|  |
| --- |
|  |
| |  |  |  | | --- | --- | --- | |  |  |  | | 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»  по теме «Системы контроля версий»   |  |  | | --- | --- | | **Выполнил:**  Студент группыИКБО-20-22 | Шумахер М. Е. | | **Проверил:** | преподаватель Мельников Д. А. |   МОСКВА 2024 г. |

**СОДЕРЖАНИЕ**

[Задание (Вариант 10) 3](#_Toc159231525)

[Выполнение заданий 5](#_Toc159231526)

[Часть 1. Основные команды Git 5](#_Toc159231527)

[Часть 2. Системы управления репозиториями 10](#_Toc159231528)

[Часть 3. Ветвление и оформление кода 10](#_Toc159231529)

[Ответы на вопросы 10](#_Toc159231530)

Задание (Вариант 10)

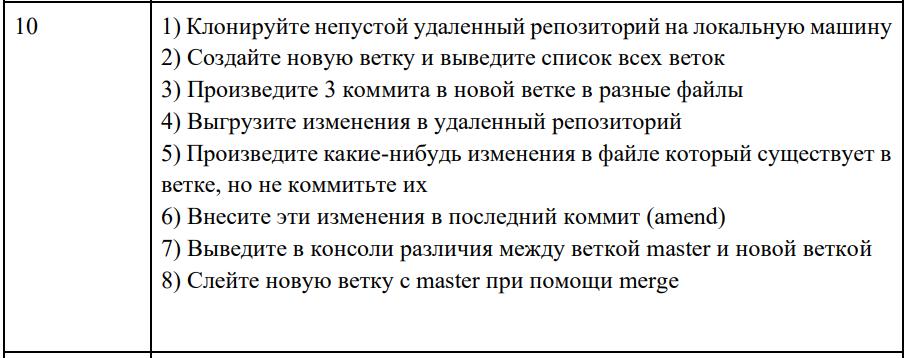
Цель работы: получить навыки по работе с командной строкой и git’ом.

Часть 1. Основные команды Git

1. Установите и настройте клиент git на своей рабочей станции.
2. Создайте локальный репозиторий и добавьте в него несколько файлов.
3. Внесите изменения в один из файлов.
4. Проиндексируйте изменения и проверьте состояние.
5. Сделайте коммит того, что было проиндексировано в репозиторий. Добавьте к коммиту комментарий.
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.
7. Просмотрите историю коммитов с помощью команды git log. Ознакомьтесь с параметрами команды и используйте некоторые из них для различного формата отображения истории коммитов.
8. Верните рабочий каталог к одному из предыдущих состояний.
9. Изучите, как создавать теги для коммитов для использования в будущем.
10. Отмените некоторые изменения в рабочем каталоге (до и после индексирования).
11. Отмените один из коммитов в локальном репозитории.

Часть 2. Системы управления репозиториями

1. Создайте аккаунт на GitHub (у кого нет)
2. Создайте репозиторий на GitHub и на локальной машине, согласно выбранной теме проекта
3. Создайте несколько файлов на локальной машине при помощи консоли
4. Создайте SSH-ключ для авторизации
5. Свяжите репозиторий локальной машины с репозиторием на GitHub при помощи консоли
6. Создайте новую ветку в репозитории с помощью команды, произведите в ней какие-нибудь изменения, а после слейте с веткой master
7. Выполните цепочку действий в репозитории, согласно вариантам.



Часть 3. Работа с ветвлением и оформление кода

* 1. Сделайте форк репозитория в соответствии с вашим вариантом (10 - )
  2. Склонируйте его на локальную машину
  3. Создайте две ветки branch1 и branch2 от последнего коммита в master'е
  4. Проведите по 3 коммита в каждую из веток, которые меняют один и тот же кусочек файла
  5. Выполните слияние ветки branch1 в ветку branch2, разрешив конфликты при этом
  6. Выгрузите все изменения во всех ветках в удаленный репозиторий
  7. Проведите еще 3 коммита в ветку branch1
  8. Склонируйте репозиторий еще раз в другую директорию
  9. В новом клоне репозитории сделайте 3 коммита в ветку branch1
  10. Выгрузите все изменения из нового репозитория в удаленный репозиторий

1. Вернитесь в старый клон с репозиторием, выгрузите изменения с опцией --force
2. Получите все изменения в новом репозитории

Выполнение заданий

Часть 1. Основные команды Git

Установим и настроим клиент git на своей рабочей станции. Для этого перейдем на сайт https://git-scm.com/download/win и установим Git на свою рабочую станцию. Далее установим глобальные переменные имени и почты юзера и выведем информацию на консоль (рис. 1).

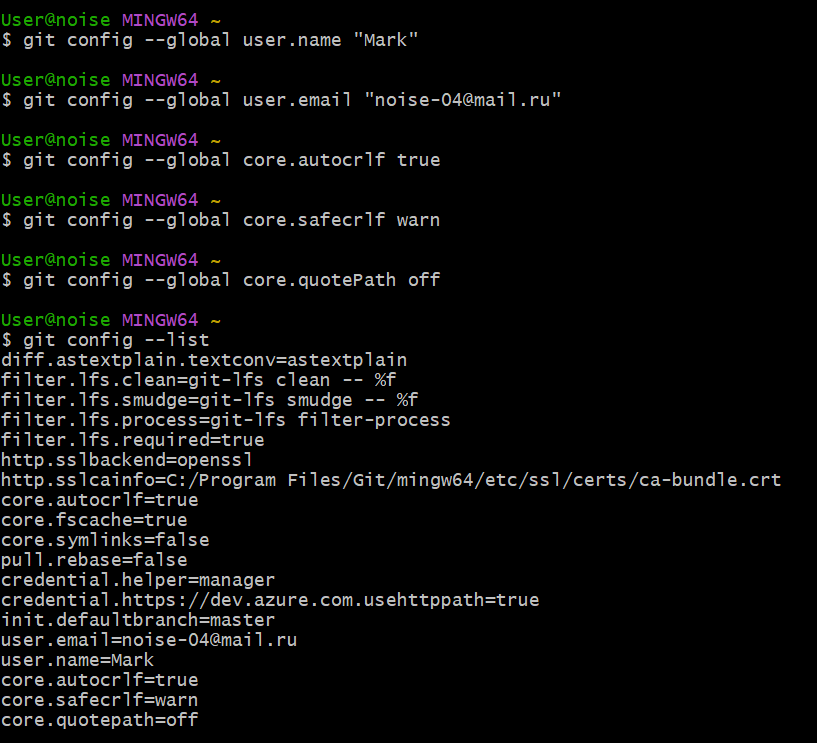


Рисунок 1 – Настройка гита

Создадим гит и проиндексируем созданные файлы (рис 2, 3).

