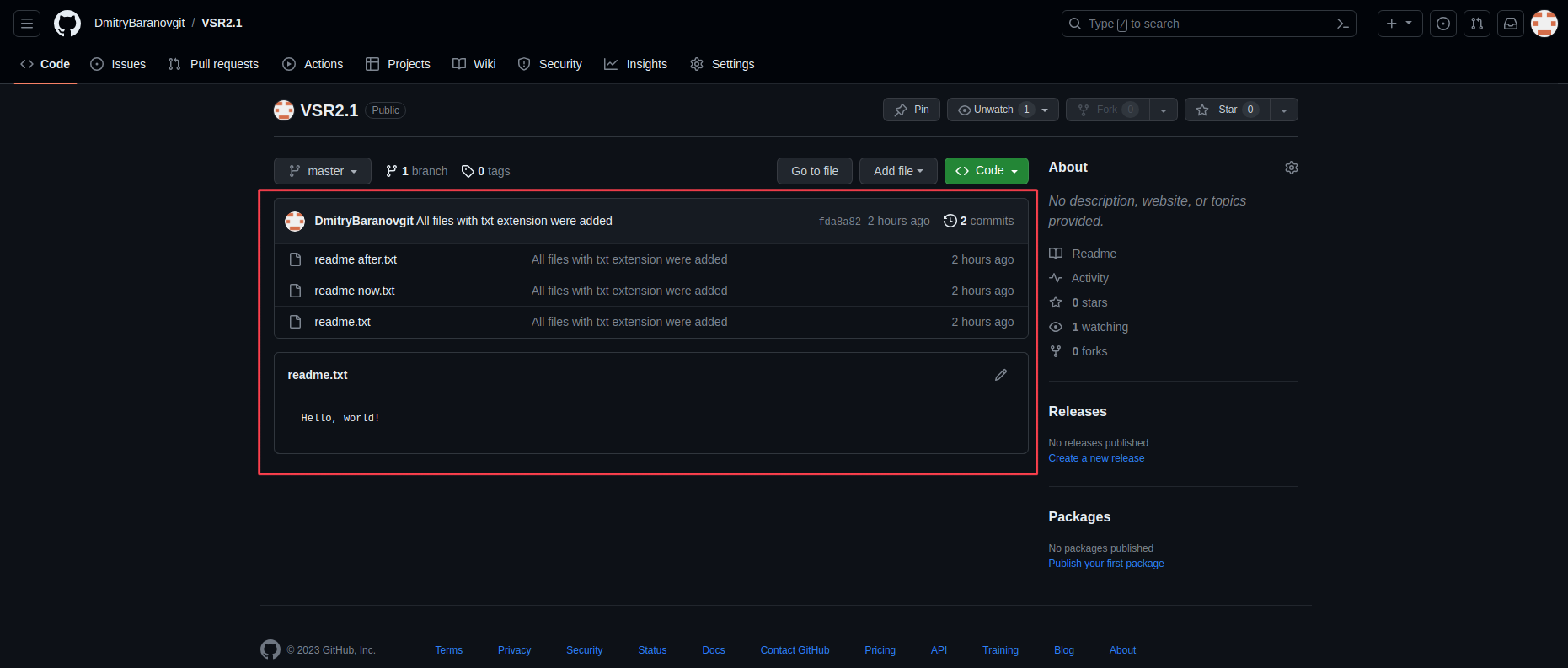


Рисунок 36.

1. Изменим в браузере readme.txt, нажав на специальный значок.

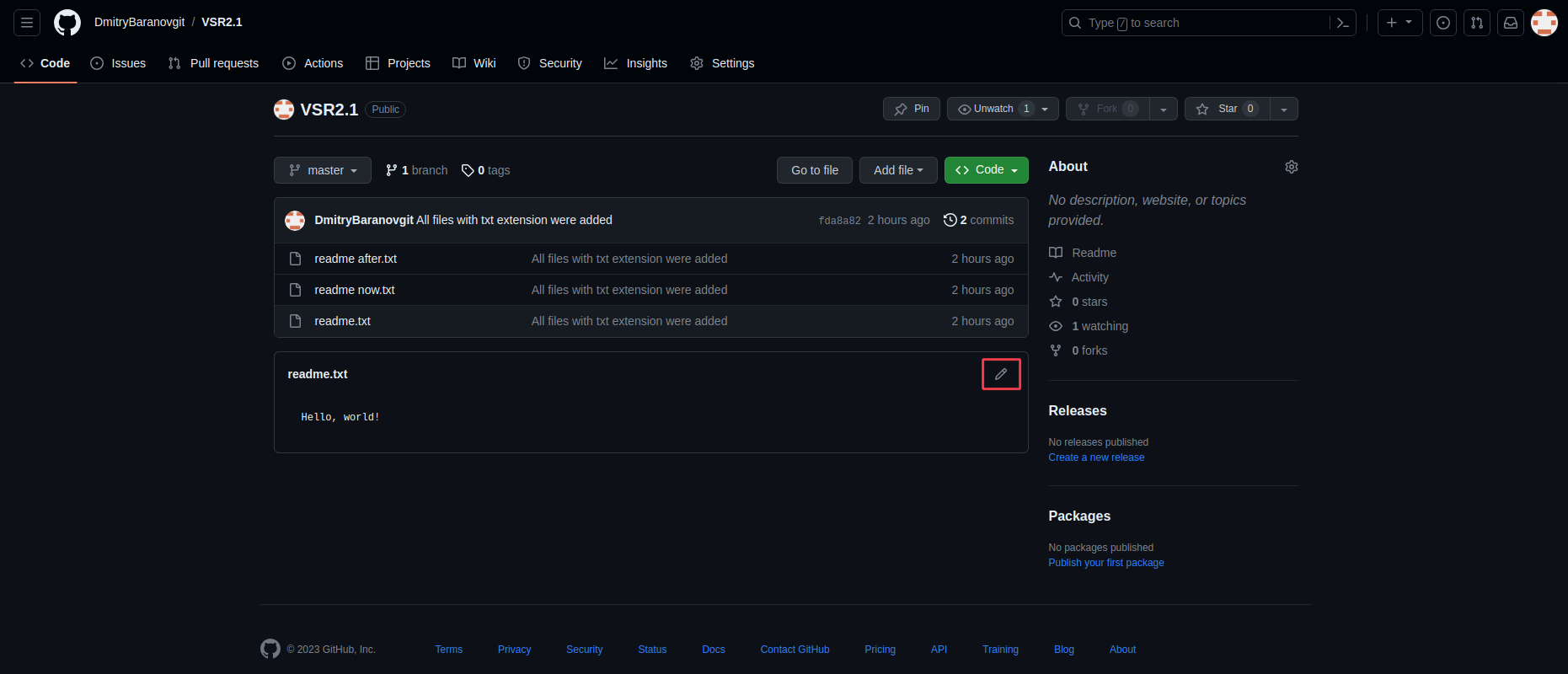


Рисунок 37.

Напишем ещё 2 раза Hello, world! и сохраним изменения.

