# Работа с Git

**Первичная настройка** При первом использовании Git требуется задать идентификационные данные пользователя. Выполняются две команды терминала (Рис. 1) для регистрации имени и электронного адреса.

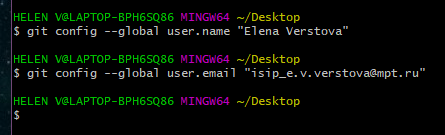


Рисунок 1 - Установка имени и электронной почты

Далее настраивается обработка символов перевода строки (Рис. 2)

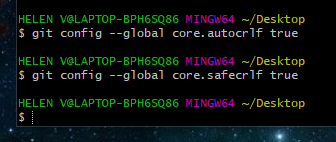


Рисунок 2 - Параметры окончаний строк

и параметры отображения Unicode-символов (Рис. 3).

Рисунок 3 - Установка отображения unicode

1. **Инициализация проекта** Создайте пустую директорию и поместите в нее файл hello.html (Рис. 4)

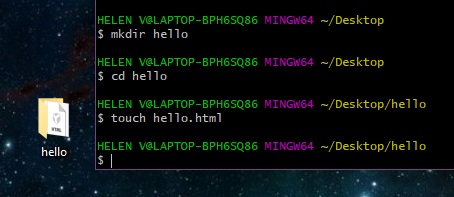


Рисунок 4 - Создание каталога и файла

Добавьте текстовое содержимое в файл, например, строку приветствия (Рис. 5)

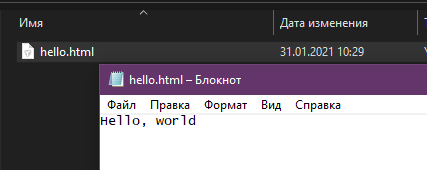


Рисунок 5 - Содержание файла

Инициализируйте репозиторий командой git init (Рис. 6)

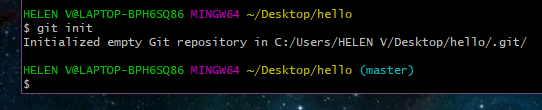


Рисунок 6 - Создание репозитория

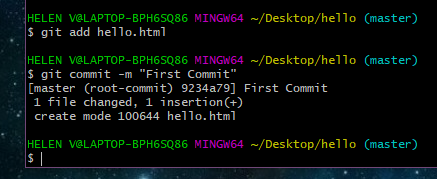
Зарегистрируйте файл в системе контроля версий через git add (Рис. 7)

Рисунок 7 - Добавление в репозиторий

**Мониторинг состояния** Текущий статус отслеживаемых файлов проверяется командой git status. При отсутствии изменений вывод соответствует Рис. 8.

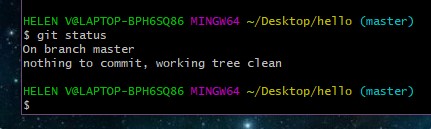


Рисунок 8 - Проверка состояния репозитория

**Внесение правок** Измените содержимое файла (Рис. 9)

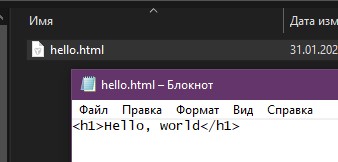


Рисунок 9 - Внесение изменений в файл

Повторная проверка статуса отобразит уведомление о незафиксированных изменениях (Рис. 10)

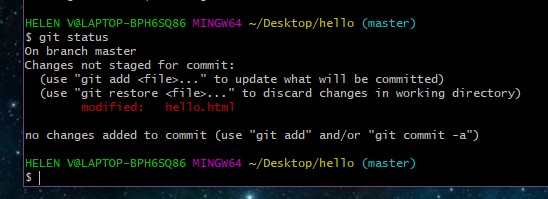


Рисунок 10 - Сообщение о незафиксированных изменениях

**Подготовка к фиксации**   
Для индексации изменений выполните последовательность команд из Рис. 11.

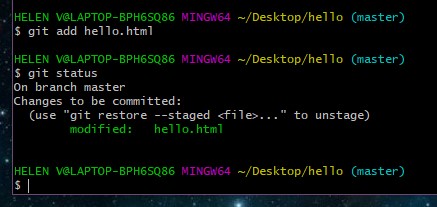


Рисунок 11 - Команды для индексации изменений

Индексацию можно отменить через git reset до момента фиксации.