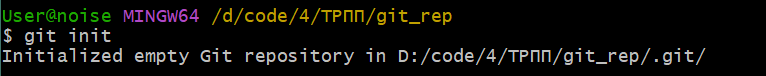


Рисунок 2 – Создание гита

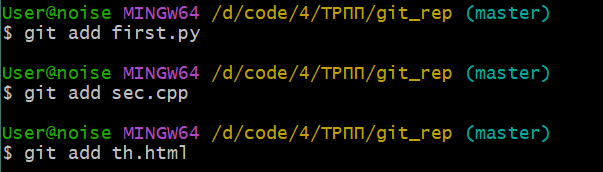


Рисунок 3 – Индексация созданных файлов

Внесем изменения и проиндексируем файл, затем проверим его состояние (рис. 4).

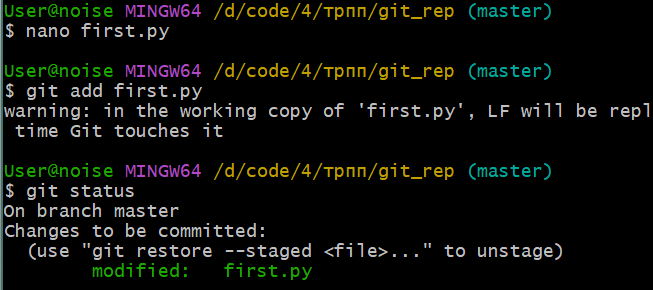


Рисунок 4 – Внесение изменений в файл и их индексация

Создадим коммит с комментариями (рис. 5).

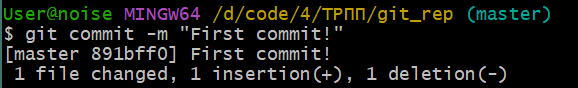


Рисунок 5 – Создание коммита

Изменим еще один файл. Добавим это изменение в индекс git и изменим файл еще раз. Проверим состояние и произведем коммит проиндексированного изменения. Добавим второе изменение в индекс. Сделаем коммит второго изменения (рис. 6).

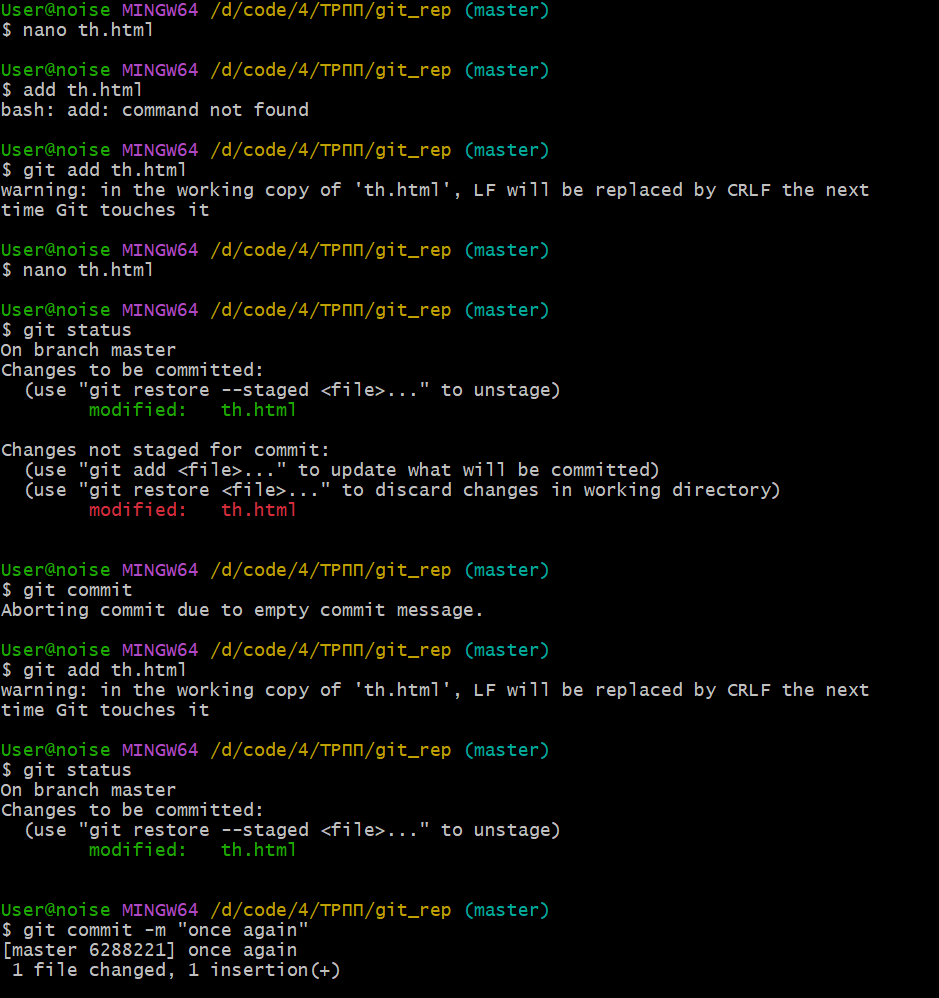


Рисунок 6 – Индексация и коммит файлов в разном порядке

Просмотрим историю коммитов с помощью команды git log. Ознакомимся с параметрами команды и используйте некоторые из них для различного формата отображения истории коммитов (рис. 7).

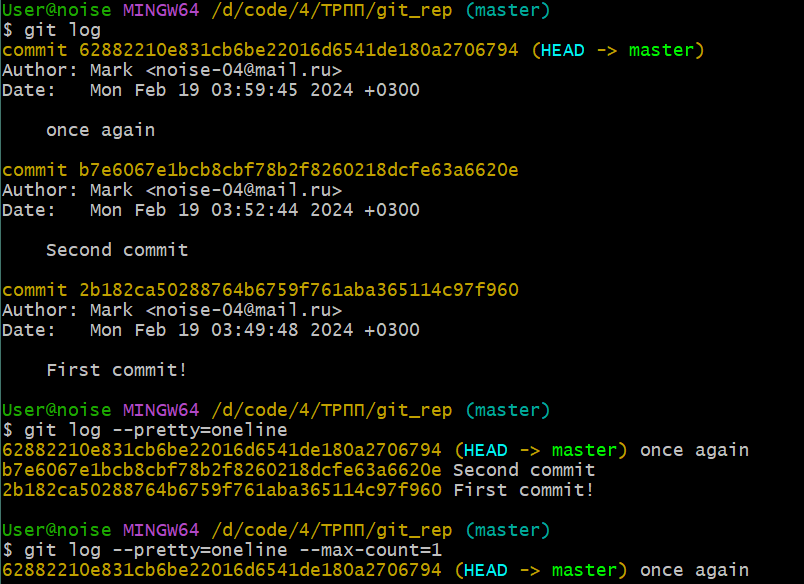


Рисунок 7 – Просмотр истории коммитов

Вернем рабочий каталог к одному из предыдущих состояний (рис. 8).