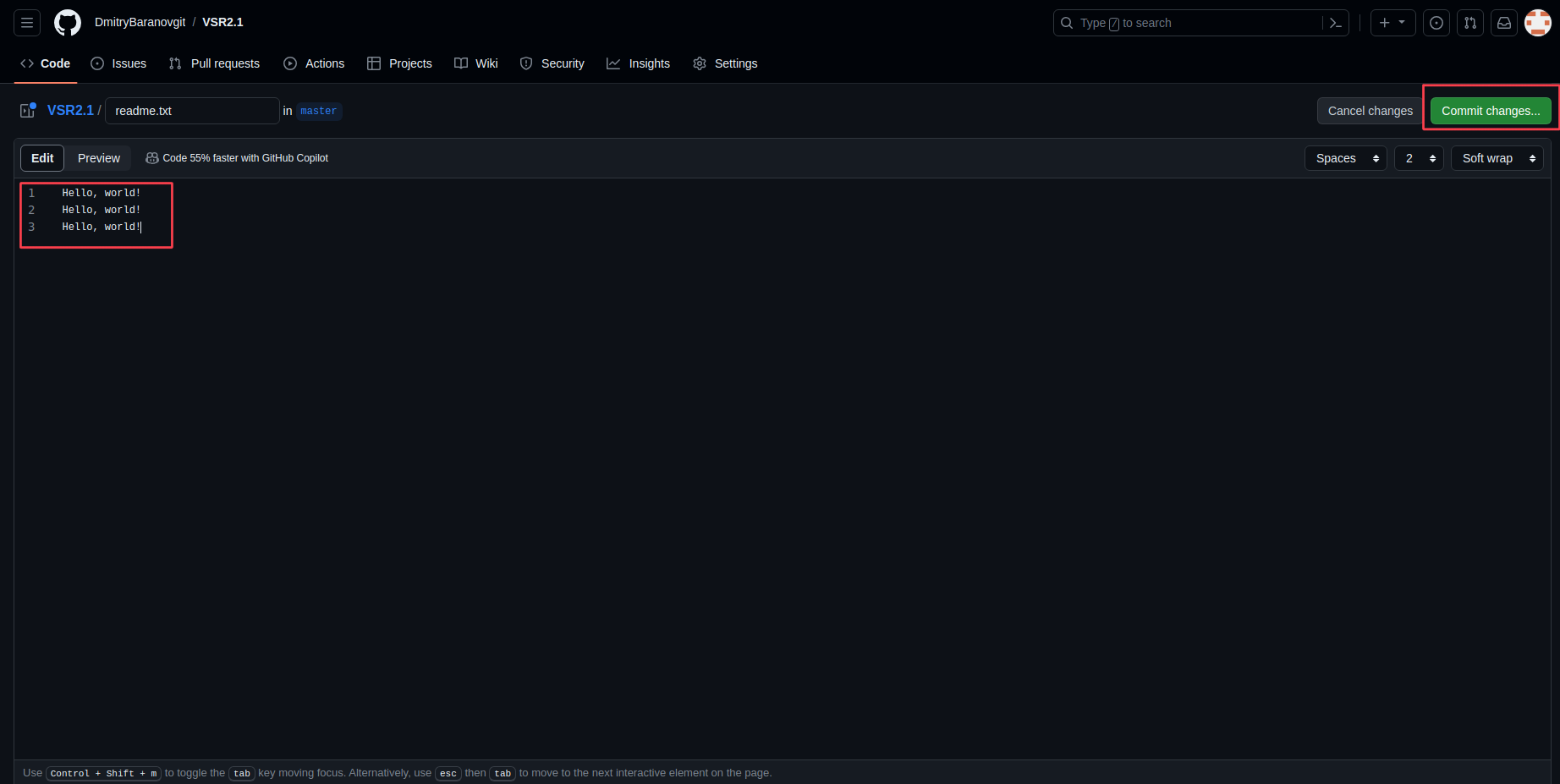


Рисунок 38.

1. git pull origin master - Позволяет посмотреть сделанные изменения в удаленном репозитории. Добавлено две новые строчки.

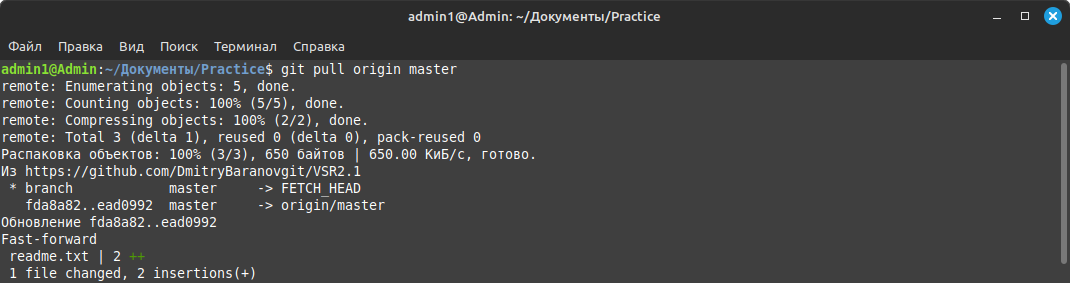


Рисунок 39.

1. nano readme.txt - Добавим изменения в файл

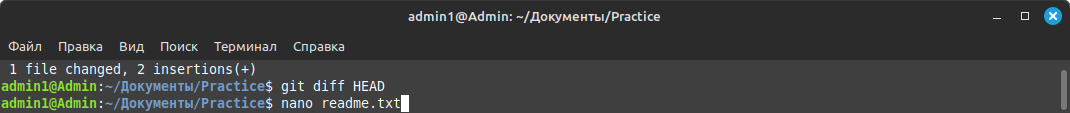


Рисунок 40.

Изменяем файл. Удаляем ! знаки.

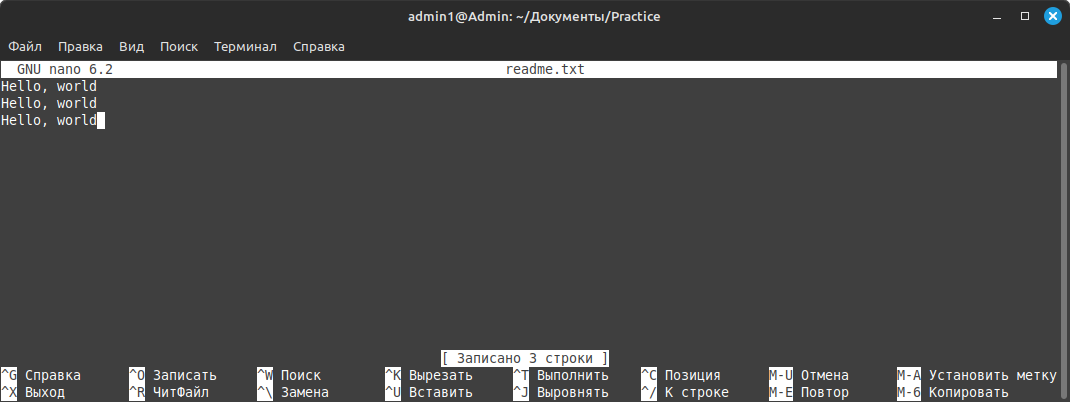


Рисунок 41.

1. git diff HEAD - Позволяет увидеть изменения. HEAD - указатель на последнюю зафиксированную версию.

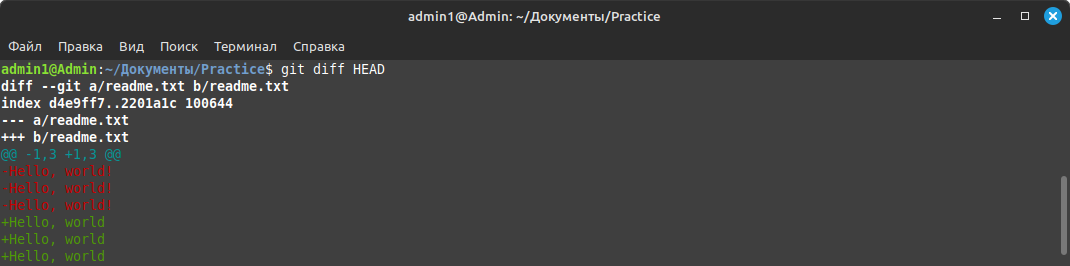


Рисунок 42.