**Многоэтапная фиксация** При работе с несколькими файлами возможна поэтапная фиксация:

* Создайте три файла (Рис. 12)
* Зафиксируйте первые два файла отдельным коммитом (Рис. 13)
* Третий файл добавьте следующим коммитом (Рис. 14)

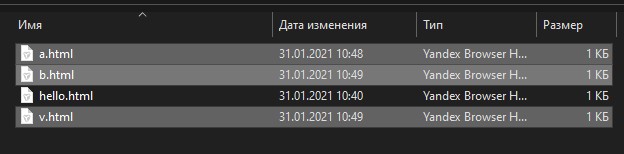


Рисунок 12 - Создано 3 файла

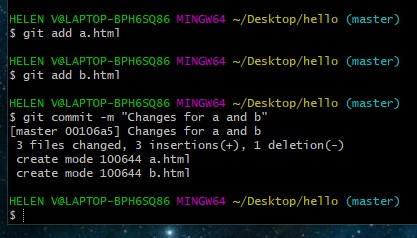


Рисунок 13 - Индексация и коммит для 2 файлов

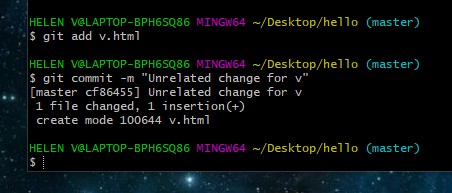


Рисунок 14 - Индексация и коммит для третьего файла

**Редактирование комментариев** Используйте git commit без параметра -m (Рис. 15)

1. В открывшемся редакторе укажите описание (Рис. 16)
2. После сохранения система выведет подтверждение (Рис. 17)
3. Проверьте статус репозитория (Рис. 18)

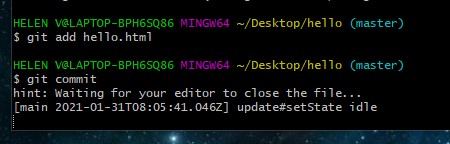


Рисунок 15 - Коммит изменений

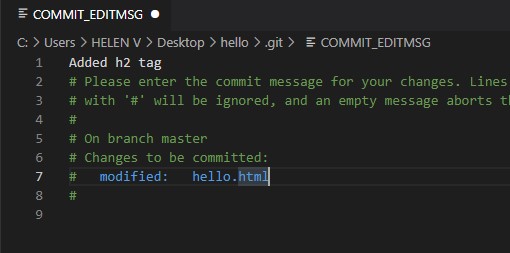


Рисунок 16 - Ввод комментария

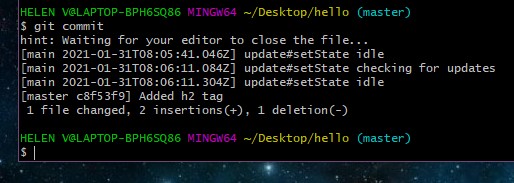


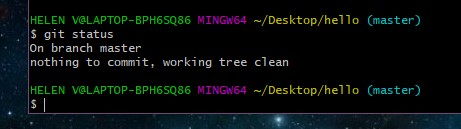
Рисунок 17 - Коммит-сообщения

Рисунок 18 - Проверка состояния

**Фиксация изменений вместо файлов** Система отслеживает изменения в файлах, а не сами файлы:

1. Добавьте HTML-теги в файл (Рис. 19) и проиндексируйте
2. Внесите дополнительные правки (тег head) без индексации (Рис. 20)
3. Проверка статуса покажет два состояния одного файла (Рис. 21)
4. Зафиксируйте первое изменение (Рис. 22)
5. Проиндексируйте и закоммитьте второе изменение (Рис. 23-24)