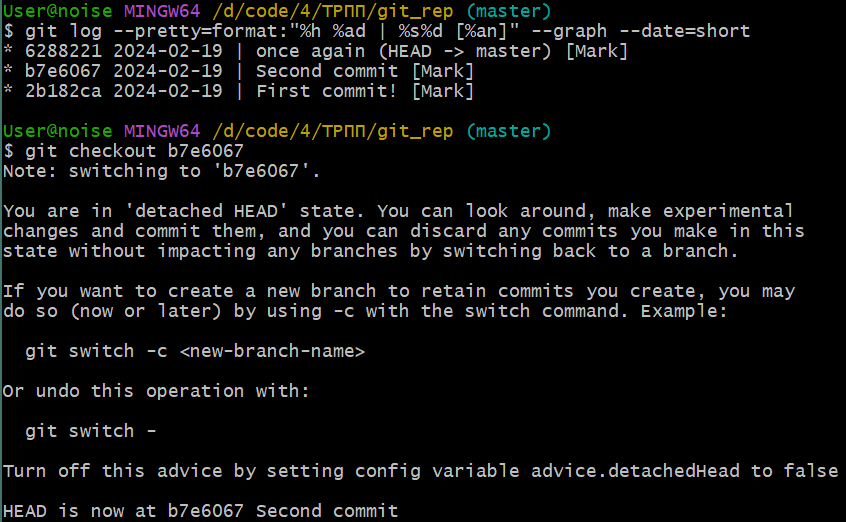


Рисунок 8 – Возврат каталога в предыдущее состояние

Создадим тег для коммита (рис. 9).



Рисунок 9 – Создание тега для коммита

Отменим некоторые изменения в рабочем каталоге (до и после индексирования) (рис. 10).

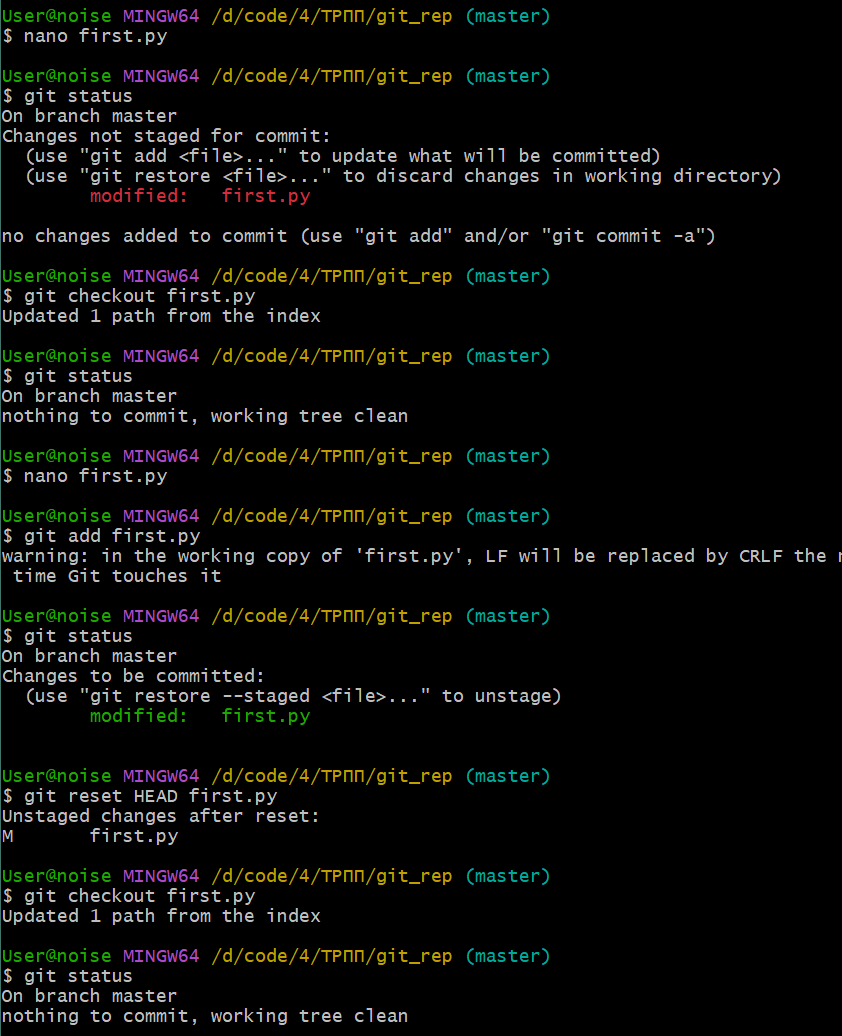


Рисунок 10 – Отмена изменений в файле

Отменим один из коммитов в локальном репозитории (рис. 11).

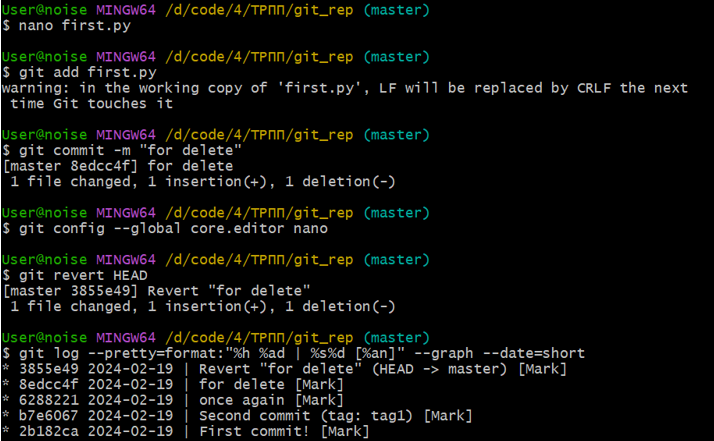


Рисунок 11 – Отмена коммита

Часть 2. Системы управления репозиториями

Создадим репозиторий на GitHub и на локальной машине и создадим в нем несколько файлов (рис. 12, 13).