1. mkdir folder - Добавляет каталог folder.

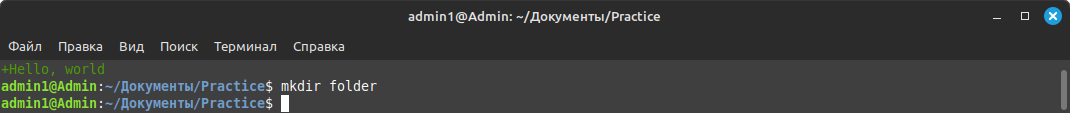


Рисунок 43.

1. touch folder/bye.txt - Добавляет файл bye.txt в каталог folder.

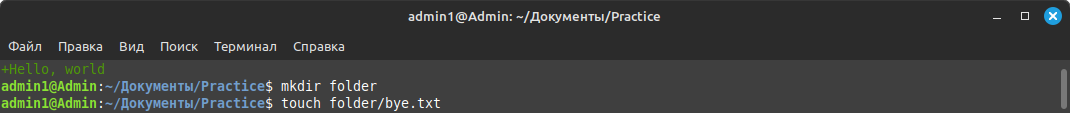


Рисунок 44.

1. git status - Показывает статус. Каталог сделан, но не отслеживается в репозитории.

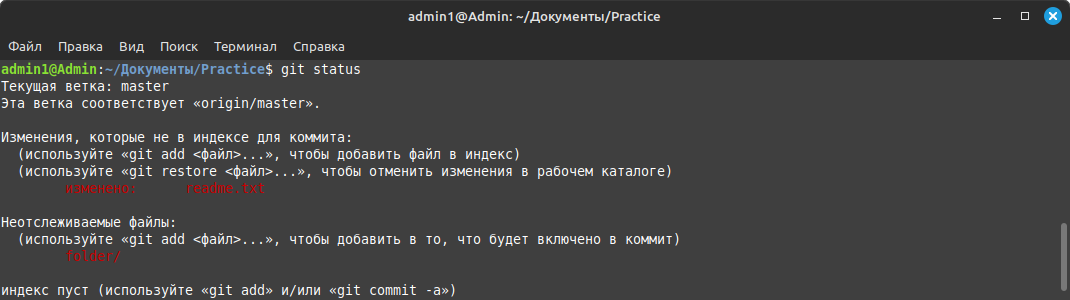


Рисунок 45.

Посмотрим каталог.

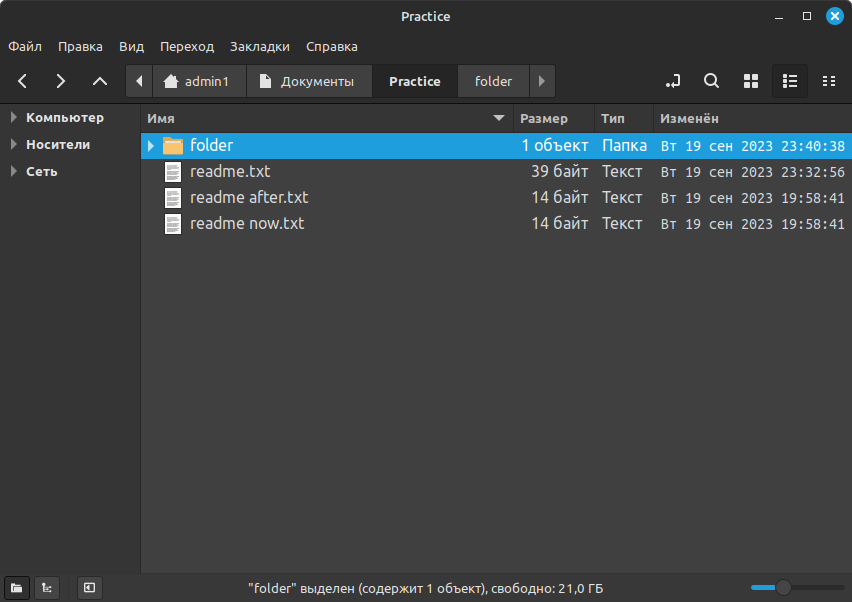


Рисунок 46.

И файл.

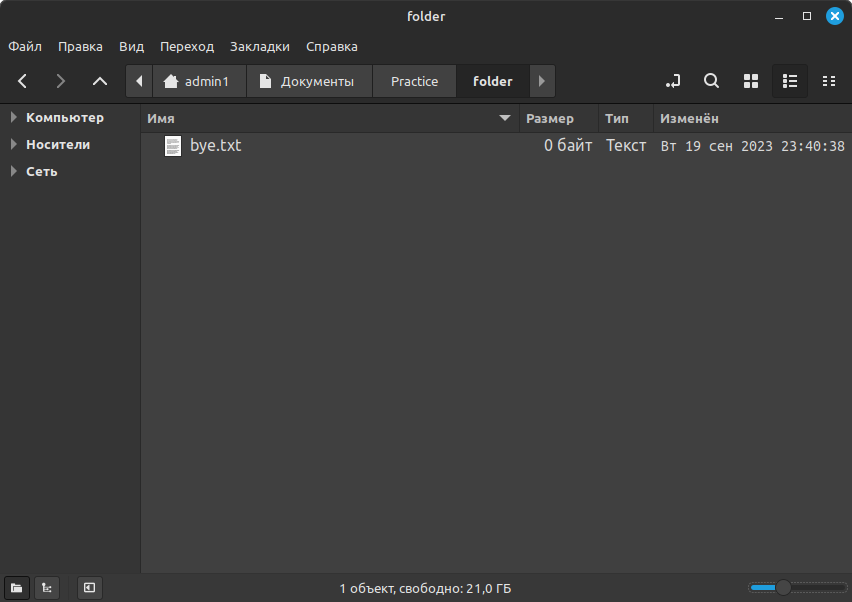


Рисунок 47.

1. git add folder/bye.txt - Добавляет каталог и файл в отслеживаемые.

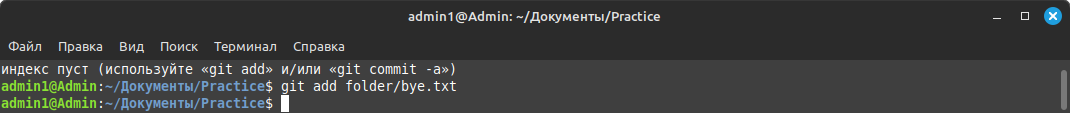


Рисунок 48.

1. git status - Статус файла.

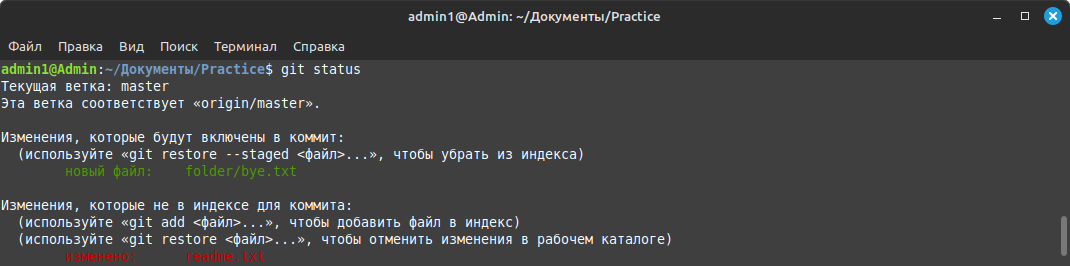


Рисунок 49.

1. git diff –staged - Позволяет посмотреть изменения в стадии staged.