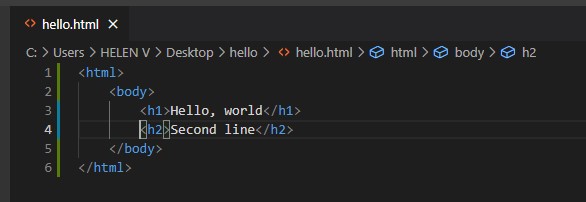


Рисунок 19 - Добавление тегов html и body

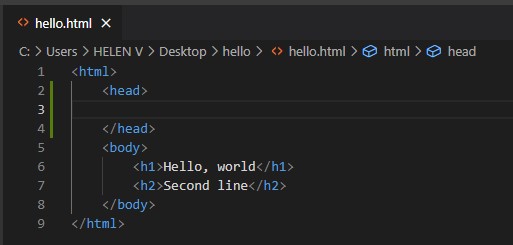


Рисунок 20 - Добавление тега head

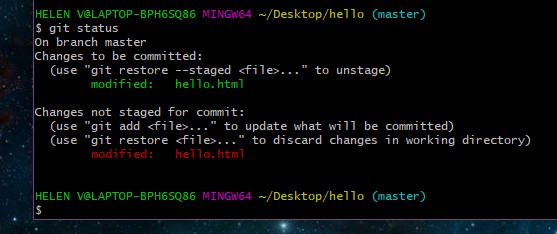


Рисунок 21 - Проверка состояния

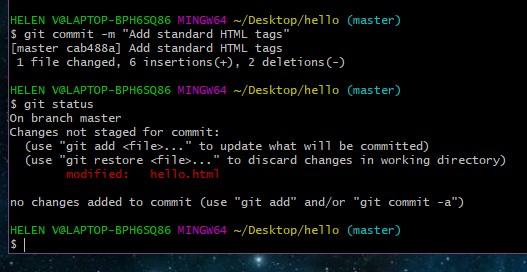


Рисунок 22 - Коммит 1 изменения и проверка состояния

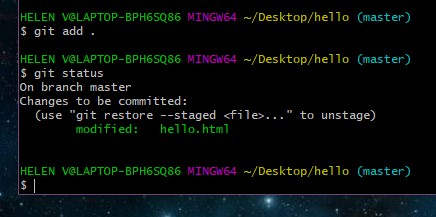
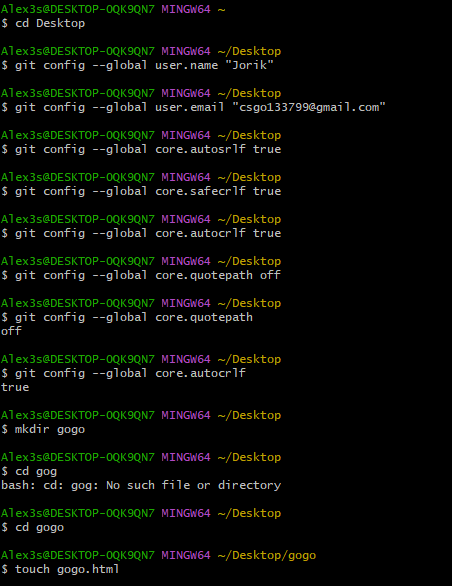


Рисунок 23 - Индексация 2 изменения и проверка состояния



Рисунок 24 - Коммит 2 изменения



Изображение выглядит как текст, снимок экрана

Контент, сгенерированный ИИ, может содержать ошибки.

Изображение выглядит как текст, снимок экрана, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.

Изображение выглядит как текст, снимок экрана, Шрифт, дизайн

Контент, сгенерированный ИИ, может содержать ошибки.

# Работа с Git

## История

Для того, чтобы просмотреть список произведенных изменений в проекте, используется команда git log (рисунок 1).



Рисунок 25 - Просмотр истории изменений

На рисунке 1 была выведена полная история. Для того, чтобы увидеть однострочный формат используется команда git log –pretty=oneline (рисунок 2).



Рисунок 26 - Однострочный формат вывода

Далее на рисунках 3-8 показано несколько вариантов вывода истории изменений.

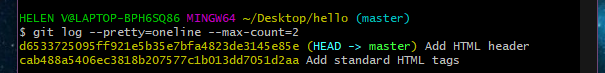


Рисунок 27 - Вывод последних 2 изменений

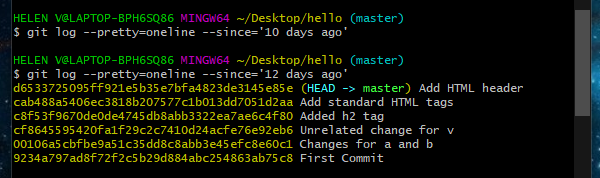


Рисунок 28 - Вывод изменений начиная с определенного времени

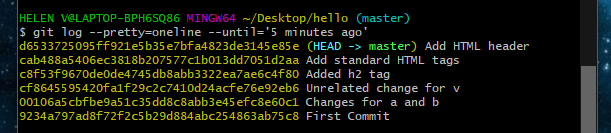


Рисунок 29 - Вывод изменений до определенного времени

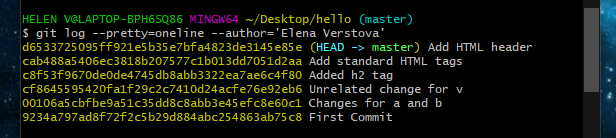


Рисунок 30 - Вывод изменений, внесенных определенным автором

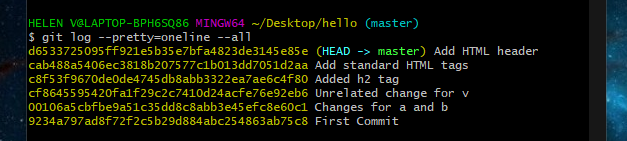


Рисунок 31 - Вывод всех изменений

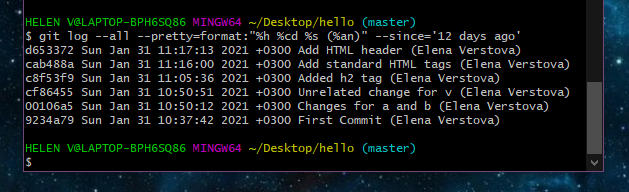


Рисунок 32 - Использование нескольких параметров

## Алиасы

Для настройки алиасов используется команда, показанная на рисунке 9.