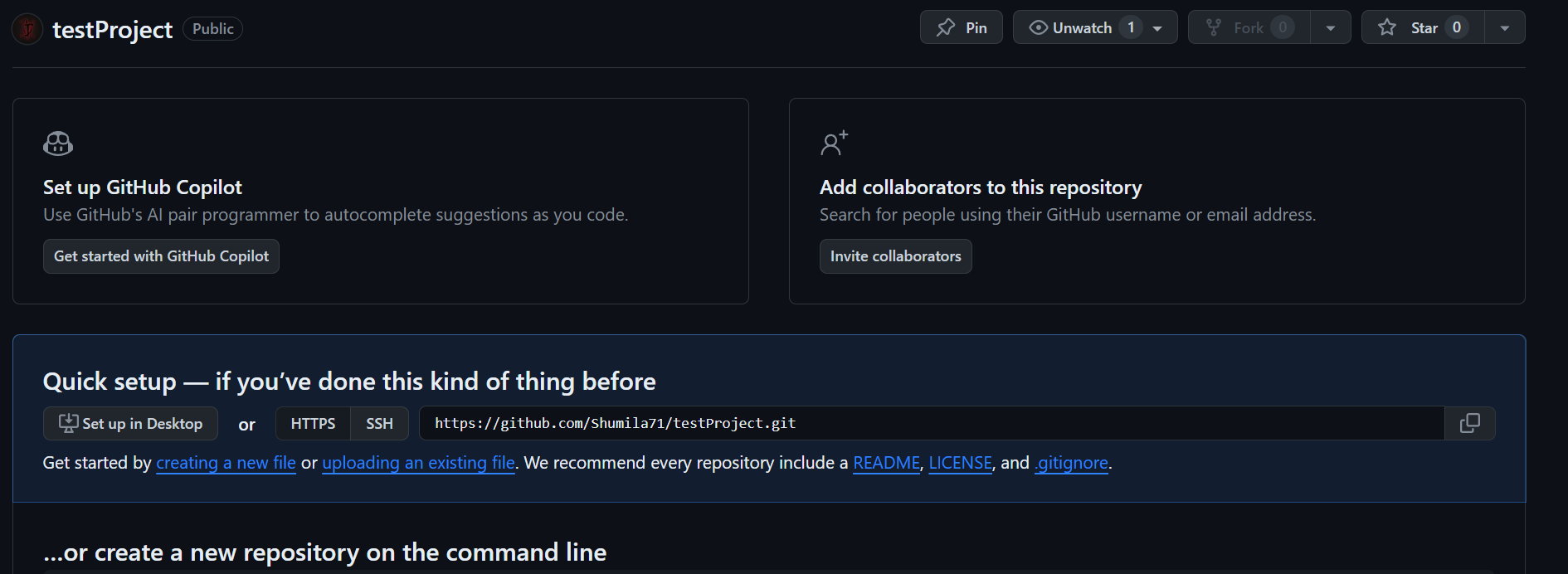


Рисунок 12 – Создание репозитория на GitHub

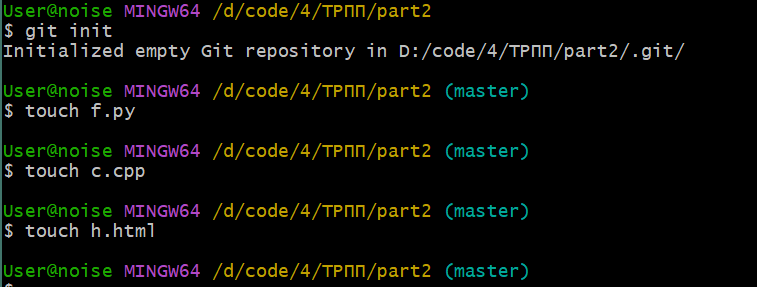
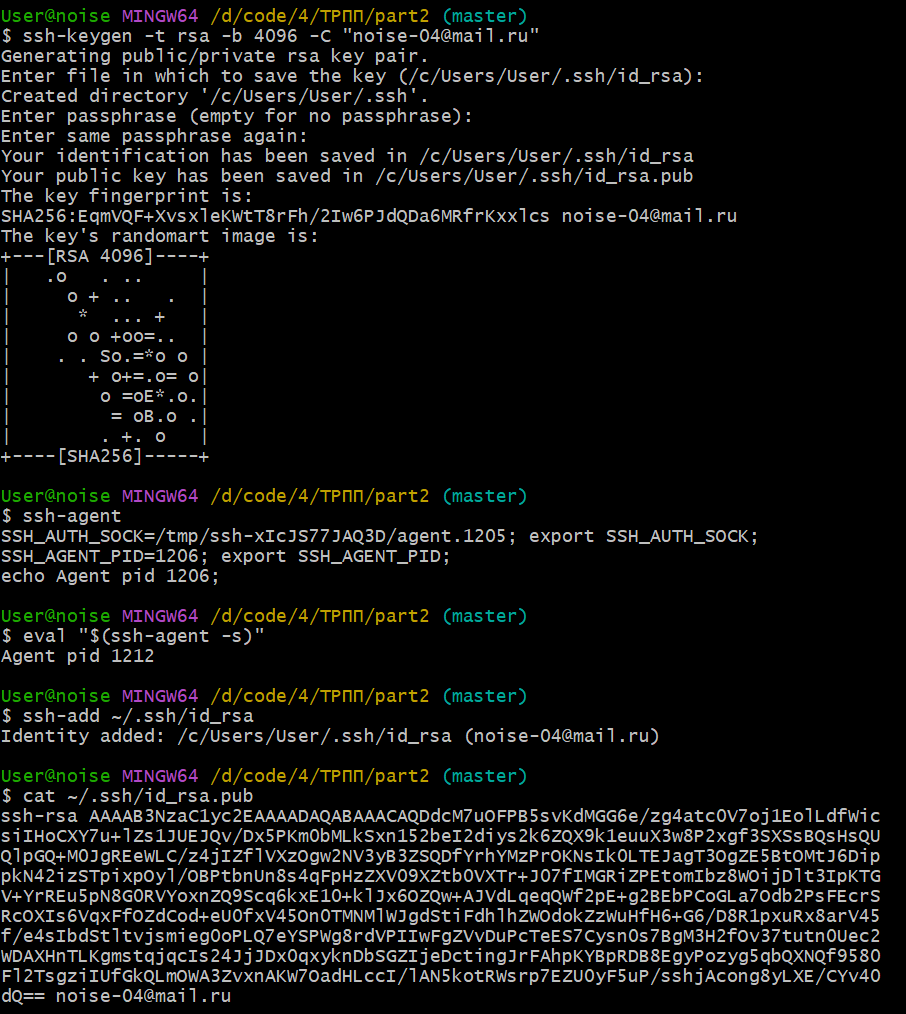


Рисунок 13 – Создание репозитория и нескольких файлов

Создадим SSH-ключ для авторизации (рис. 13).