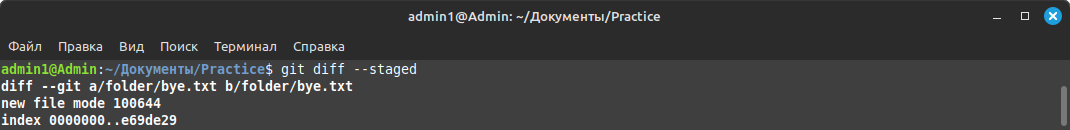


Рисунок 50.

1. git reset folder/bye.txt - Отменяет действие.

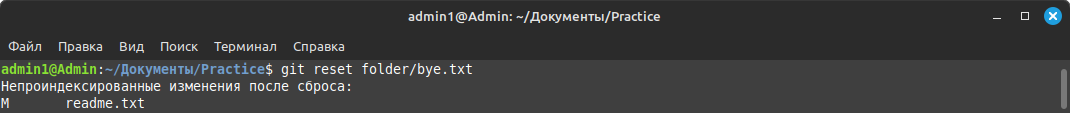


Рисунок 51.

1. git diff –staged - Вернёт пустое значение.

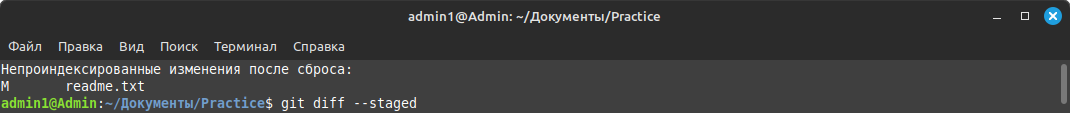


Рисунок 52.

1. git diff - Показывает, что было без выполнения команды git staged.

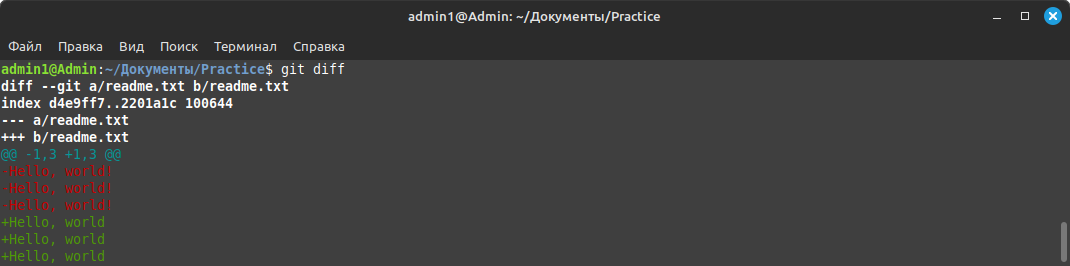


Рисунок 53.

1. git status - Показывает, что снялось отслеживание с readme.txt.

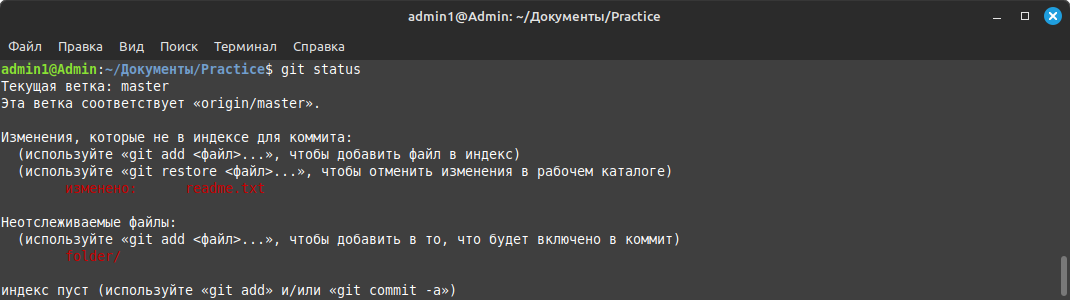


Рисунок 54.

1. git checkout – readme.txt - Возвращает изменения для файла readme.txt.

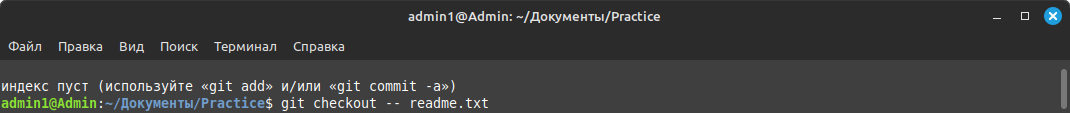


Рисунок 55.

1. git status - Показывает, что последние изменения файла убраны.

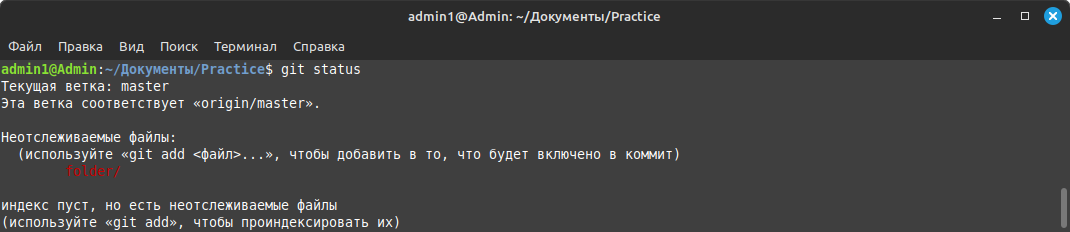


Рисунок 56.

1. cat readme.txt - Показывает файл до изменений.

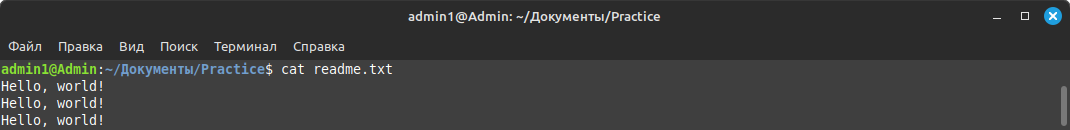


Рисунок 57.

1. git branch clean\_up - Создаёт ветку clean\_up.

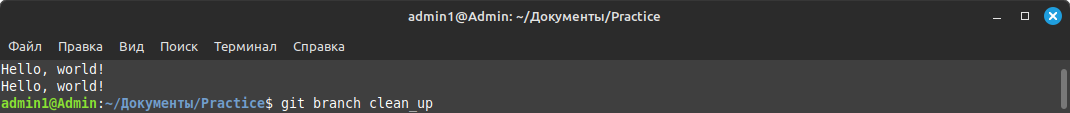


Рисунок 58.

1. git branch - Показывает в какой ветке идёт работа.

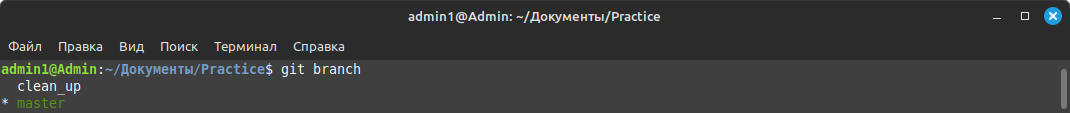


Рисунок 59.

1. git checkout clean\_up - Переключает на ветку clean\_up.

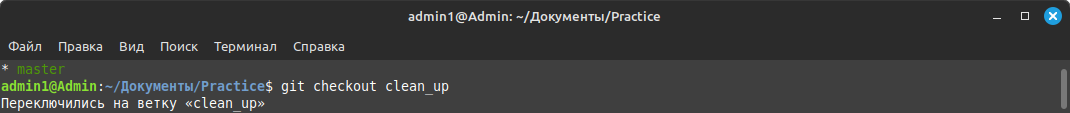


Рисунок 60.