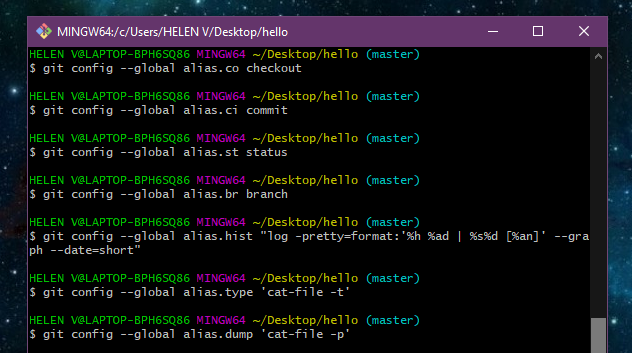


Рисунок 33 - Настройка алиасов для некоторых команд

При выполнении алиаса будет выполнена определенная команда и выведены нужные данные (рисунок 10).

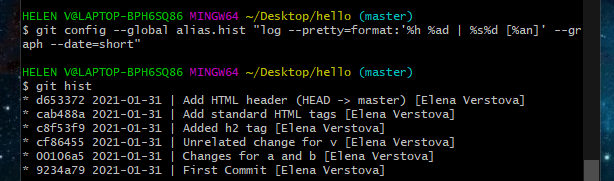


Рисунок 34 - Выполнение алиаса hist

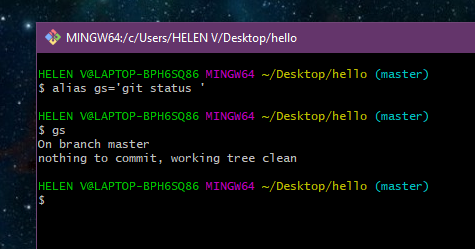


Рисунок 35 - Установка и выполнение алиаса gs

## Получение старых версий

Для того, чтобы вернуть рабочий каталог к предыдущему состоянию, можно использовать следующий способ: для начала нужно узнать хэши предыдущих версий, что можно сделать с помощью ранее заданного алиаса hist (рисунок 12).

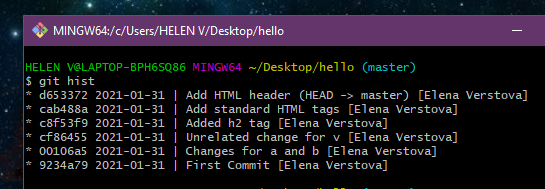


Рисунок 36 - Просмотр хэшей предыдущих версий

Далее нужно выполнить команду git checkout с номером нужного хэша (достаточно первых 7 знаков). После этого можно просмотреть содержимое файла с помощью команды cat (рисунок 13).

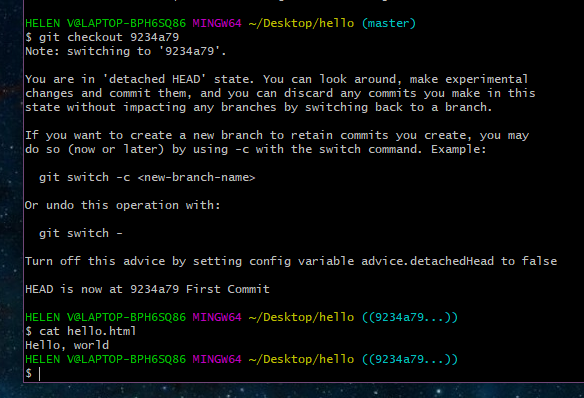


Рисунок 37 - Возвращение к нужной версии и просмотр содержимого файла

## Возвращение к последней версии в ветке master

Для возвращения к последней версии в ветке master (имя ветки по умолчанию) надо ввести команду git checkout master, что показано на рисунке 14.

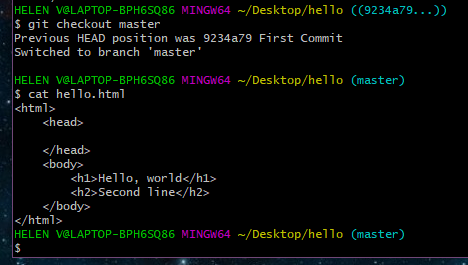


Рисунок 38 - Возвращение к последней версии в ветке master

## Создание тегов версий

Для создания тега используется команда git tag. На рисунке 15 показано, тегом ver1 была названа текущая версия страницы.

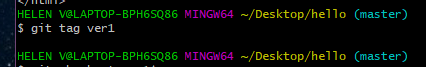


Рисунок 39 - Задание тега

Чтобы перейти к предыдущей версии, можно использовать символ «^», который означает «родитель».