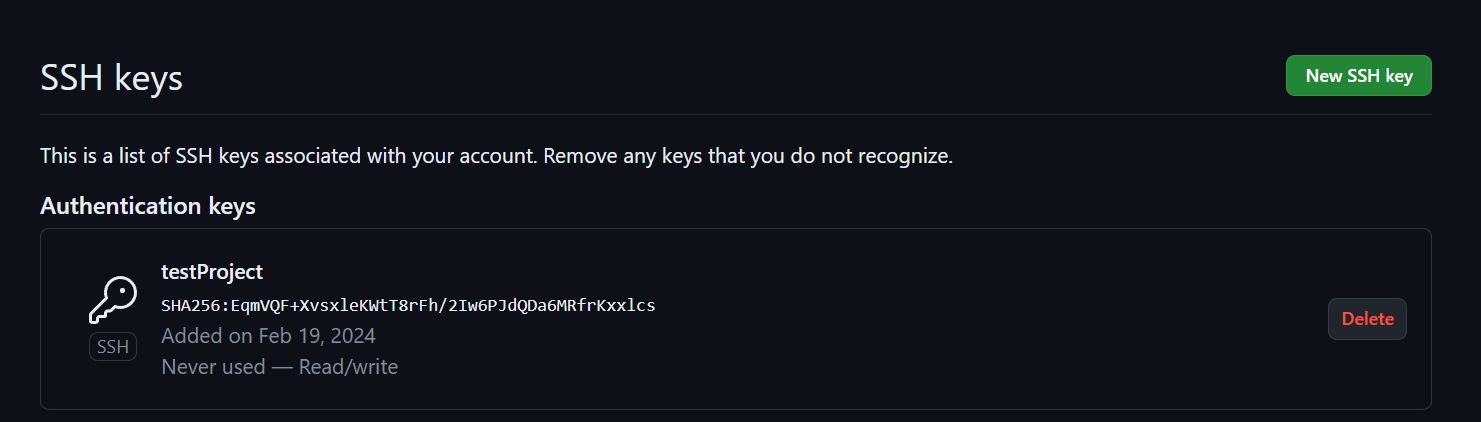


Рисунок 14 – Создание SSH-ключей

Свяжем локальный и удаленный репозитории между собой (рис. 15).



Рисунок 15 – Связь репозиториев

Создадим новую ветку в репозитории с помощью команды, произведем в ней какие-нибудь изменения, а после merge с веткой master (рис. 16, 17).

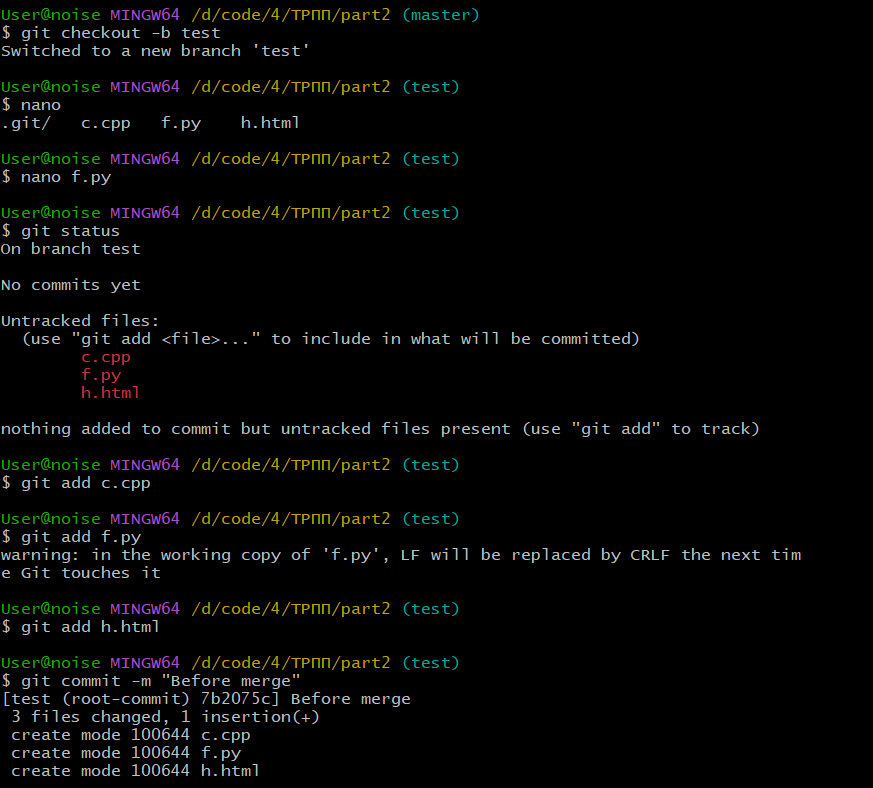
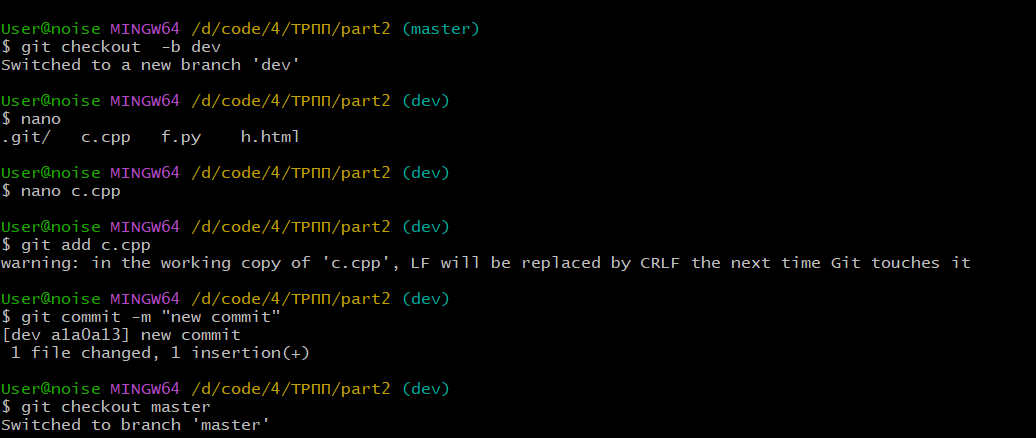


Рисунок 16 – Создание ветки и ее коммит



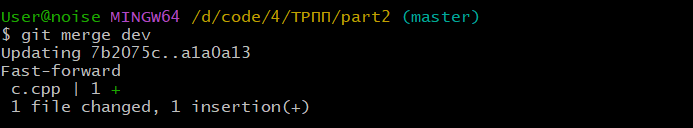


Рисунок 17 – Слияние веток

Выполним цепочку действий в репозитории, согласно варианту 10:

Клонируем непустой удаленный репозиторий на локальную машину.

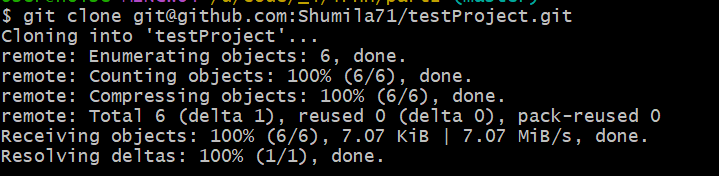


Рисунок 18 ­– Клонирование репозитория

Создадим новую ветку и выведем их список.

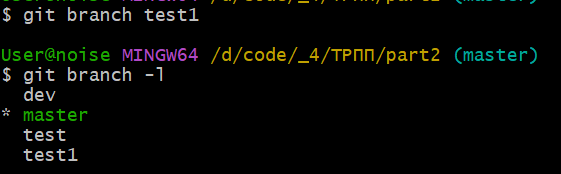


Рисунок 19 – Создание и вывод веток

Произведем 3 коммита в разных файлах.