1. git branch - Показывает, что переключилось на ветку clean\_up.

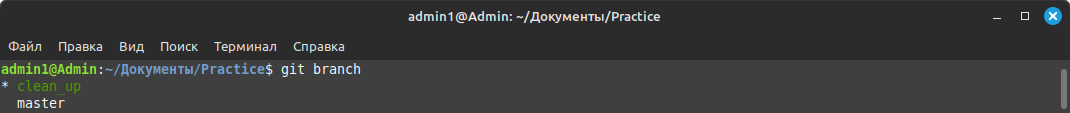


Рисунок 61.

1. ls - Показывает каталог.

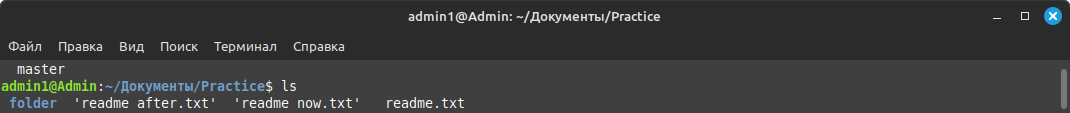


Рисунок 62.

Каталог.

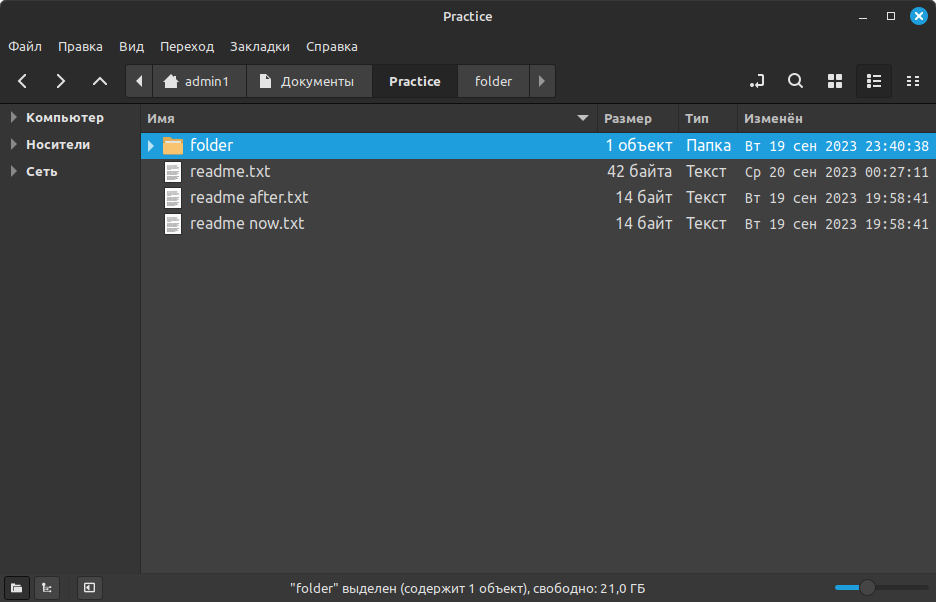


Рисунок 63.

Файл каталога.

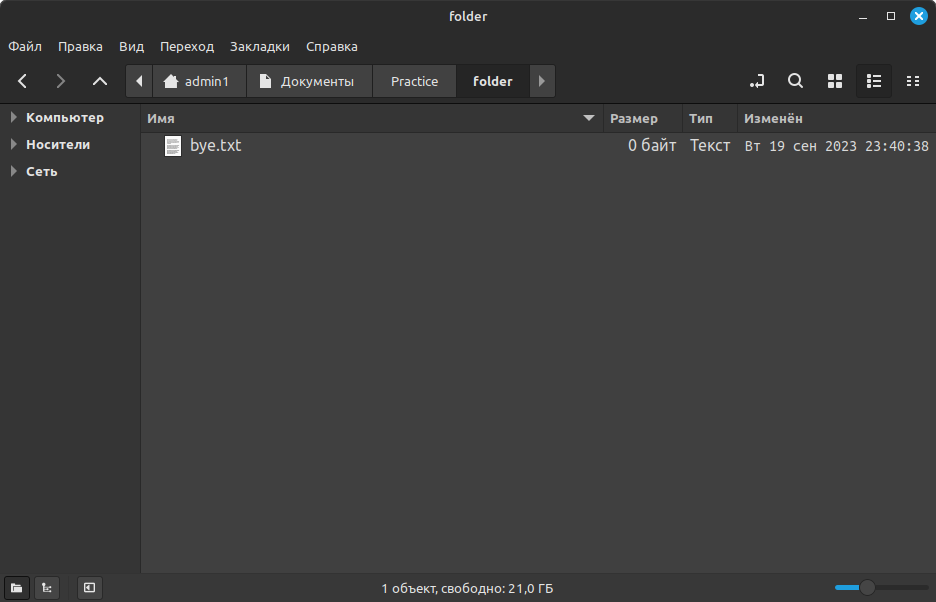


Рисунок 64.

1. rm -r folder - Удаляет каталог folder и файлы в нём.

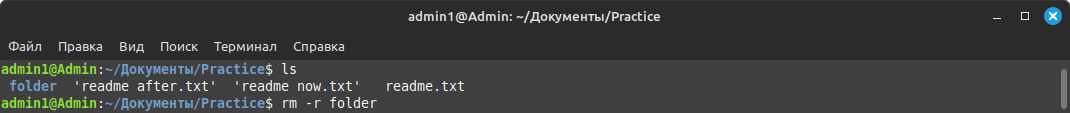


Рисунок 65.

Каталог.

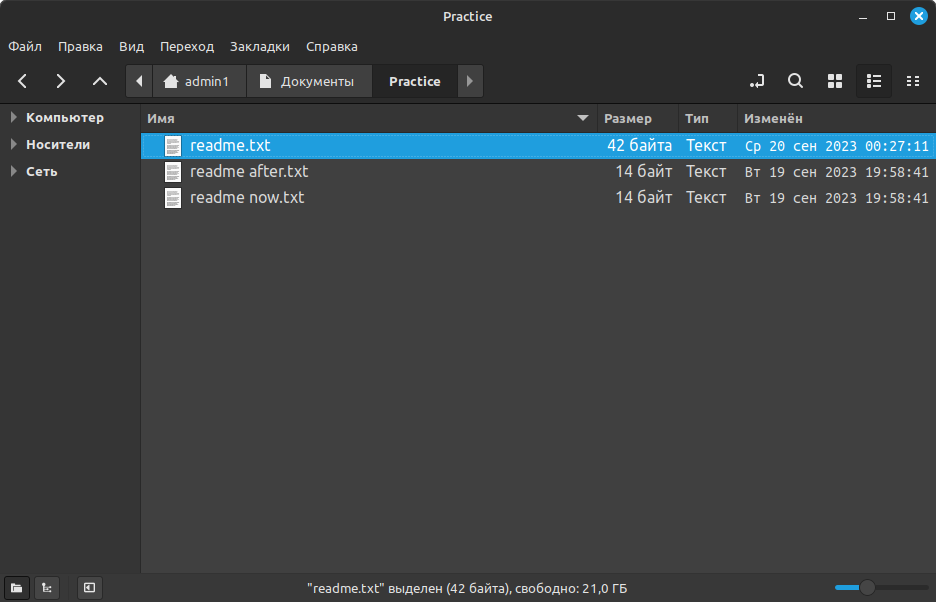


Рисунок 66.

1. git commit -m ‘Deleted folder and files’ - Фиксирует изменения.