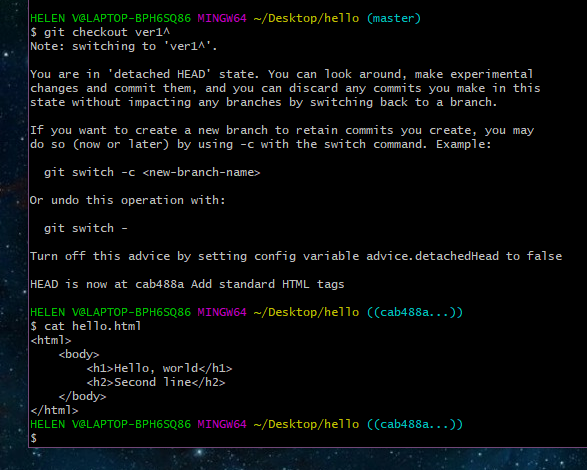


Рисунок 40 – Переход к предыдущей версии с помощью тега

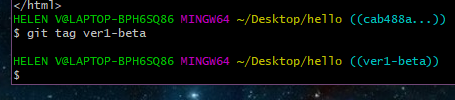


Рисунок 41 - Задание тега предыдущей версии

Теперь с помощью тегов можно переключаться между версиями (рисунок 18).



Рисунок 42 - Переключение между версиями с помощью тегов

Для просмотра всех тегов используется команда git tag (рисунок 19).

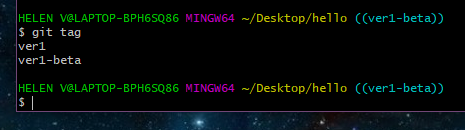


Рисунок 43 - Просмотр тегов

Также можно просмотреть теги в логе, как показано на рисунке 20.

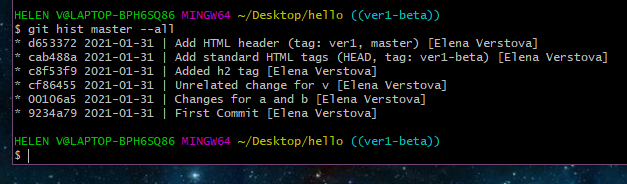


Рисунок 44 - Просмотр тегов в логе

## Отмена локальных изменений (до индексации)

Сначала нужно переключиться на последний коммит master (рисунок 21).

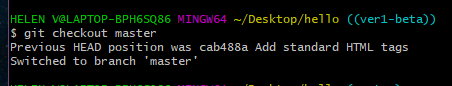


Рисунок 45 - Переключение на последний коммит

Далее для работы нужно внести изменение в файл (рисунок 22).

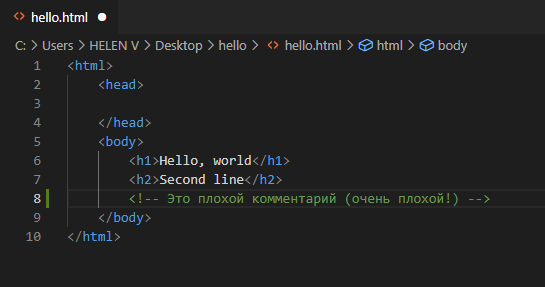


Рисунок 46 - Внесение изменения в файл

После выполнения команды git status будет показано, что есть не проиндексированное изменение (рисунок 23).

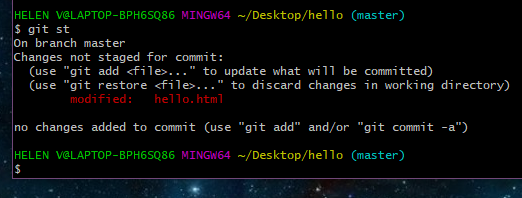


Рисунок 47 - Не проиндексированное изменение

Для переключения в версию файла без изменений используется команда git checkout hello.html (рисунок 24). Команда git status покажет, что не было произведено изменений, не зафиксированных в каталоге.

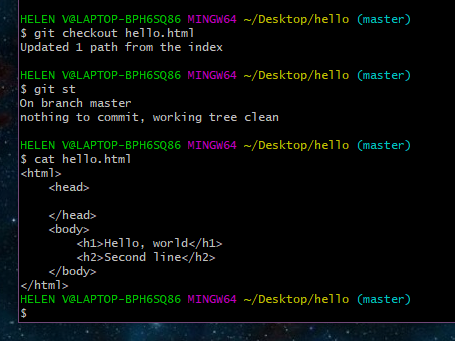


Рисунок 48 - Возвращение к версии

## Отмена проиндексированных изменений (перед коммитом)

Для того, чтобы научиться отменять проиндексированные изменения, сначала нужно внести ненужное изменение в файл (рисунок 25). После этого производится индексация (рисунок 26).