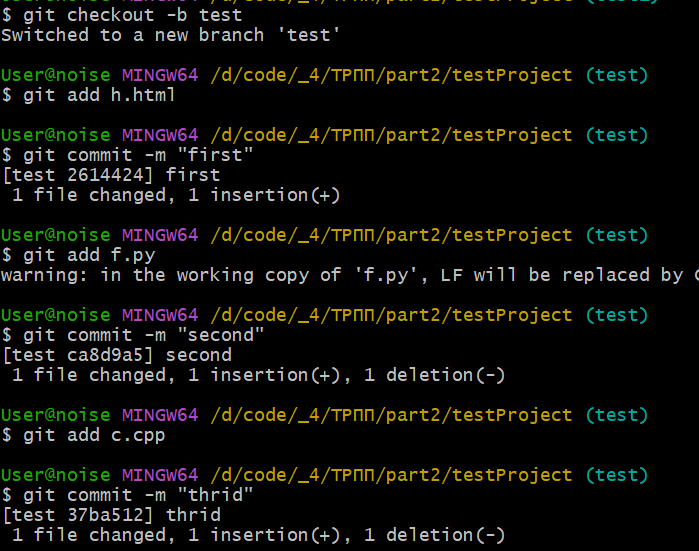


Рисунок 20 – Создание коммитов

Выгрузим изменения в удаленный репозиторий.



Рисунок 21 – Выгрузка изменений в удаленный репозиторий.

Произведите какие-нибудь изменения в файле, который существует в ветке, но не будем их коммитить.



Рисунок 22 – Изменения в файле

Внесем эти изменения в последний коммит.

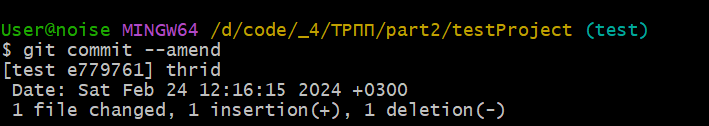


Рисунок 23 – Внесения изменений в последний коммит

Сольем новую ветку с веткой main.

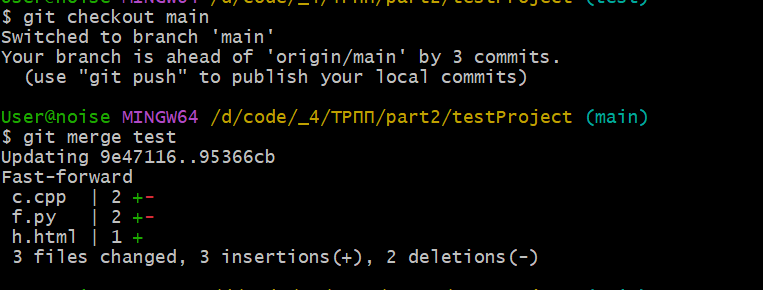


Рисунок 24 – Объединение веток

Часть 3. Ветвление и оформление кода

Сделаем форк репозитория в соответствии с вашим вариантом.

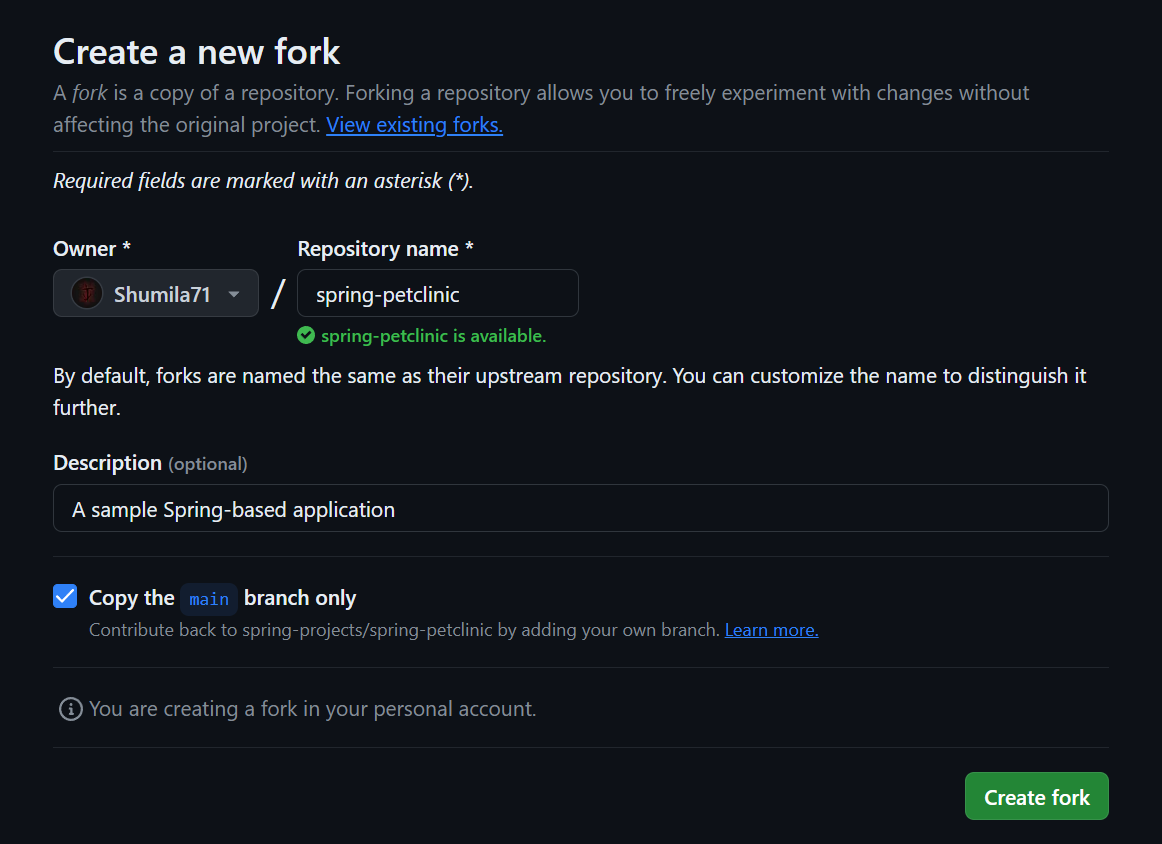


Рисунок 25 – Форк репозитория

Создадим копию на локальной машине.

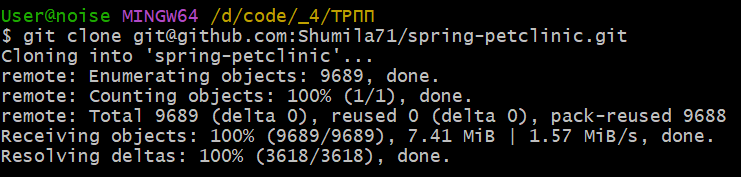


Рисунок 26 – Клонирование репозитория

Создаем две ветки branch1 и branch2 от последнего коммита в master'е.