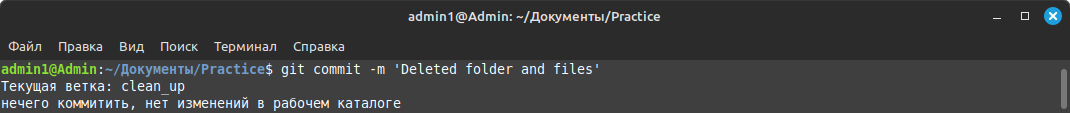


Рисунок 67.

1. git status - Статус репозитория.

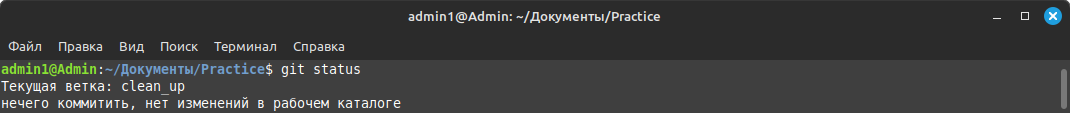


Рисунок 68.

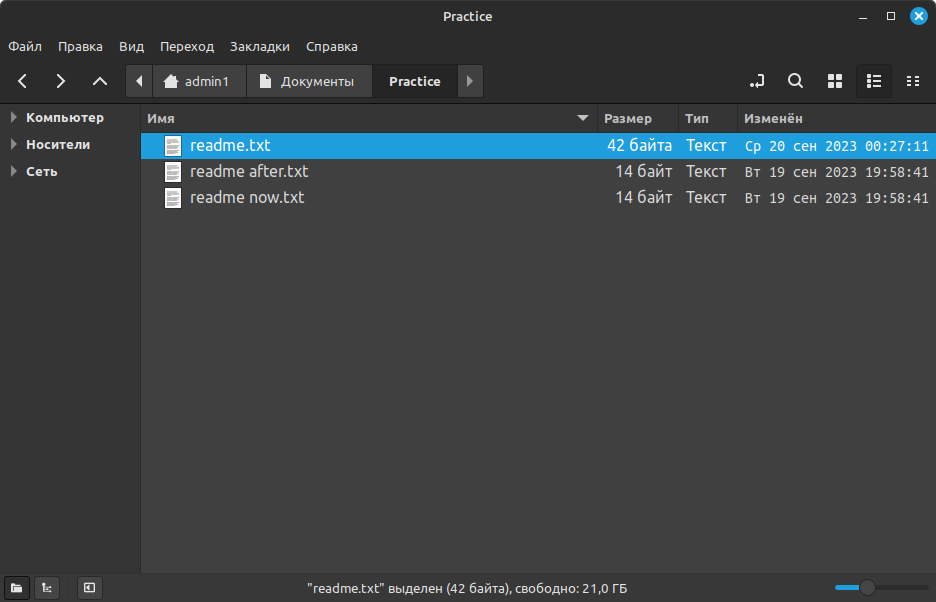


Рисунок 69.

1. git rm readme\ after.txt - Удаляет файл after.txt.

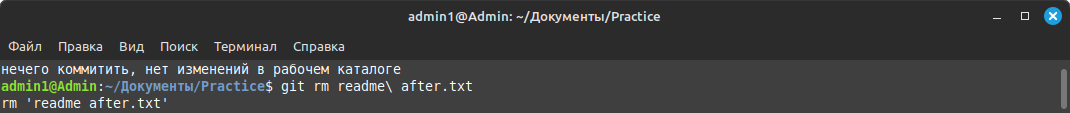


Рисунок 70.

Каталог.

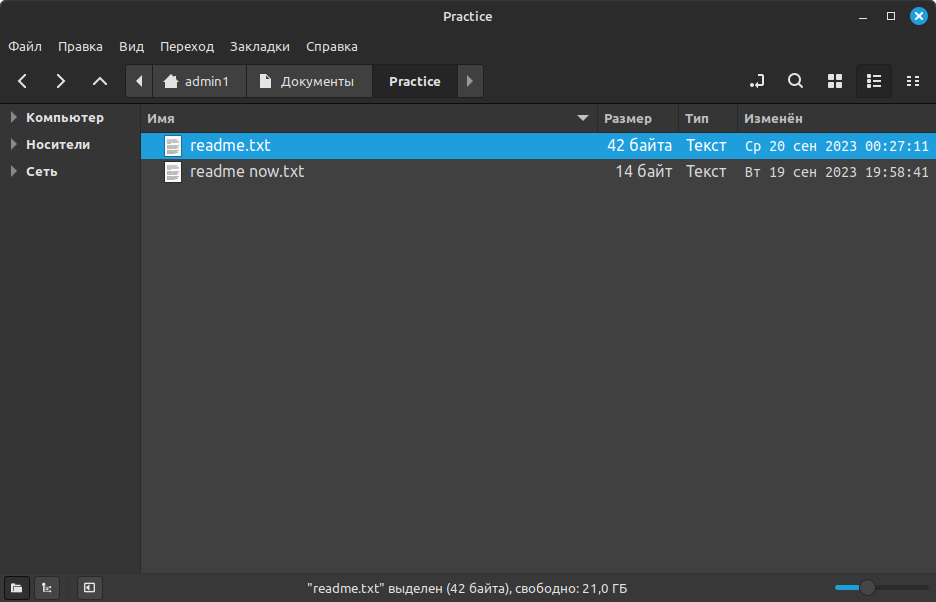


Рисунок 71.

1. git rm readme\ now.txt - Удаляет now.txt.

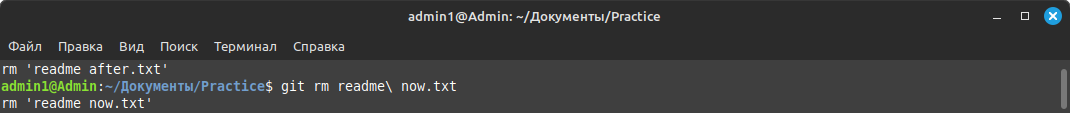


Рисунок 72.

Каталог.

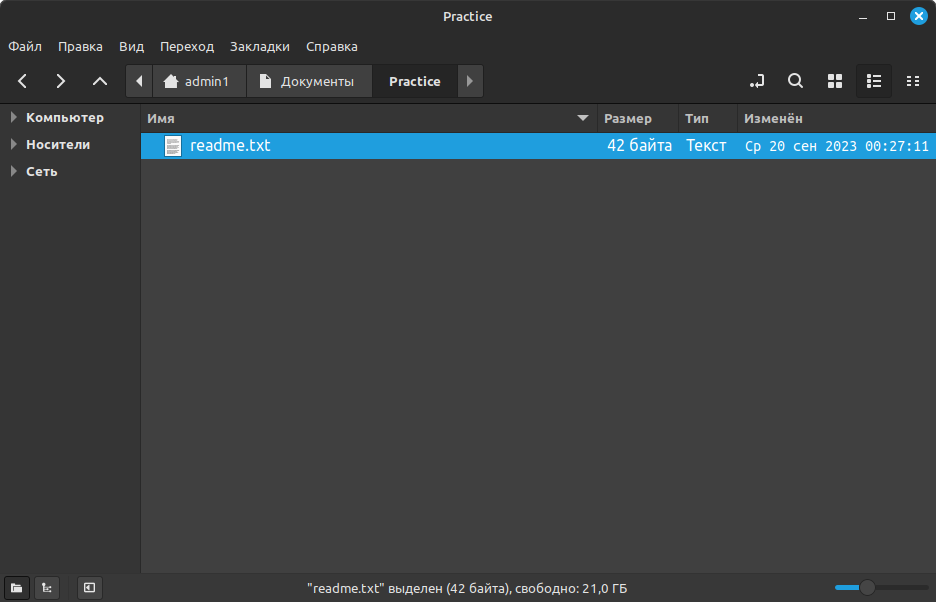


Рисунок 73.