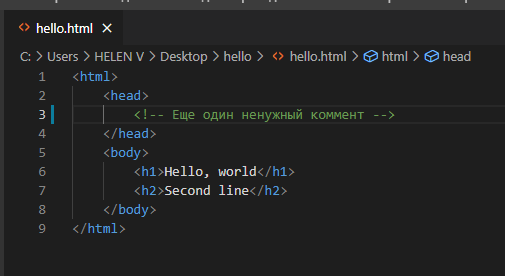


Рисунок 49 - Внесение ненужного изменения

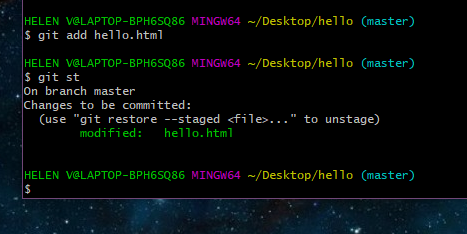


Рисунок 50 - Индексация изменения

Для отмены индексация изменения используется команда git reset HEAD hello.html (рисунок 27). Команда reset сбрасывает буферную зону к HEAD и очищает ее от проиндексированных изменений. Но для удаления ненужного по-прежнему используется команда git checkout (рисунок 28).

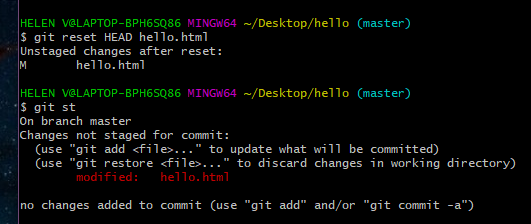


Рисунок 51 - Очистка буферной зоны

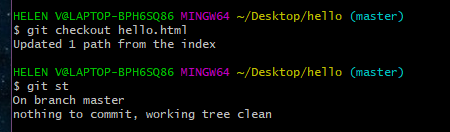


Рисунок 52 - Удаление ненужных изменений

## Отмена коммитов

Для отмены коммита можно использовать способ создания нового коммита, отменяющего изменения.

Для начала надо внести изменение, проиндексировать его и записать коммит (рисунки 29-30).



Рисунок 53 - Внесение изменения в файл

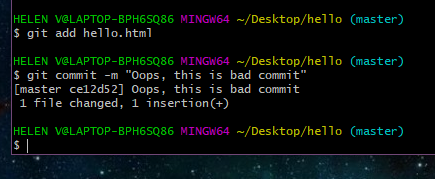


Рисунок 54 - Индексация и коммит

Для создания коммита, который удалит ненужные изменения, используется команда git revert HEAD (рисунок 31). После этого будет открыт редактор, в котором можно отредактировать коммит сообщение (рисунок 32), затем надо сохранить файл и закрыть редактор (рисунок 33).

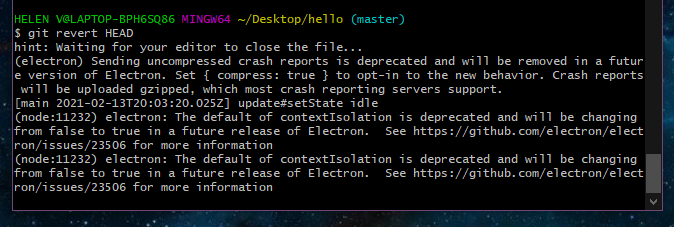


Рисунок 55 - Выполнение команды git revert

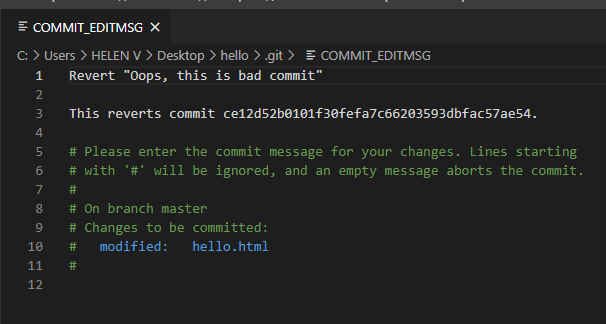


Рисунок 56 - Коммит сообщение в редакторе

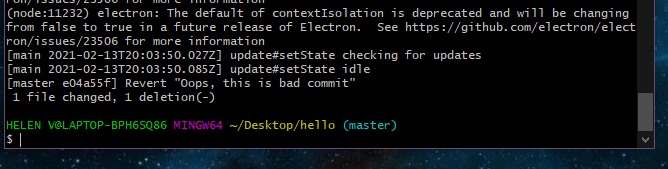


Рисунок 57 - Редактор закрыт

При проверке лога будут показаны все коммиты, в том числе и отмененные (рисунок 34).