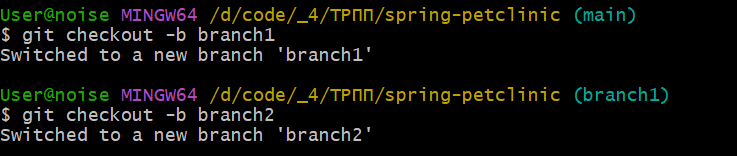
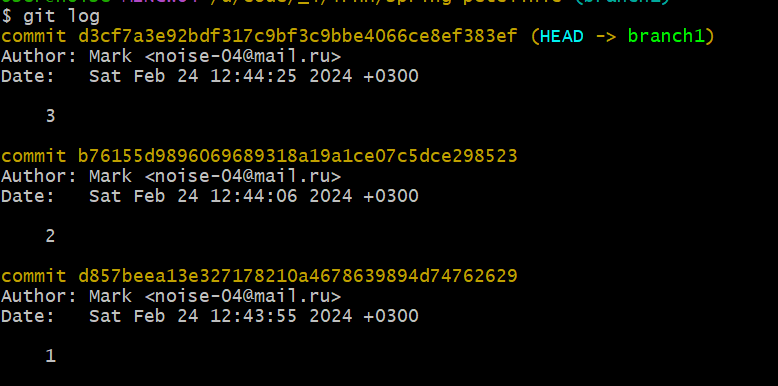


Рисунок 27 – Создание веток

Проведем по 3 коммита в каждую из веток, которые меняют один и тот же кусочек файла.



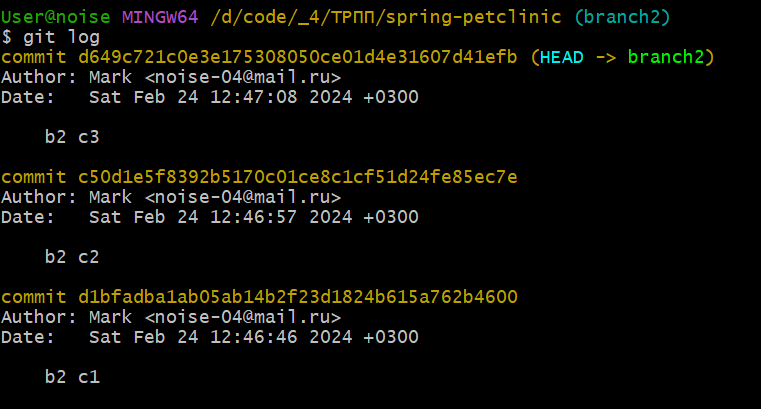


Рисунок 28 – Создание коммитов

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

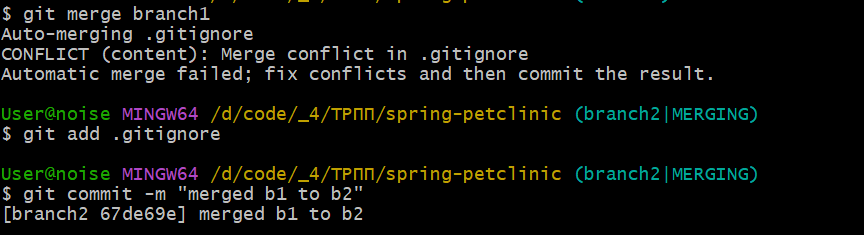


Рисунок 29 – Слияние веток

Выгрузим все изменения во всех ветках в удаленный репозиторий.

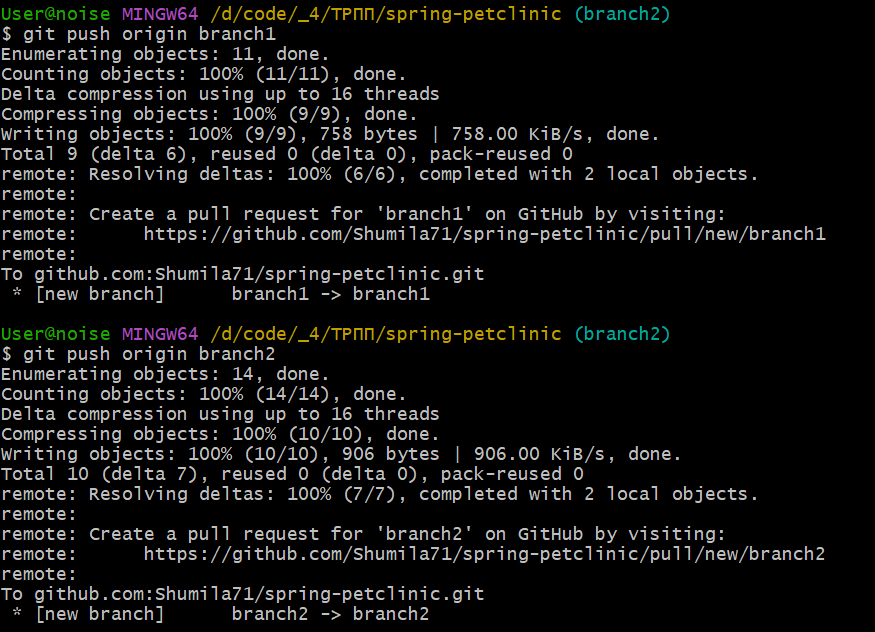


Рисунок 30 – Выгрузка изменений

Проведите еще 3 коммита в ветку branch1.

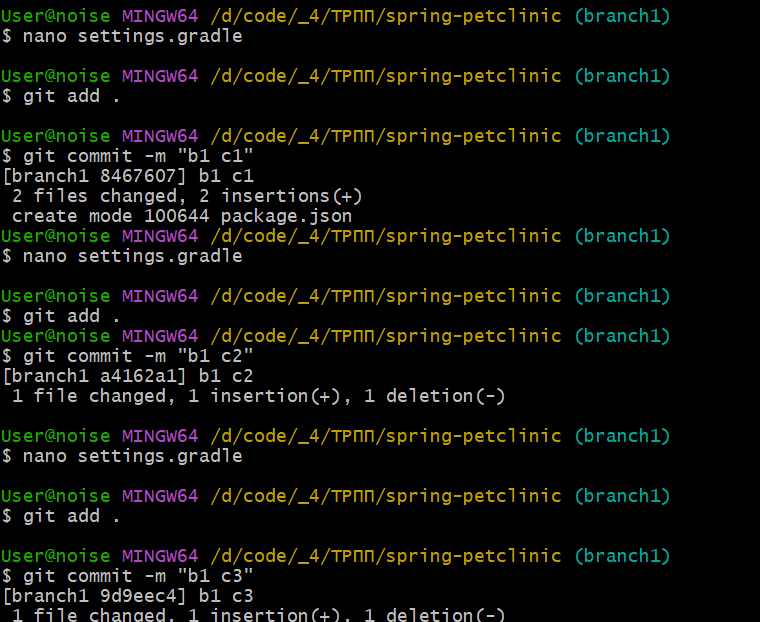


Рисунок 31 – Создание коммитов

Склонируем репозиторий еще раз в другую директорию.

