|  |
| --- |
|  |
| 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 № 2** | |
| **по дисциплине** |  |
| **«**Технология разработки программных приложений**»** | |

|  |  |
| --- | --- |
| **Выполнил:**  Студент группыИКБО-10-23 | Лазаренко С. А. |
| **Проверил:**  Преподаватель к.э.н., доцент | Петросян Л. Э. |

Москва 2025 г.

**СОДЕРЖАНИЕ**

[ЧАСТЬ 1. ОСНОВНЫЕ КОМАНДЫ Git 3](#_bookmark0)

[ЧАСТЬ 2. СИСТЕМЫ УПРАВЛЕНИЯ РЕПОЗИТОРИЯМИ 10](#_bookmark1)

[ЧАСТЬ 3. РАБОТА С ВЕТВЛЕНИЕМ И ОФОРМЛЕНИЕ КОДА 16](#_bookmark2)

[ВЫВОД 22](#_bookmark3)

# ЧАСТЬ 1. ОСНОВНЫЕ КОМАНДЫ Git

1. Установка и настройка клиент git на рабочей станции Сначала зайдем на офицальный сайт Git:

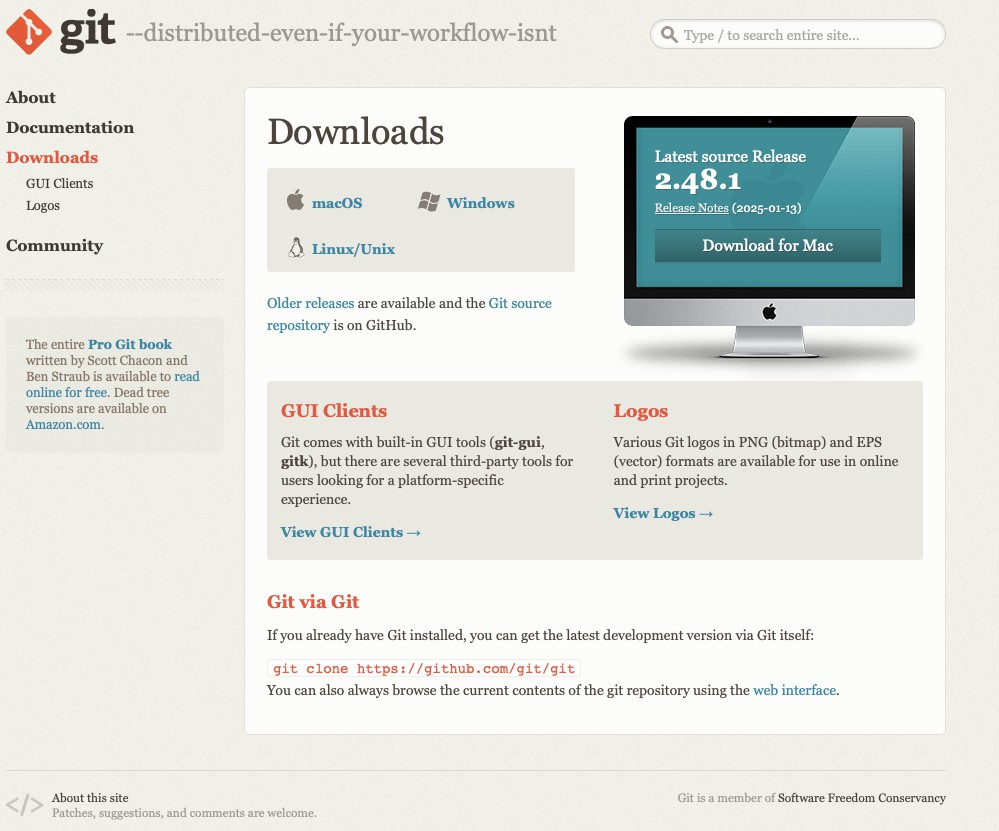


Рисунок 1.1 – Главная страница сайта Git

Далее выберем нужную версию под операционную систему (macOS) и скачаем установочный файл. После этого запустим его и , следуя инструкциям на экране, установим Git.

После завершения установки откроем терминал и проверим, что система появилась на устройстве:

Рисунок 1.2 – Проверка версии Git в консоли

Откроем терминал и выполним следующие команды в консоле:



Рисунок 1.3 – Настройка имени пользователя и электронной почты в Git Выполним следующую команду и для проверки верности всех

введенных команд, напишем *git config --list*:



Рисунок 1.4 – Отключение кавычек для путей файлов в Git

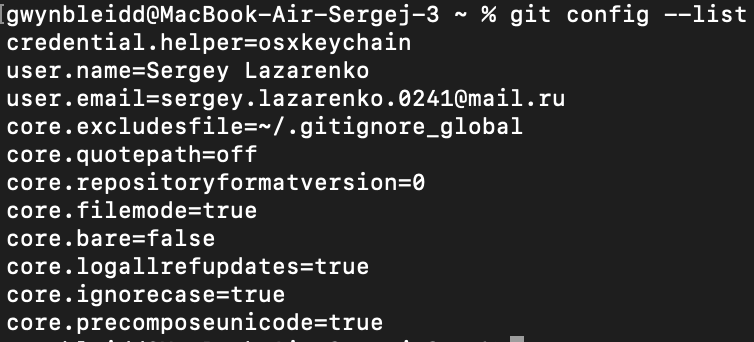


Рисунок 1.5 – Базовая настройка Git для macOS

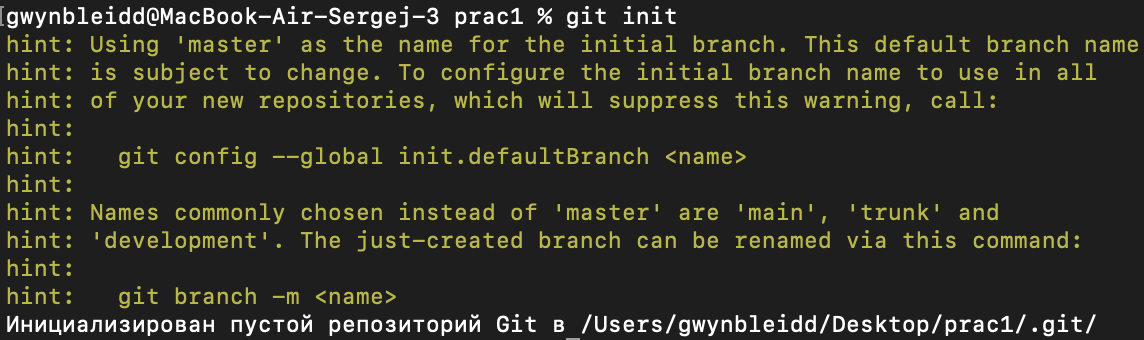
1. Создание локального репозитория и добавления в него файлов Cначала иницилизируем локальный репозиторий:

Рисунок 1.6 – Иницилизация пустого репозитория Git Создадим несколько файлов, затем добавим их в репозиторий:

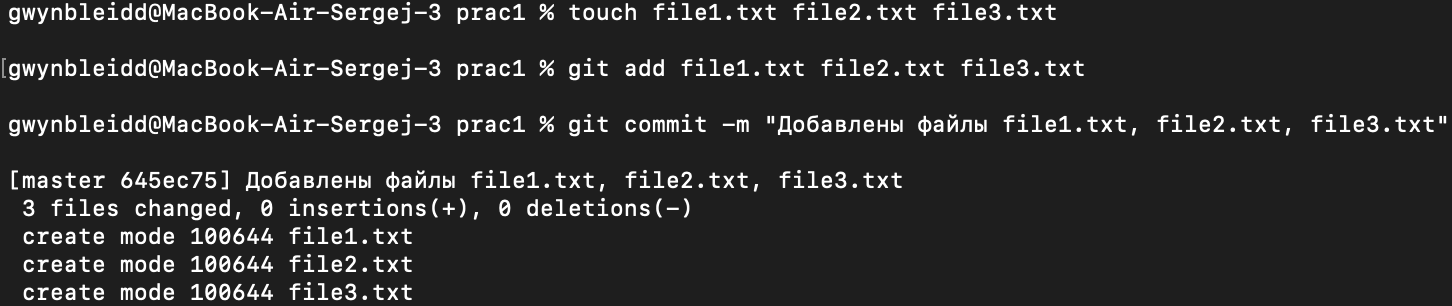


Рисунок 1.7 – Создание и добавление файлов в локальный репозиторий

1. Внесение изменений в один из файлов

Откроем файл с помощью текстового редактора nano для внесения изменений в один из файлов:



Рисунок 1.8 – Внесение изменений в файле

1. Индексация изменений и проверка состояния

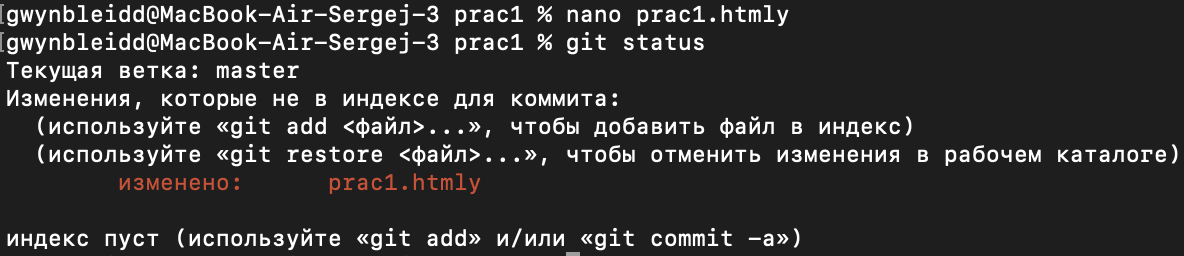
Внесем изменений в созданный ранее файл и проверим его статус с помощью команды *git status*:

Рисунок 1.9 – Индексация изменений

1. Коммит того, что было проиндексировано. Добавления комментария Убедимся, что файл добавлен в индекс, затем добавим коммит с

комметарием:

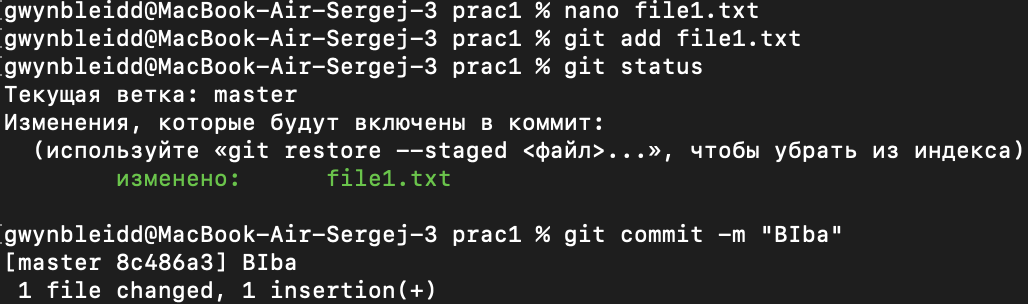


Рисунок 1.10 – Коммит изменений с комментарием в Git’

1. Коммиты нескольких изменений

Откроем файл с помощью текстового редактора nano, чтобы внести изменения в файл prac1.html. После того как внесены изменения, выполним команды для добавления и коммита изменений в Git для фиксации изменений:

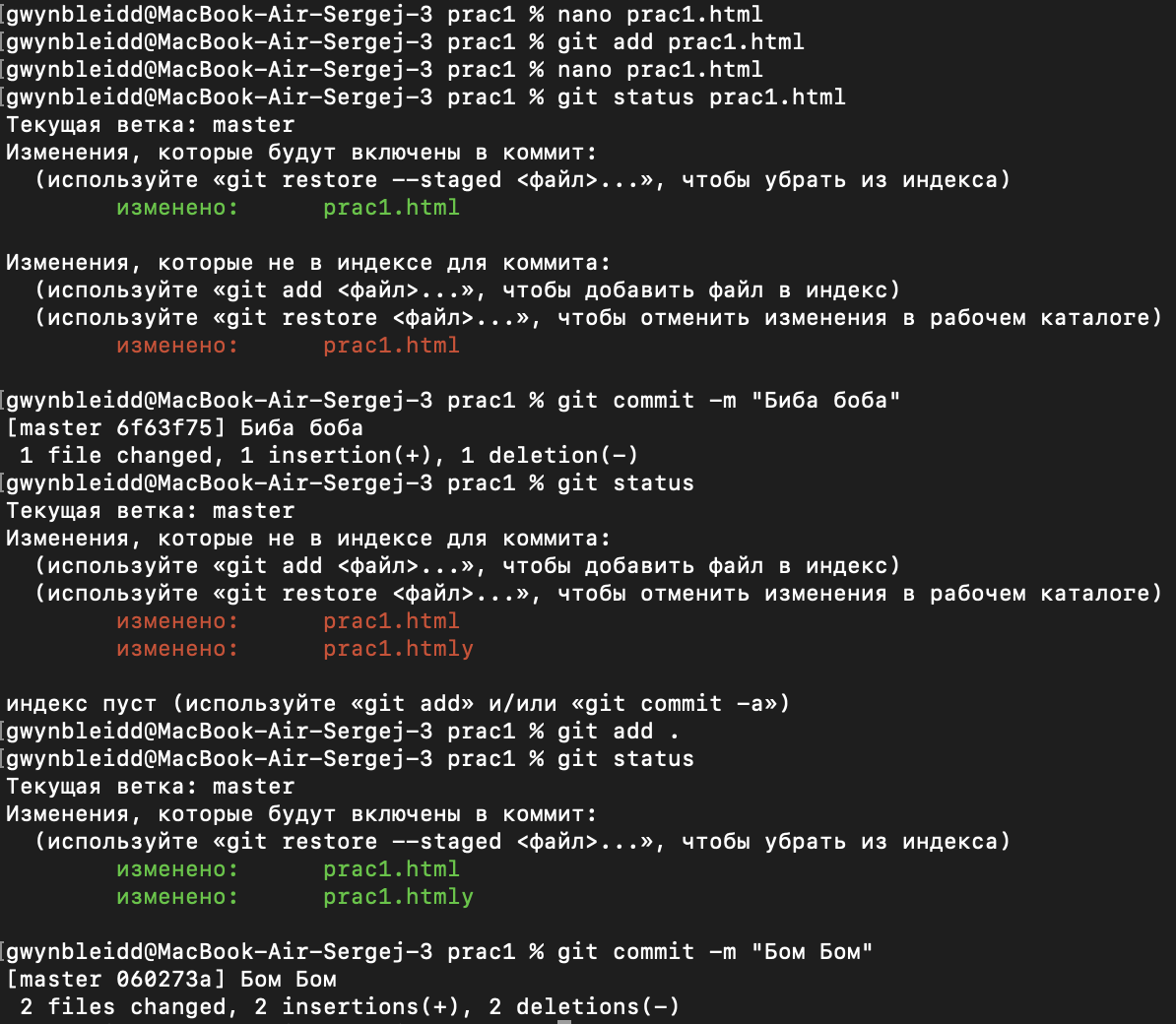


Рисунок 1.11 – Коммиты нескольких изменений

1. Просмотр истории коммитов

Для просмотра истории коммитов воспользуемся командой *git log*:



Рисунок 1.12 – Просмотр истории коммитов

Команда *git log* позволяет контролировать формат выводимой информации:

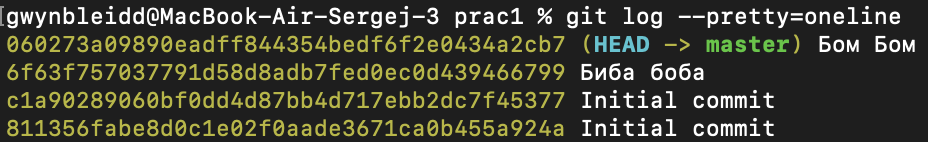


Рисунок 1.13 – Изменение формата вывода коммитов

1. Получение старых версий рабочего каталога

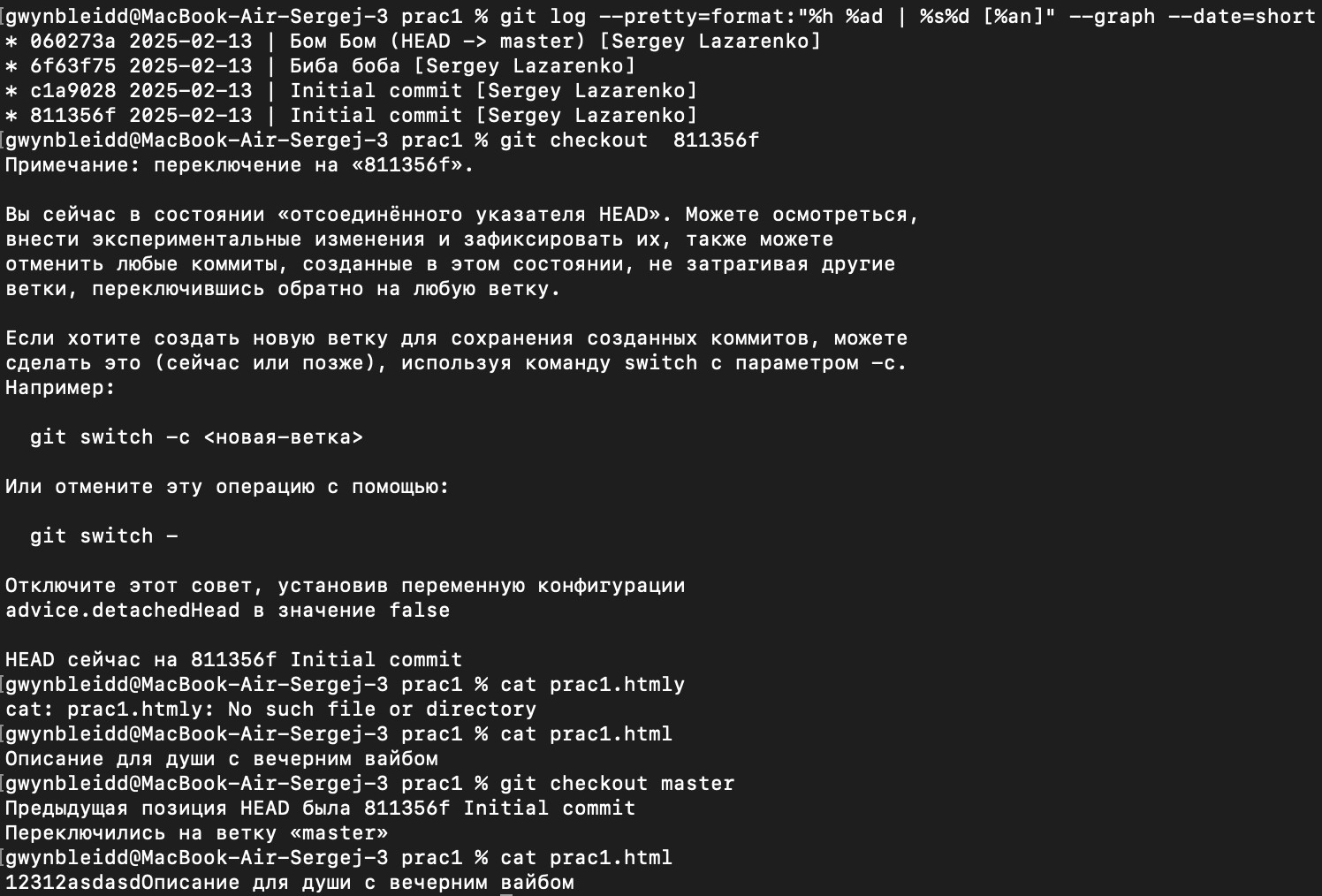
Для получения старой версии необходимо узнать для начала хэши, для этого воспользуемся командой из предыдущего шага:

Рисунок 1.14 – Получение старых версий

1. Изучение создания тегов для коммитов для использования в будущем Создадим легкий тег – ссылка на коммит, который не содержит дополнительной информации, просмотрим список всех тегов, создадим тег для конкретного коммита, а не для последнего, указав его хеш:

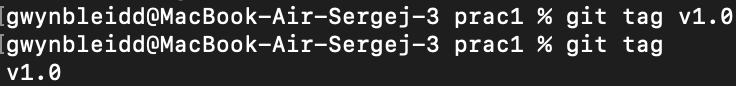


Рисунок 1.15 – Создание и просмотр тега

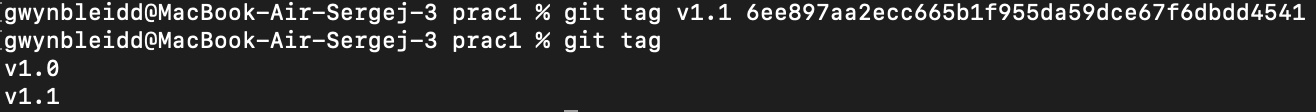


Рисунок 1.16 – Создание тега для конкретного коммита с указанием хеша

1. Отмена локальных изменений (до индексации и после индексации)

Проверим статус репозитория с помощью *git status*, увидим, что файл был изменен, но не добавлен в индексации для коммита. Затем используем команду *git checkout prac1.html*, чтобы отменить изменения и вернуть файл в состояние последнего коммита, после чего проверим содержимое с помощью *cat*:

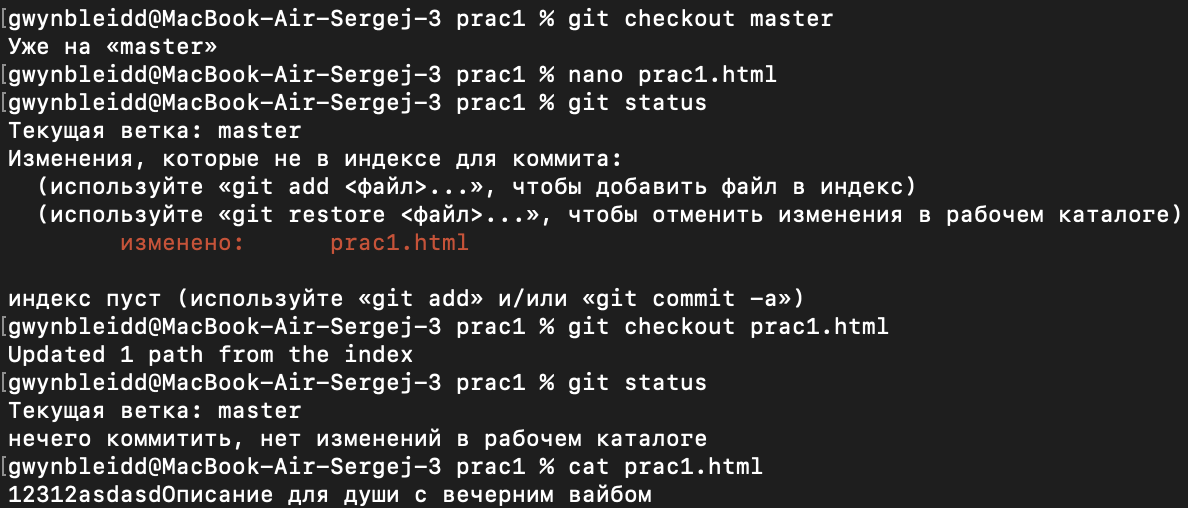


Рисунок 1.17 – Отмена локальных изменений до индексации

Внесем изменения в файл и добавим его в индекс с помощью команды *git add*. Затем, командой *git reset HEAD*, отменим добавление файла в индекс, вернув его состояние “неидексированных изменений”. После используем команду *git checkout prac1.html*, чтобы отменить все изменения в файле и восстановить последную сохраненную версию:

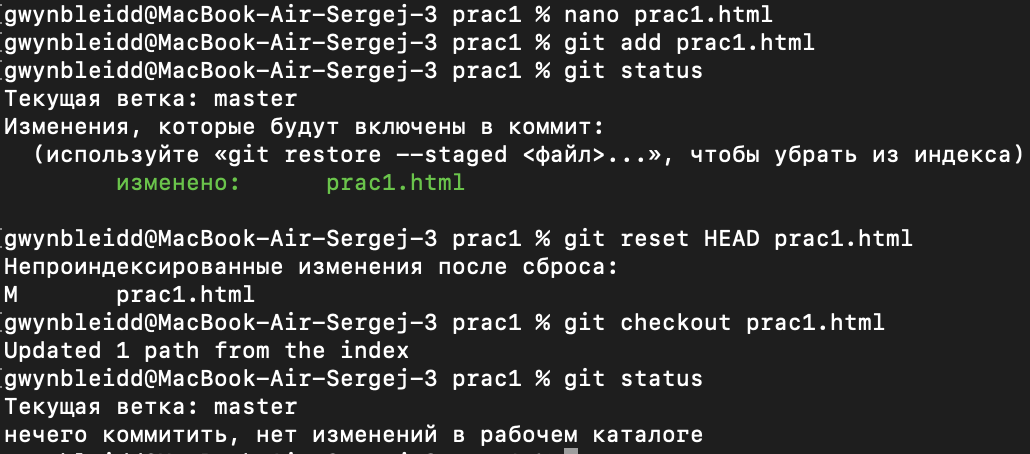


Рисунок 1.18 – Отмена локальных изменений после индексации, до коммита

1. Отмена коммита в локальном репозитории

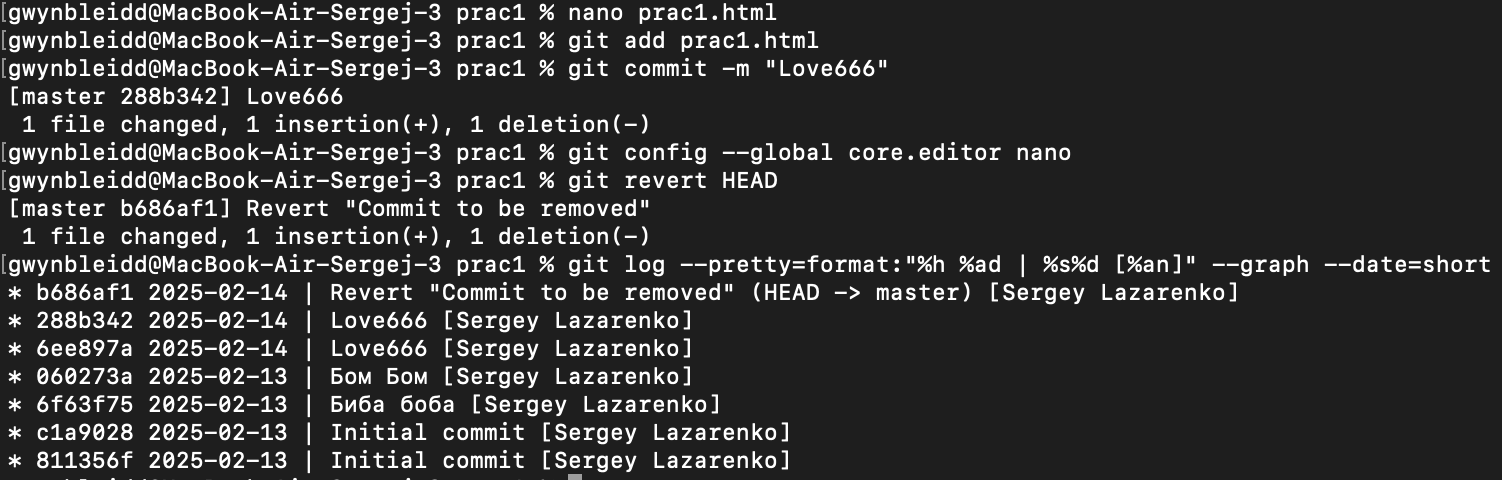
Внесем изменения в файл, добавим его в индекс, а затем зафиксируем изменения с помощью команды *git commit*. После этого настроим редактор по умолчанию для Git на nano. Затем отменим последний коммит, создав новый коммит, который отменяет изменения старого. Проверим историю коммитов, убедившись в присутствии коммита отмены:

Рисунок 1.19 – Отмена коммита

# ЧАСТЬ 2. СИСТЕМЫ УПРАВЛЕНИЯ РЕПОЗИТОРИЯМИ

1. Создание аккаунта на GitHub

Перейдем на сайт GitHub, пройдем регистрацию, после которой будет небольшой опрос.