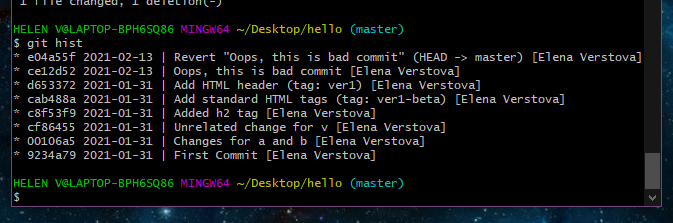


Рисунок 58 - Все коммиты при просмотре лога

Перед удалением коммита последний из них нужно отметить тегом, чтобы не потерять его (рисунок 35).

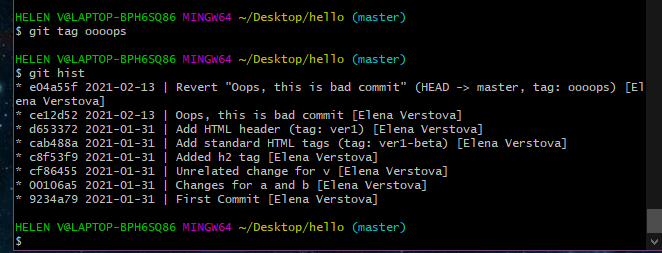


Рисунок 59 - Отметка тегом коммита

Для сброса коммитов используется команда git reset –hard ver1 (рисунок 36). Она сбрасывает ветку до версии с тегом ver1.

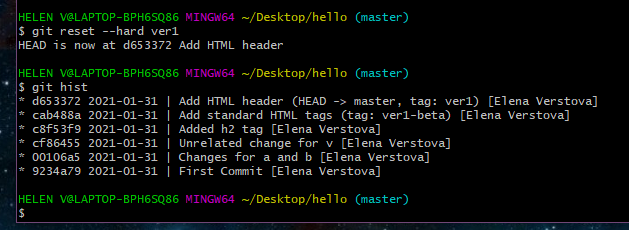


Рисунок 60 - Сброс коммита

Но при просмотре лога с помощью команды git hist –all отмененные коммиты по-прежнему будут показываться, так как они всё еще находятся в репозитории (рисунок 37).

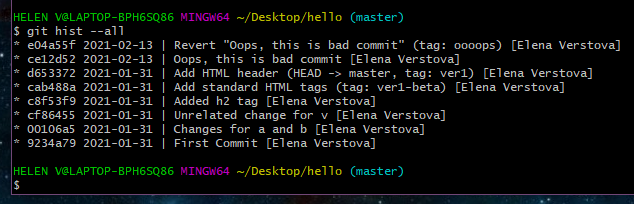


Рисунок 61 - Сброшенные коммиты находятся по-прежнему в репозитории

## Удаление тега

Так как тег «oooops» больше не нужен, его и коммиты, на которые он указывает, можно удалить с помощью команды git tag -d (рисунок 38).

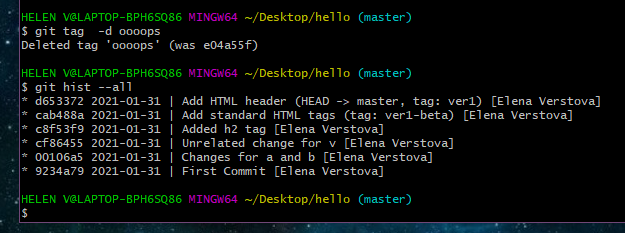
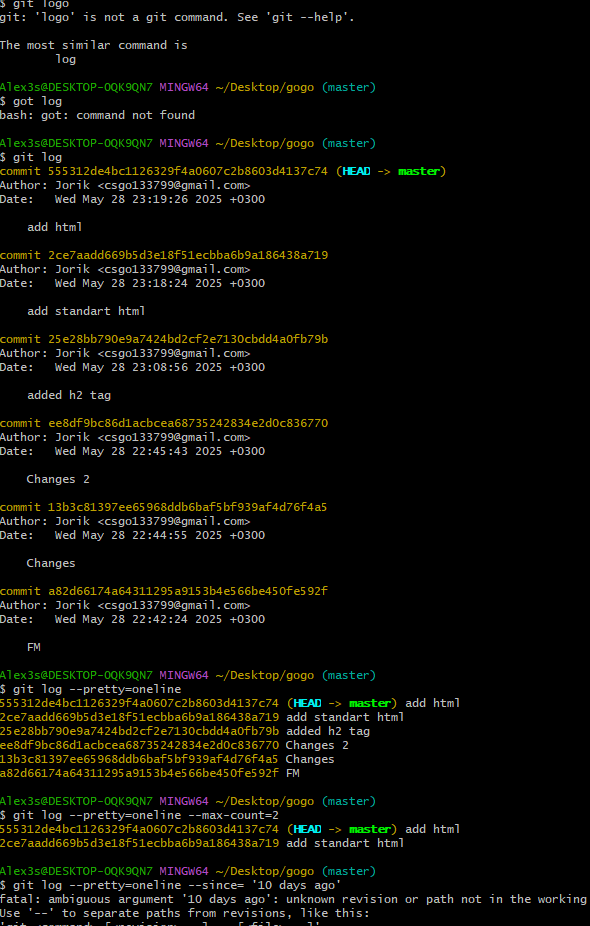


Рисунок 62 - Удаление тега



Изображение выглядит как текст, снимок экрана

Контент, сгенерированный ИИ, может содержать ошибки.