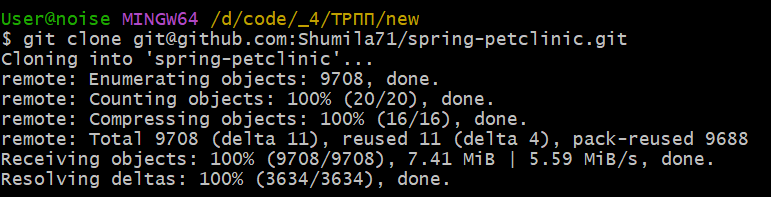


Рисунок 32 – Клонирование репозитория

В новом клоне репозитория сделаем 3 коммита в ветку branch1.

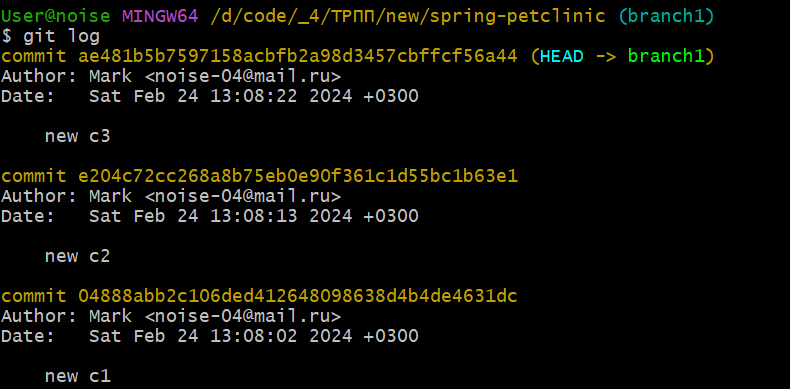


Рисунок 33 – Создание коммитов

Выгрузим все изменения из нового репозитория в удаленный репозиторий.

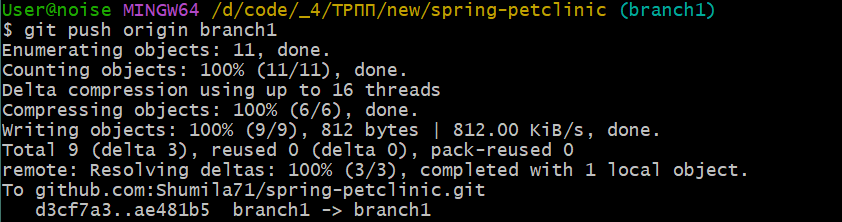


Рисунок 34 – Выгрузка в удаленный репозиторий

Выгрузим изменения с опцией --force.

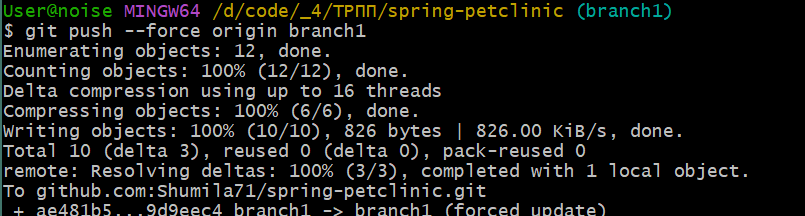


Рисунок 35 – Выгрузка с опцией force

Получим все изменения в новом репозитории.

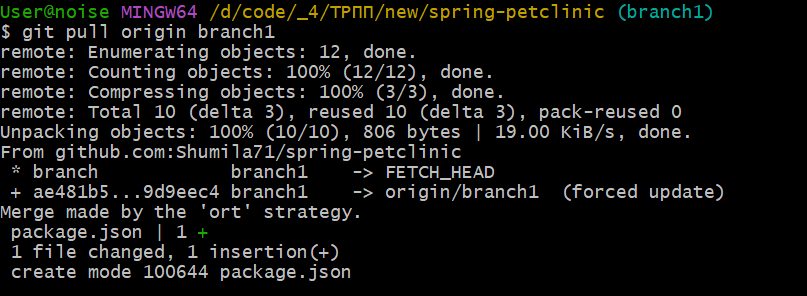


Рисунок 36 – Получение изменений

Ответы на вопросы

1 .Какие существуют типы систем контроля версий? Приведите примеры к каждому типу.

Локальные системы контроля версий: Хранятся только на компьютере пользователя. Пример: RCS.

Централизованные системы контроля версий: Используют центральный сервер для хранения файлов. Примеры: CVS, Subversion.

Распределенные системы контроля версий: Каждый пользователь имеет полную копию репозитория. Примеры: Git, Mercurial, Bazaar.

2. К какому типу систем контроля версий относится Git?

Git относится к типу распределенных систем контроля версий.

3. Что такое репозиторий Git?

Репозиторий Git - это место, где хранятся файлы вашего проекта, история их изменений, информация о ветках, метках и других аспектах управления версиями. Это основное хранилище данных, используемое Git для отслеживания изменений и совместной работы над проектом. Репозиторий Git может быть как локальным (на вашем компьютере), так и удаленным (размещенным на сервере). Он содержит все необходимые файлы и метаданные, чтобы Git мог эффективно управлять версиями вашего проекта.

4. Что такое коммит?

Коммит в Git - это сохранение изменений в проекте. Он содержит информацию о том, что изменилось, кем были внесены изменения, и когда это произошло. Коммиты помогают отслеживать историю изменений и восстанавливать предыдущие версии проекта.

5. Что такое ветка в репозитории Git?

Ветка в Git - это отдельная линия развития проекта, которая позволяет работать над изменениями независимо от основной версии.

6. Для чего нужен .gitignore?

.gitignore - это файл, используемый Git для указания файлов и каталогов, которые не должны быть отслеживаемыми системой контроля версий. Этот файл игнорирует указанные файлы и каталоги при выполнении команды git status или git add, что позволяет избежать добавления временных файлов, файлов компиляции, файлов с конфиденциальной информацией и других ненужных элементов в репозиторий Git. Таким образом, .gitignore помогает поддерживать чистоту и организованность вашего проекта, исключая из него файлы, которые не должны быть там сохранены.

7. Что означает статус файла untracked в выводе команды git status?

Статус файла "untracked" в выводе команды git status означает, что Git обнаружил файл, который не отслеживается системой контроля версий. Это означает, что файл был создан или изменен, но еще не был добавлен в индекс Git с помощью команды git add.

8. Чем отличается master и origin/master?

master: это название локальной ветки вашего репозитория. Она представляет собой основную линию разработки, на которой вы работаете локально.

origin/master: это название удаленной ветки, которая отслеживает состояние ветки master на удаленном сервере (обычно на сервере, с которого вы склонировали репозиторий). Таким образом, origin/master - это локальная копия удаленной ветки master, которая отражает ее текущее состояние на удаленном сервере.

Когда вы клонируете репозиторий, Git автоматически настраивает отслеживание удаленной ветки master под именем origin/master. При получении обновлений из удаленного репозитория, вы можете обновить вашу локальную ветку master, чтобы она отражала изменения на сервере, используя команду git pull origin master.

Вывод

В результате выполнения работы были получены навыки по работе с командной строкой и Git.