После прохождения всех этапов на сайте, на указанный при регистрации ящик придет письмо от GitHub. Отроем его и подтвердим свой почтовый адреc. Теперь у нас есть профиль на GitHub:

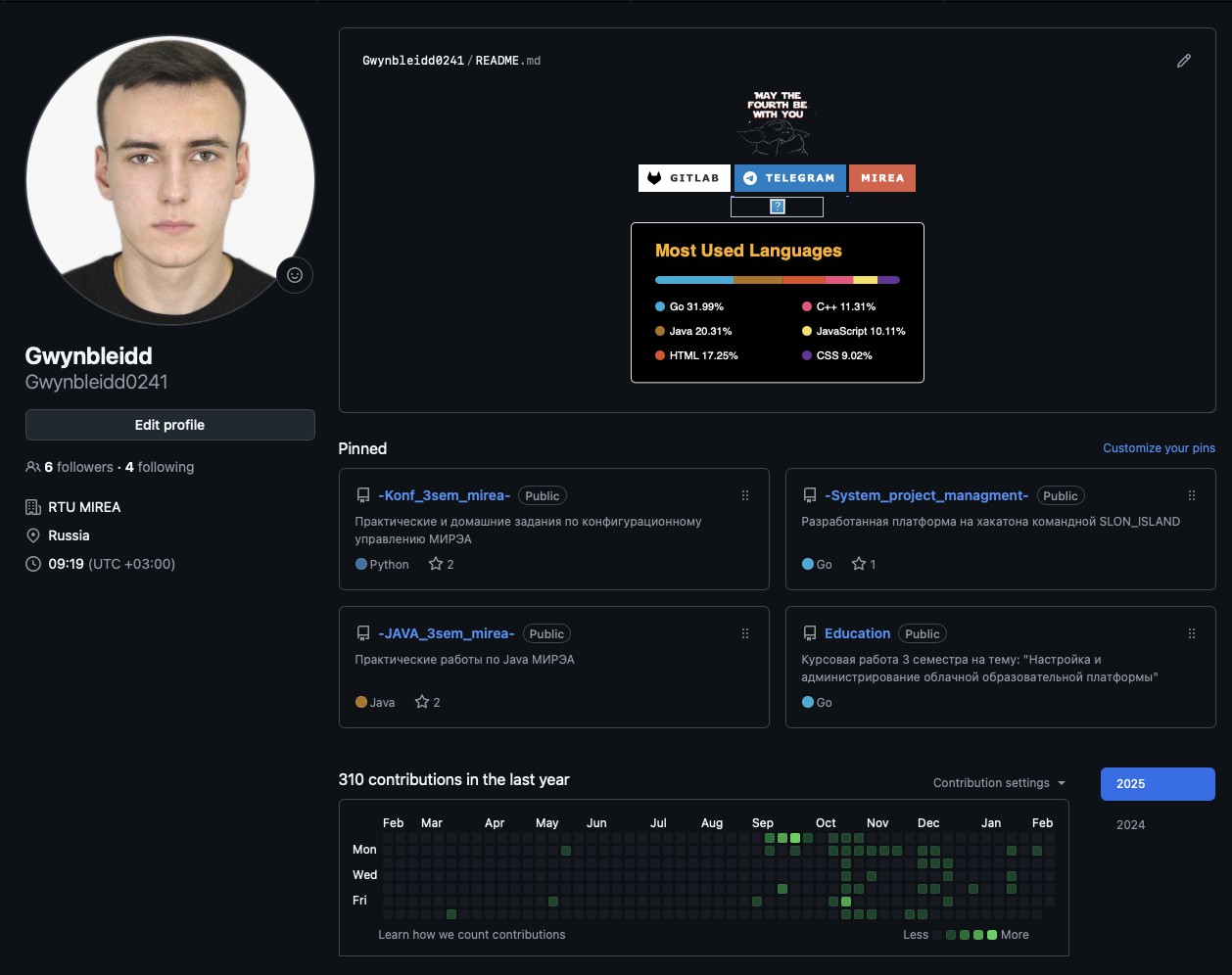


Рисунок 2.1 – Созданный профиль на GitHub

1. Создание SSH-ключа для авторизации

Сначала проверим, есть ли уже на компьютере ключ. По умолчанию SSH-ключи храняться в каталоге *~/.ssh*, поэтому нужно проверить содержимое этого каталога:



Рисунок 2.2 – Проверка каталога, содержащего SSH-ключи

Открываем консоль и вводим команду:



Рисунок 2.3 – Создание SSH-ключа

Далее добавим ключ в ssh-agent (сгенерированный или уже существующий) с помощью *ssh-add ~/.ssh/id\_ed25519.pub*. В результате получим:



Рисунок 2.4 – Настройка SSH-агента и проверка его процесса

Чтобы связать локальный и удаленный репозитории, выведем в консоль содержимое файла с помощью *cat*. В итоге получим вот такой ключ (с методички):



Рисунок 2.5 – Полученный SSH-ключ

Скопируем его из консоли и перейдем на страницу для работы с ключами в нашем профиле на GitHub. Выбирем кнопку “New SSH key”, откроется окно с вводом данных, в поле “key” вставим скопированный ключ, в “Title” вводим любое имя ключа и нажимаем “Add SSH key”:



Рисунок 2.6 – Добавленный SSH-ключ в GitHub

1. Создание нового репозитория для своего проекта

Создадим новую директорию, затем создадим несколько файлов в этой директории с добавлением содержимого в файлы. После иницилизируем новый Git-репозиторий:

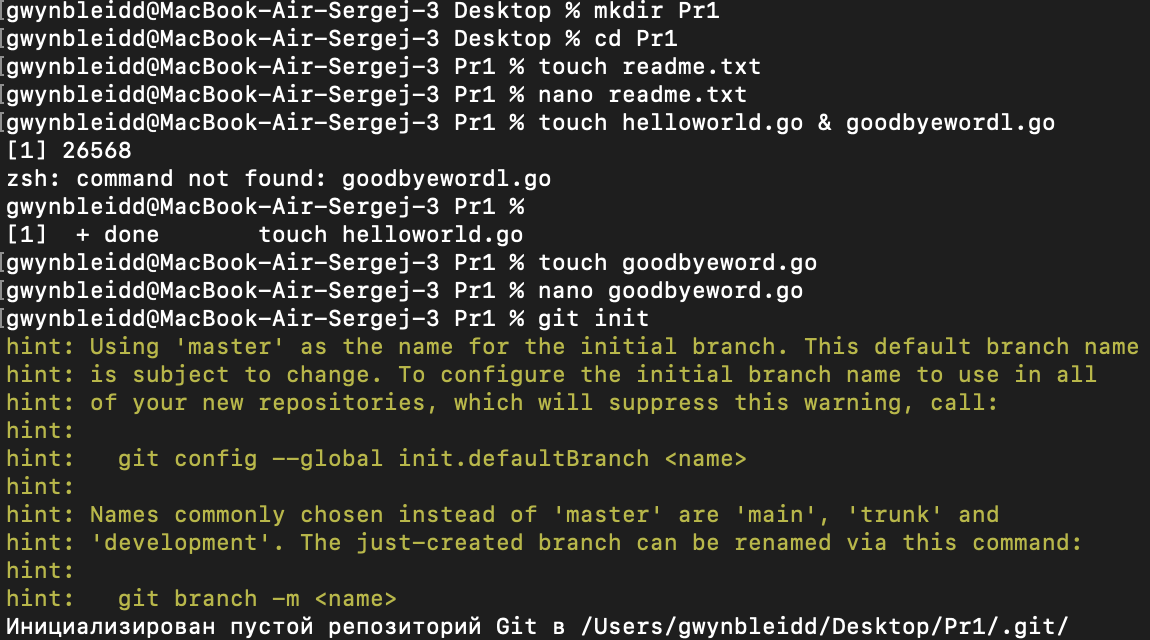


Рисунок 2.7 – Создание репозитория с несколькими файлами

1. Связывание локального и удаленного репозитория

Заходим на GitHub, на свою страницу, выбираем вкладку репозитория. На вкладке с репозиториями нажимаем на зеленую кнопку “New”. Называем наш репозиторий и выбираем приватность репозитория. После проделанных шагов, мы создали пустой репозиторий на GitHub, осталось только связать его с локальным:

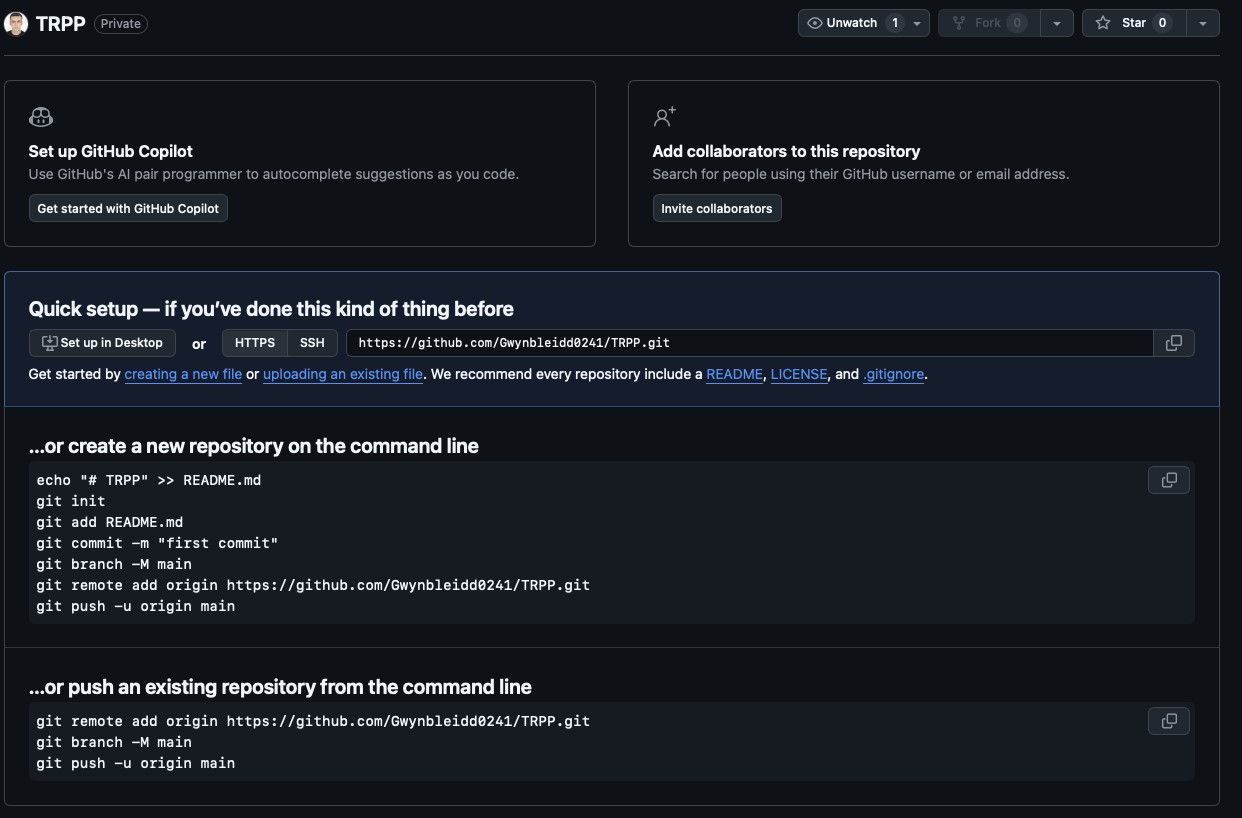


Рисунок 2.8 – Созданный репозиторий на GitHub

Чтобы связать репозитории друг с другом введем следующую команду в консоль:



Рисунок 2.9 – Связывание локального и удаленного репозитория

1. Создание веток и переключение между ними

Для создания новой ветки и переключения на нее воспользуемся командой *git checkout -b <имя ветки>*. Сделаем изменения в файле и проиндексируем их, затем коммитим:

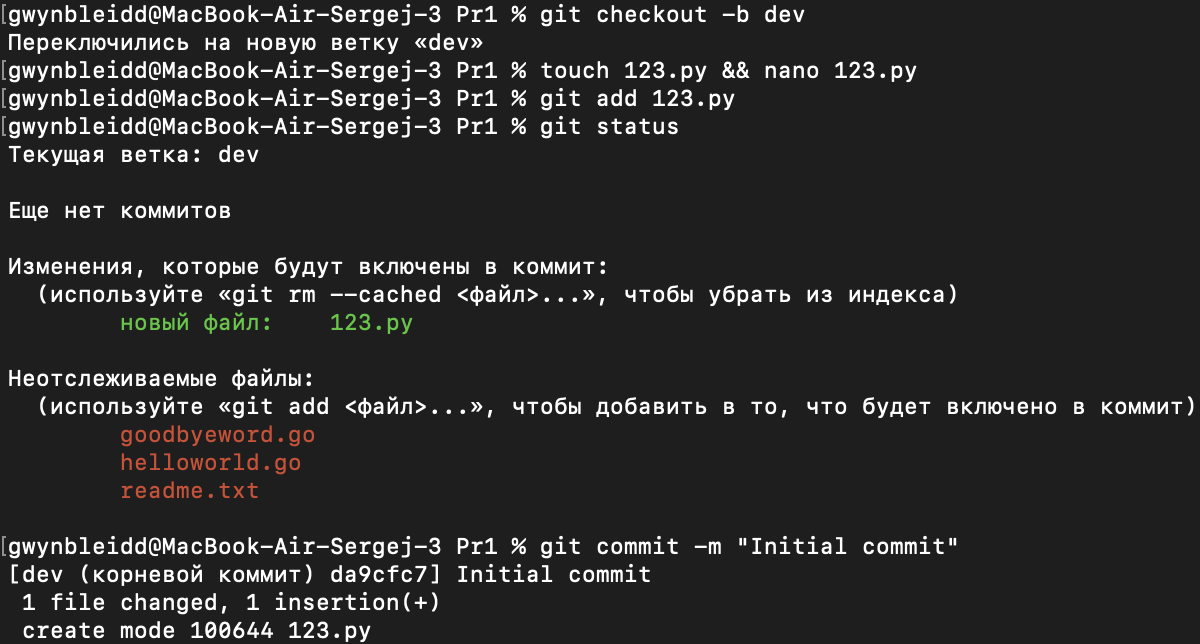


Рисунок 2.10 – Работа в новой ветке

1. Слияние веток

Воспользуемся командой *git merge* для слияния веток dev и master:

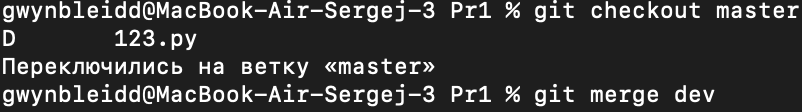


Рисунок 2.11 – Слияние веток

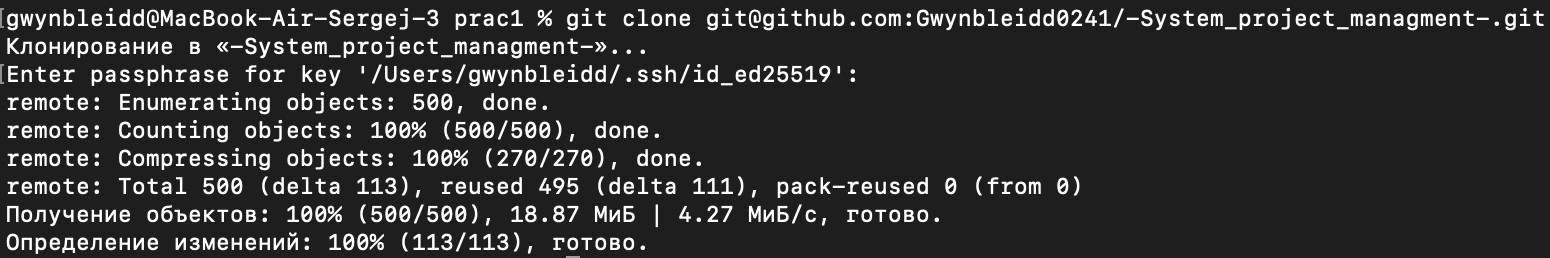
1. Выполнение цепочки действий в репозитории согласно варианту 6 Клонируем непустой удаленный репозиторий на локальную машину:

Рисунок 2.12 – Клонирование непустого удаленного репозитория

Создадим новую ветку и выведим список всех веток:

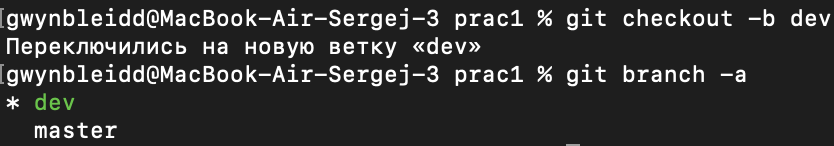
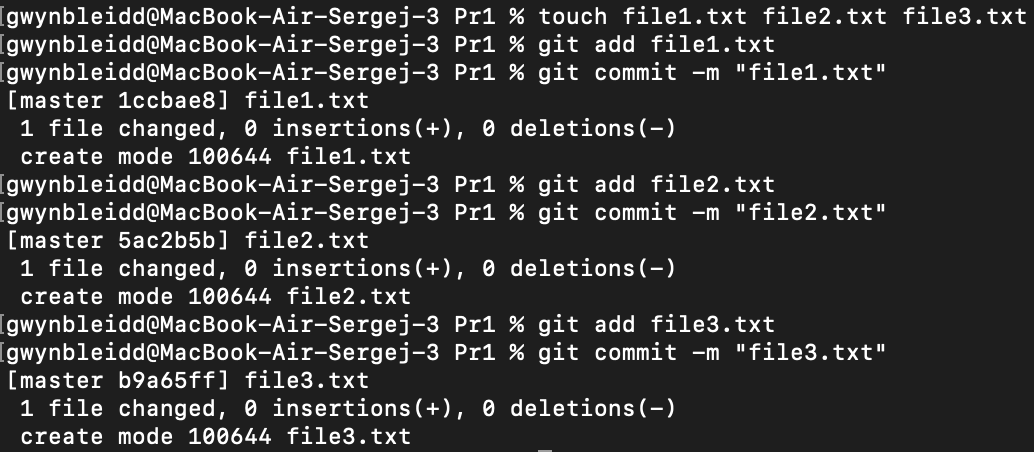


Рисунок 2.13 – Создание новой ветки и вывод списка всех веток Проведем 3 коммита в новой ветке в разные файлы:



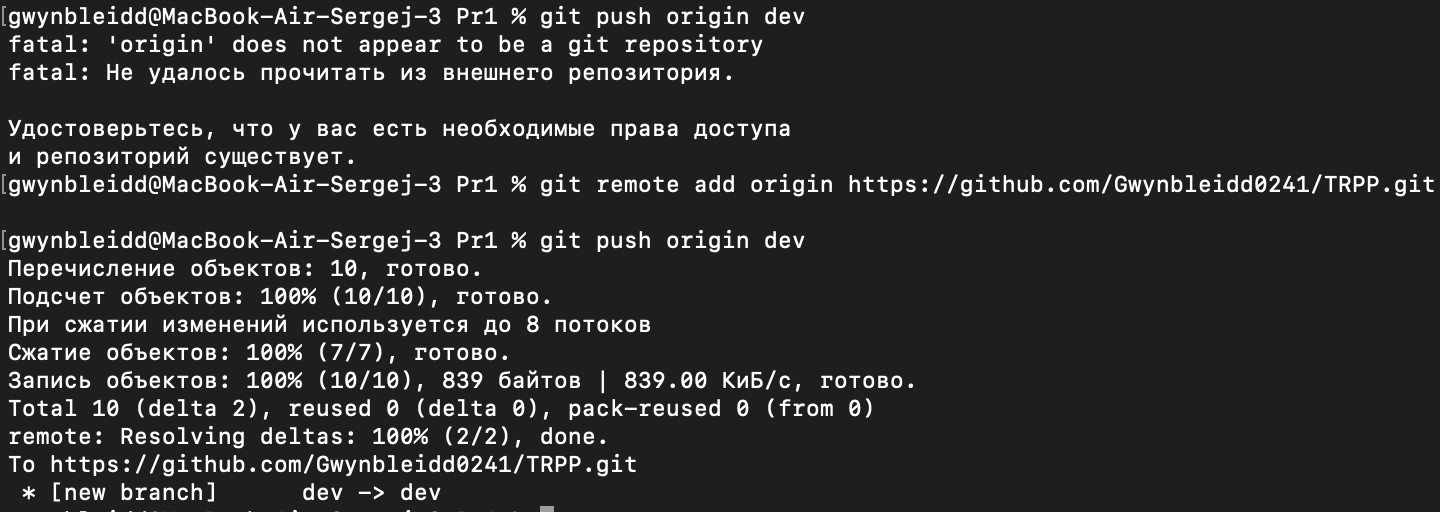
Рисунок 2.14 – Произведение 3 коммитов в новой ветке Выгрузим изменения в удаленный репозиторий:

Рисунок 2.15 – Выгрузка изменений в удаленный репозиторий

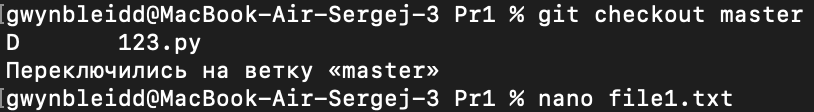
Переключимся на ветку master, затем с помощью nano изменим файл, не производя коммит:

Рисунок 2.16 – Изменения в файле в ветке master

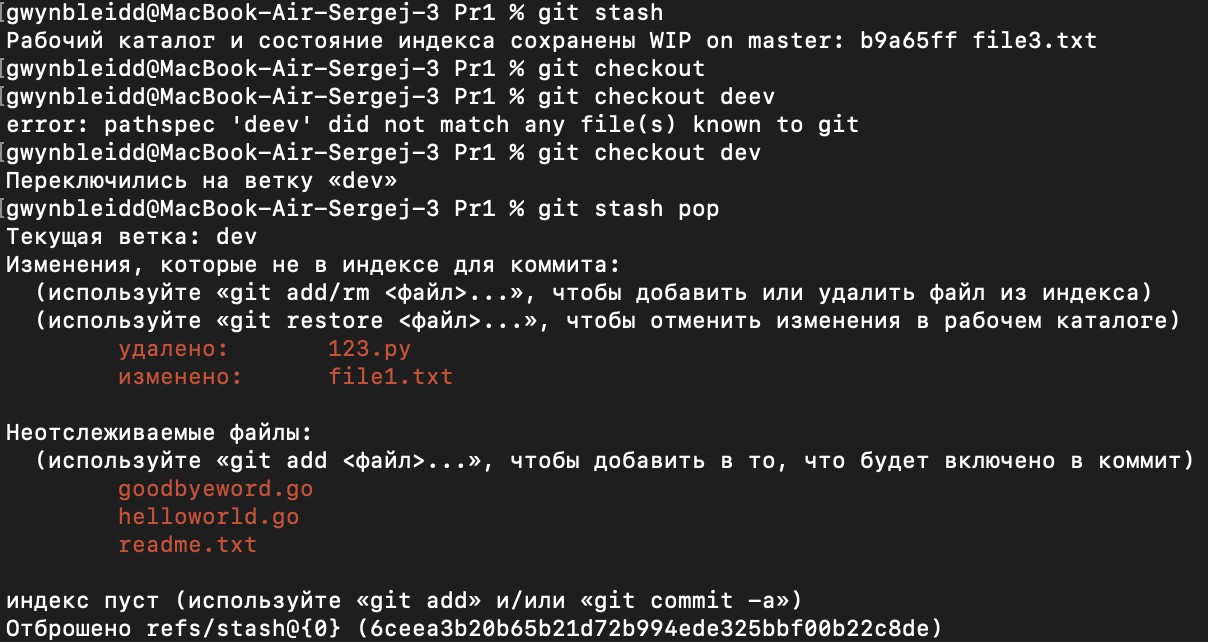
Выполним команду *git stash*, которая сохранит изменения в хранилище и вернет рабочую директорию до состояния последнего коммита, затем переключимся на ветку dev и выполним *git stash pop*, которая восстановит изменения в текущей ветке и удалит их из хранилища:

Рисунок 2.17 – Откат изменений в новой ветке

Выведим в консоли различия между веткой master и dev, затем сольем ветки при помощи *git merge*:

Рисунок 2.18 – Различия в ветках



Рисунок 2.19 – Слияние веток

# ЧАСТЬ 3. РАБОТА С ВЕТВЛЕНИЕМ И ОФОРМЛЕНИЕ КОДА

1. Получение форка репозитория в соответствии с вариантом 7

Заходим на GitHub, переходим по ссылке в репозиторий, который собираемся форкать, в верхнем правом углу нажимает кнопку “Fork”:

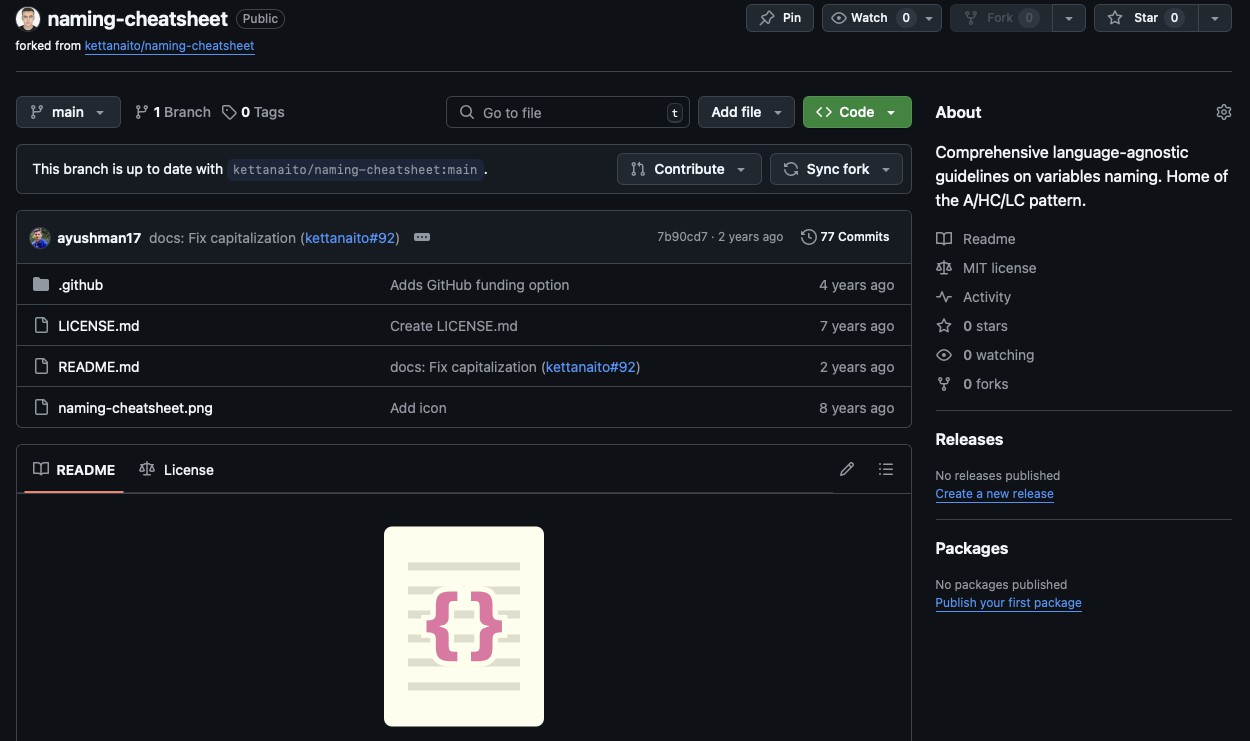


Рисунок 3.1 – Форк репозитория

1. Клонирование репозитория на локальную машину

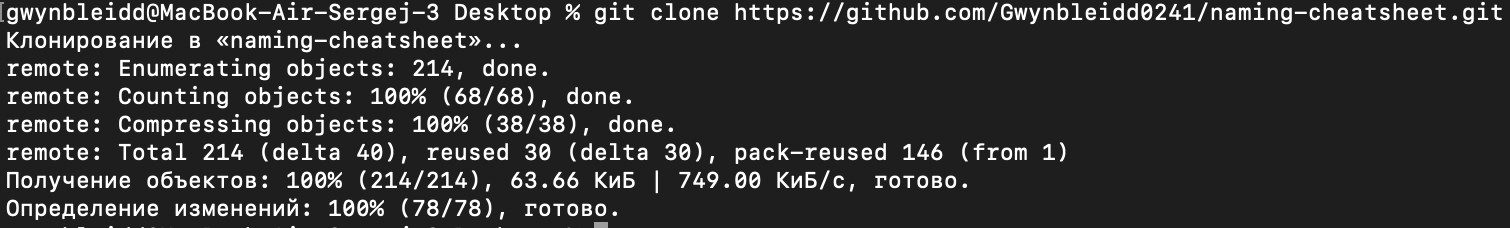


Рисунок 3.2 – Клонирование репозитория на локальную машину

1. Создание двух веток в main’e

Проверим, что мы находимся на нужной ветке и создадим две ветки от последнего коммита в ветке main:

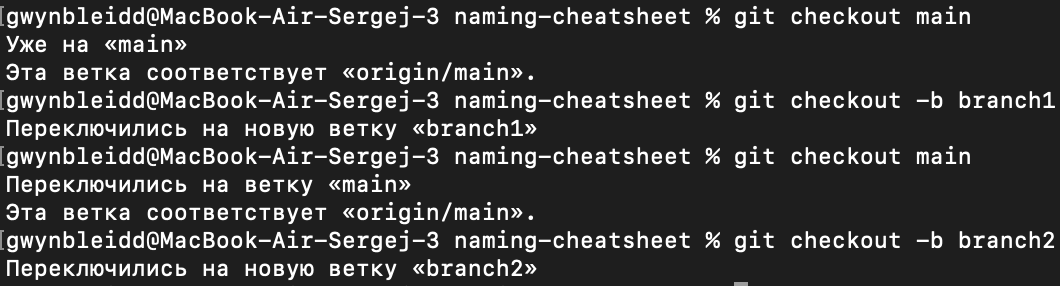


Рисунок 3.3 – Создание веток от последнего коммита в main

1. Проведение по 3 коммита в каждую из веток

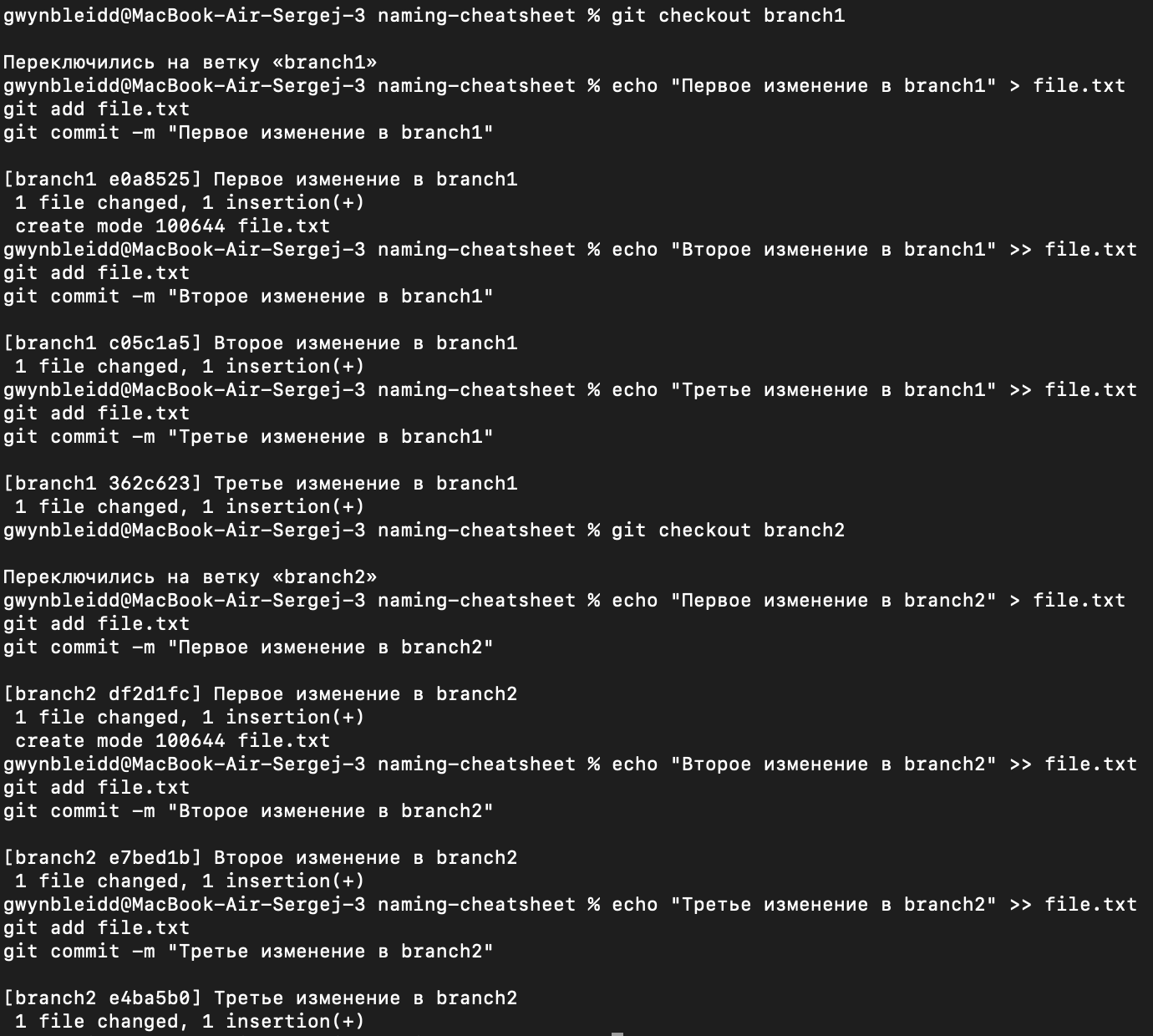
Переключимся на branch1, проведем по 3 коммита, затем также на ветку branch2:

Рисунок 3.4 – Проведение по 3 коммита в каждую из веток

1. Слияние ветки branch1 в ветку branch2

Выполним слияние ветки branch1 в ветку branch2, разрешив конфликты при этом:

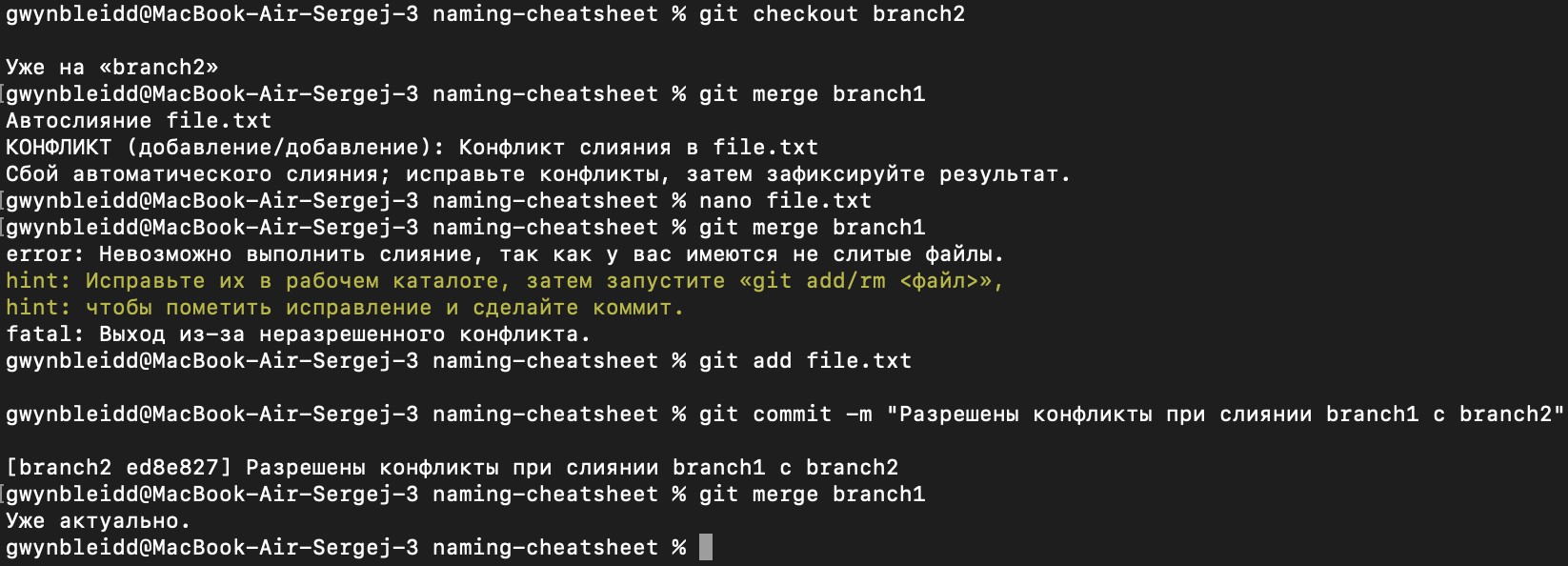


Рисунок 3.5 – Слияние веток

1. Выгрузка всех изменений во всех ветках в удаленный репозиторий

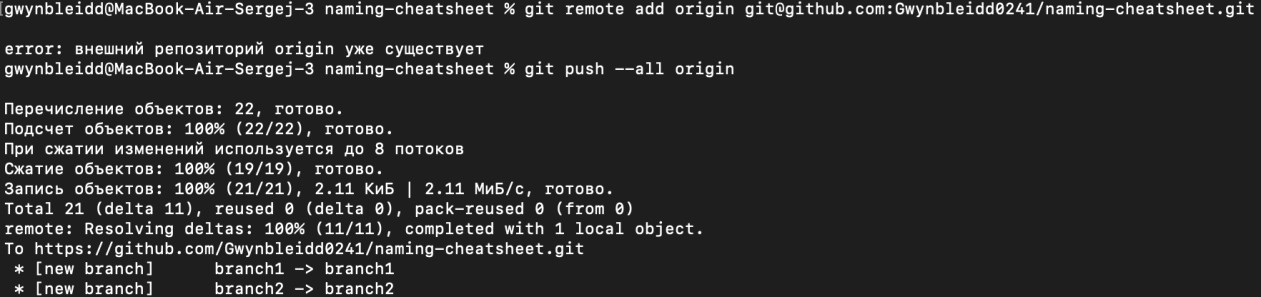


Рисунок 3.6 – Выгрузка всех изменений

1. Проведение еще 3 коммитов в ветке branch1

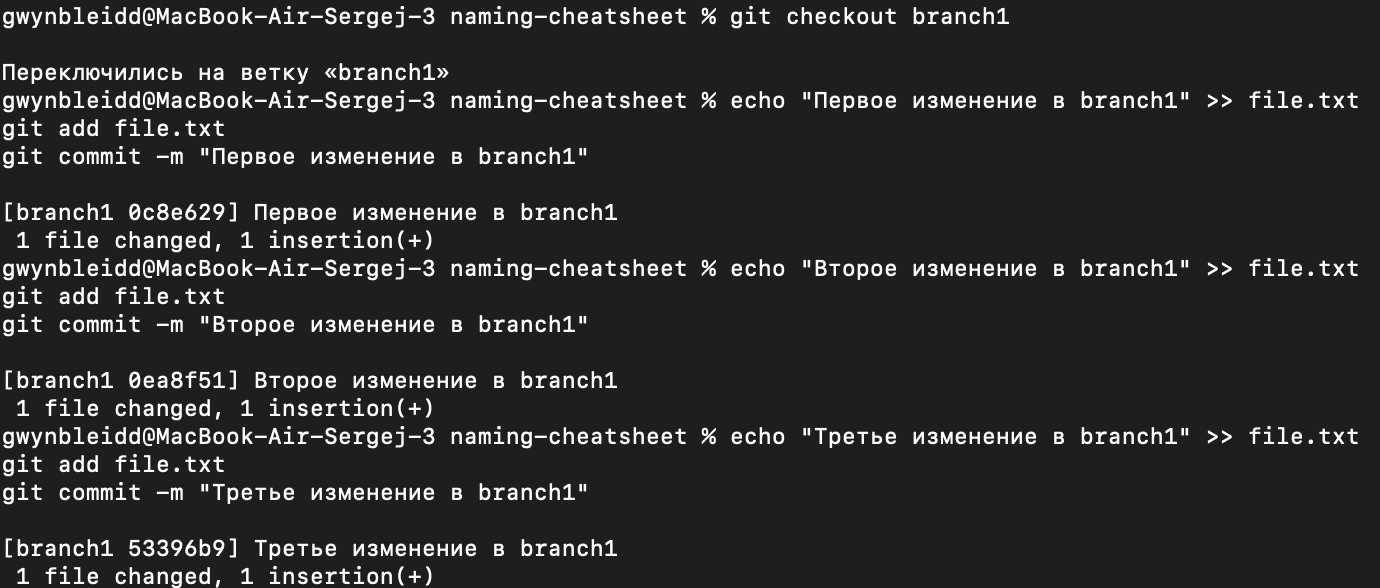


Рисунок 3.7 – Проведение 3 коммитов в ветке branch1

1. Клонирование репозитория в другую директорию

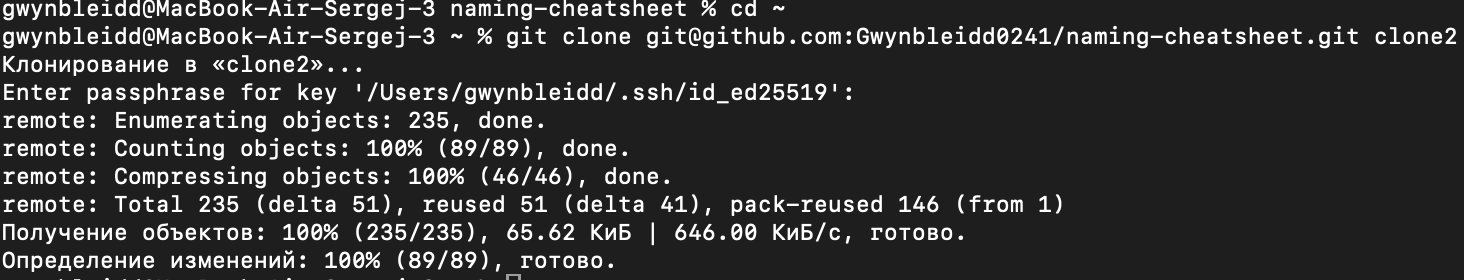


Рисунок 3.8 – Клонирование репозитория в другую директорию

1. Проведение 3 коммитов в ветке branch1 в новом клоне репозитория

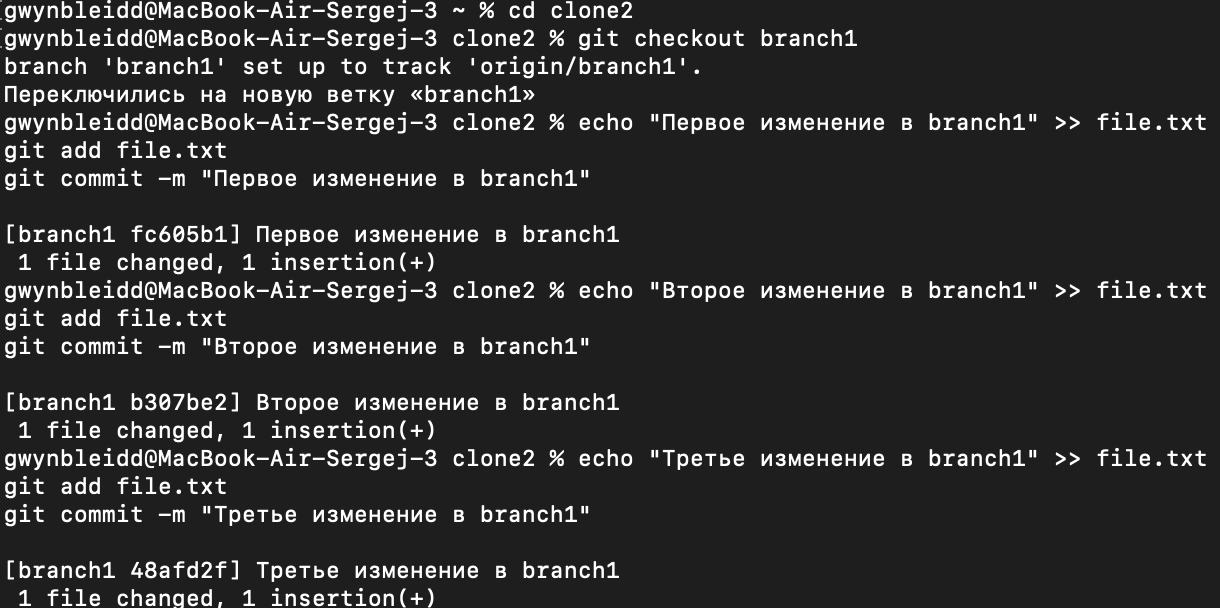


Рисунок 3.9 – Проведение 3 коммитов в новом клоне репозитортия

1. Выгрузить все изменения из нового репозитория в удаленный

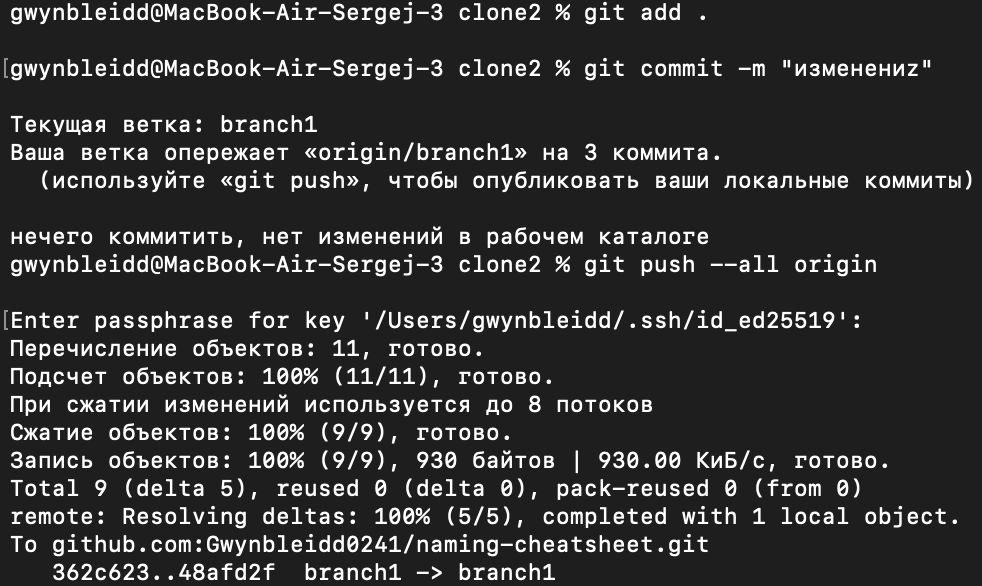


Рисунок 3.10 – Выгрузка всех изменений в удаленный репозиторий

1. Возвращение в старый клон с репозиторием, выгрузка изменений

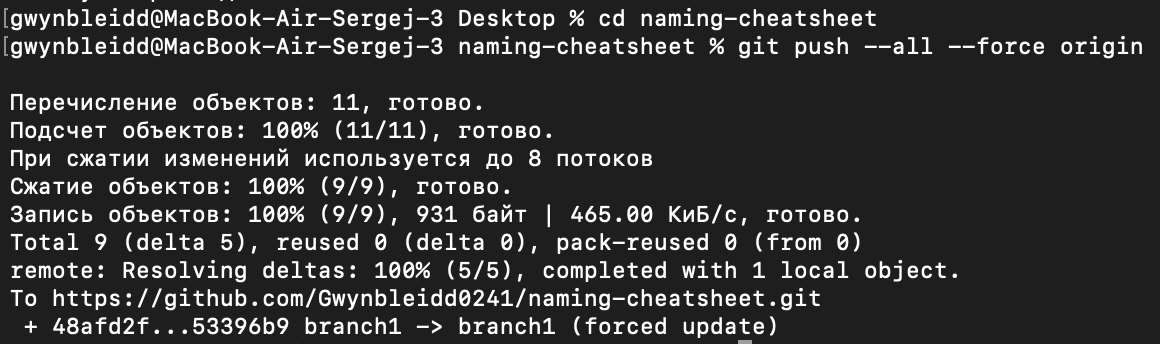


Рисунок 3.11 – Возвращение в старый клон и выгрузка изменений

1. Получение всех изменений в новом репозитории

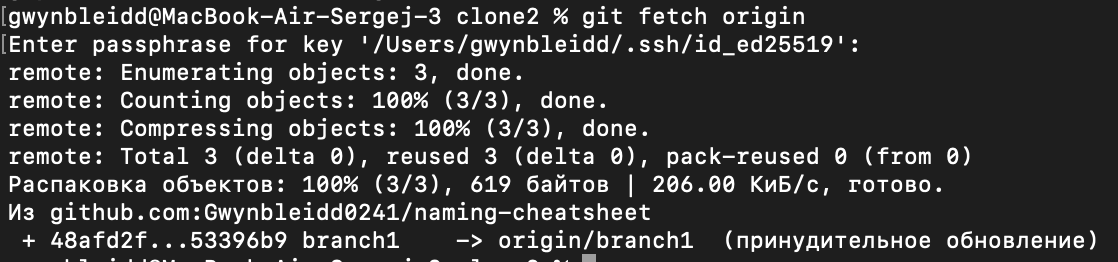


Рисунок 3.12 – Получение измений

**Контрольные вопросы**

1. Как отменить слияние веток, если произошел конфликт?

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

1. 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.

1. Что делает команда git status?

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

1. 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.

1. Что делает команда git show?

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

1. 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

Мы освоили основные команды Git для управления локальным репозиторием: создание, изменение, индексирование, коммиты и отмена изменений. Также изучили работу с тегами для метки коммитов и использование систем управления репозиториями, таких как GitHub, в том числе создание, клонирование, связывание и выгрузка репозиториев по SSH- ключу. Научились создавать, переключаться, сливать и удалять ветки, а также разрешать конфликты при слиянии. Познакомились с работой с ветками в удаленном репозитории и использованием опции --force для принудительной выгрузки изменений. Кроме того, мы провели форк репозитория и склонировали его для дальнейшей работы. Полученный опыт работы с Git и системами управления репозиториями будет полезен в будущих проектах.