1. git status - Показывает, что файлы удалены.

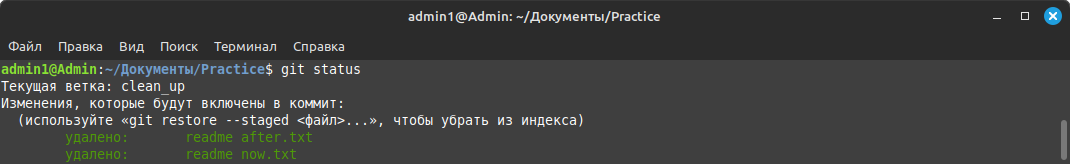


Рисунок 74.

1. git commit -m ‘Deleted folder and files’ - Фиксируем, что файлы удалены из репозитория.

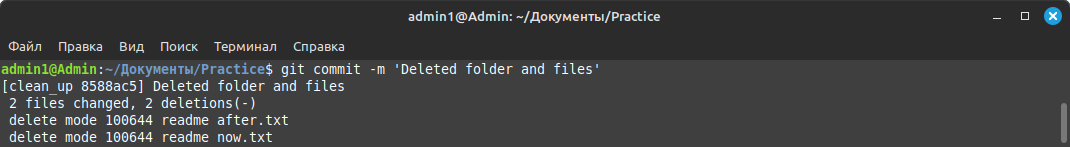


Рисунок 75.

1. git checkout master - Переключение на ветку master.

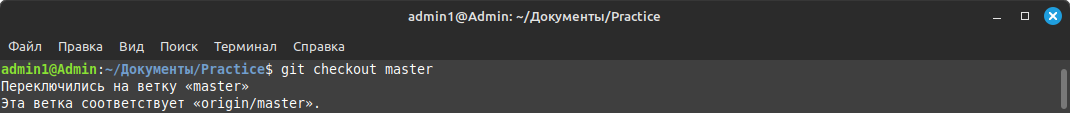


Рисунок 76.

Каталог.

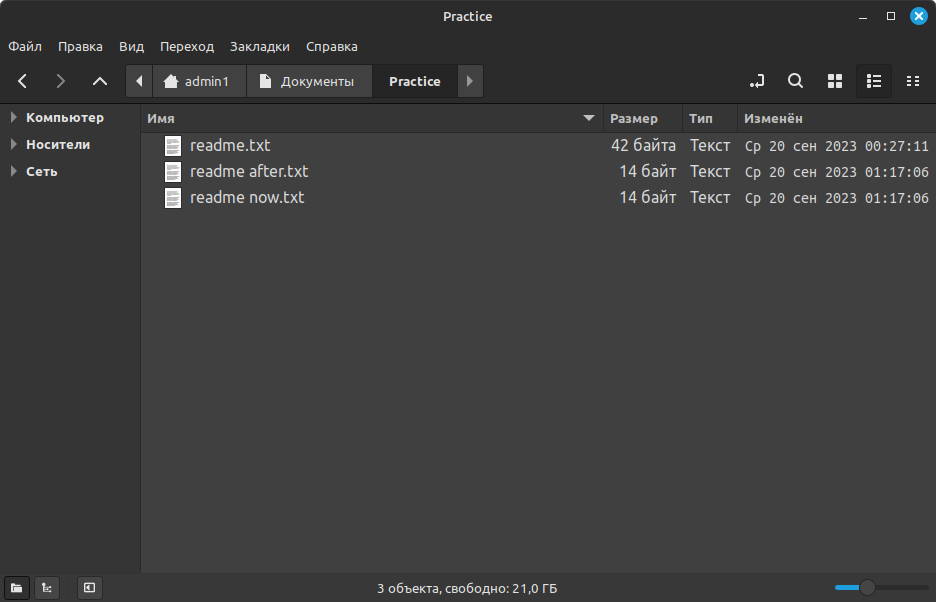


Рисунок 77.

1. git merge clean\_up - Слияние с веткой clean\_up.

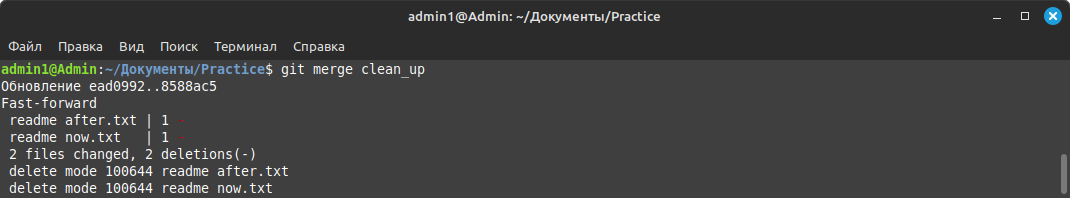


Рисунок 78.

Каталог.

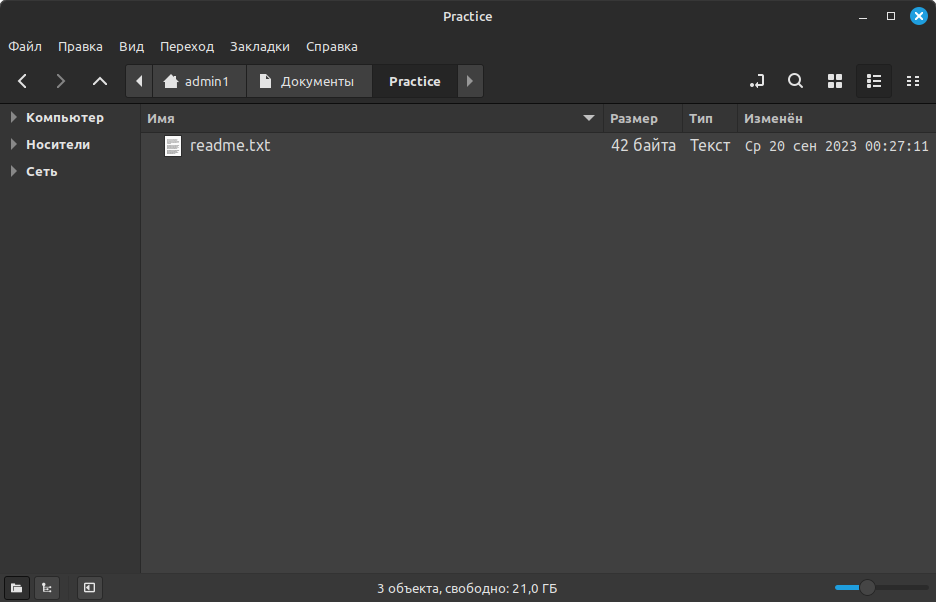


Рисунок 79.

1. git branch -d clean\_up - Удаляем ветку clean\_up.

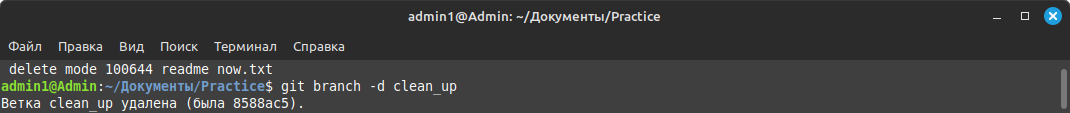


Рисунок 80.

Каталог.

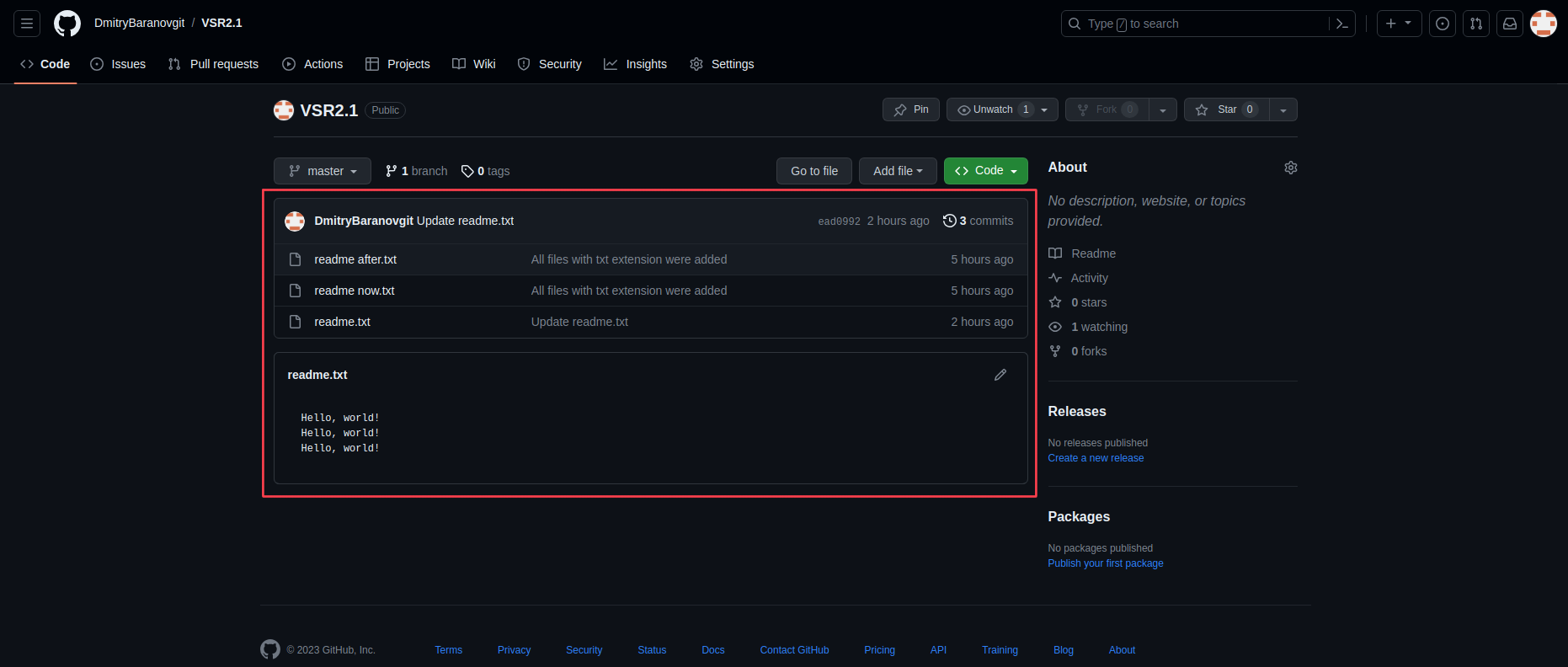


Рисунок 81.

1. git push - Выгружает последние изменения и показывает, что GitHub файлы были удалены.

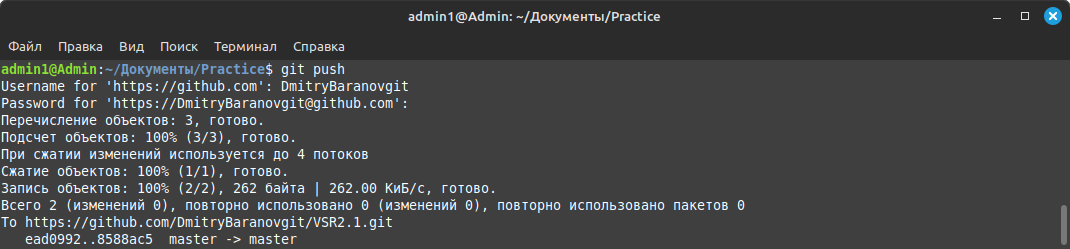


Рисунок 82.

GitHub. Последние изменения были добавлены.

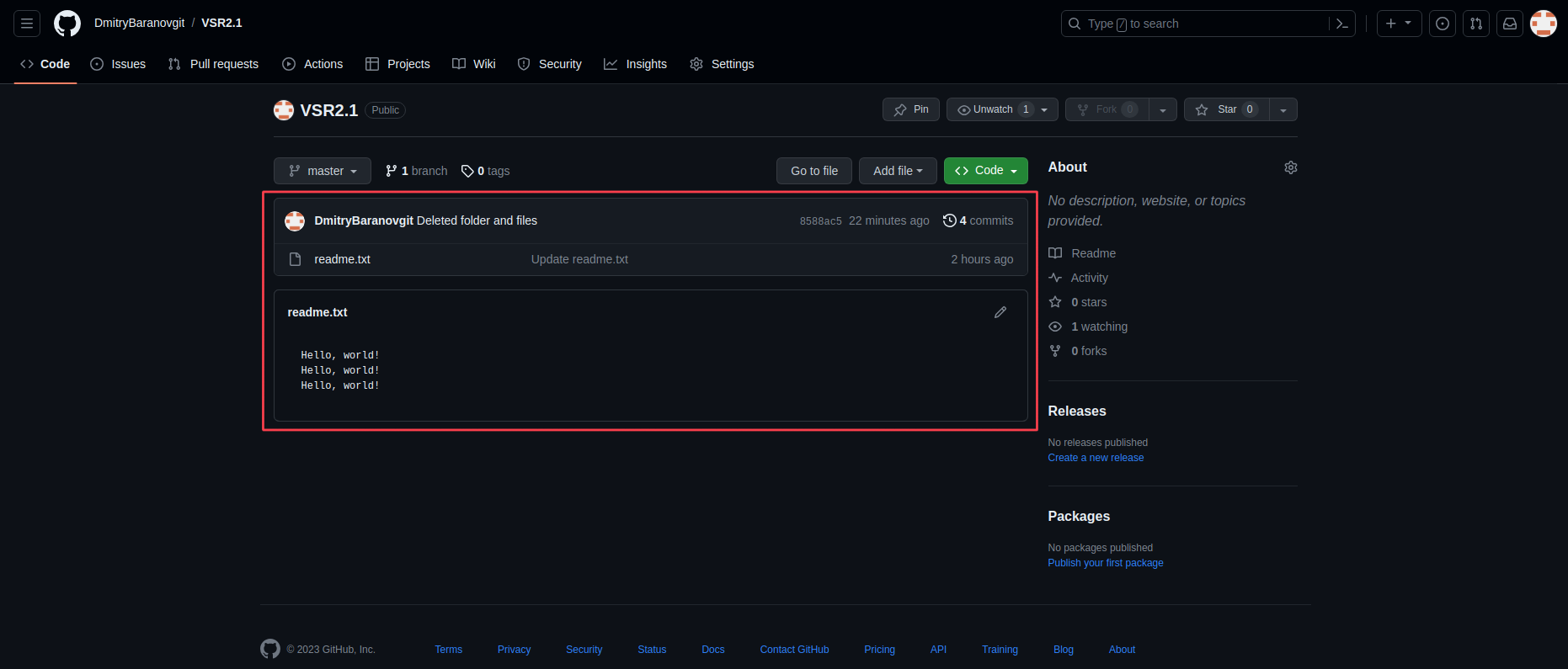


Рисунок 83.