# Работа с Git

## Внесение изменений в коммиты

Для начала будет создан коммит, в который позже будут внесены изменения. На рисунках 1 и 2 происходит добавление комментария в файл hello.html и его индексация и коммит.

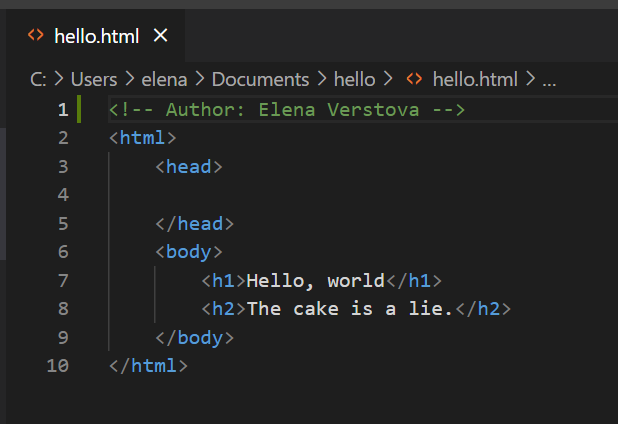


Рисунок 63 - Добавление комментария в файл

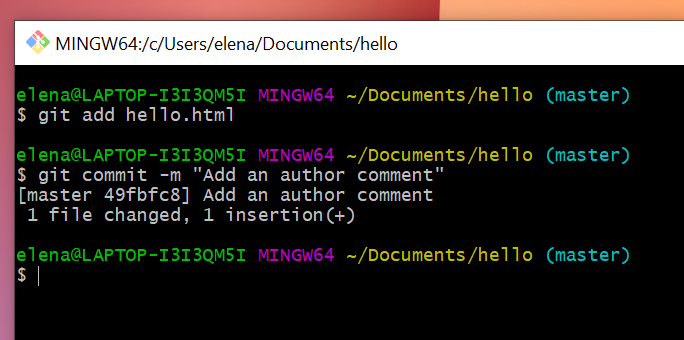


Рисунок 64 - Индексация и коммит

Далее необходимо добавить электронную почту в комментарий (рисунок 3).

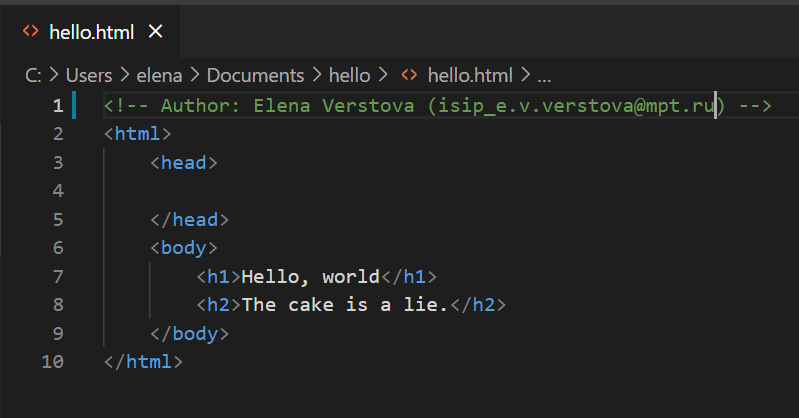


Рисунок 65 - Добавление электронной почты

Но для того, чтобы не создавать отдельный коммит ради электронной почты, можно изменить предыдущий так, как показано на рисунке 4.

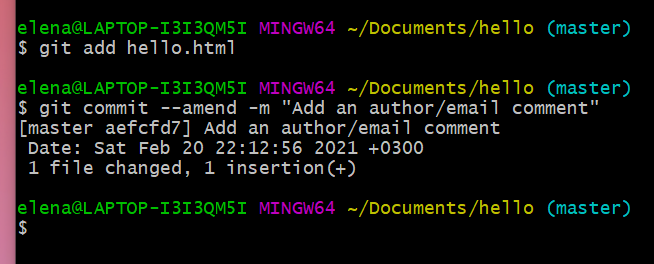


Рисунок 66 - Индексация и изменение коммита

При просмотре истории можно будет заметить, что последний коммит был изменен.

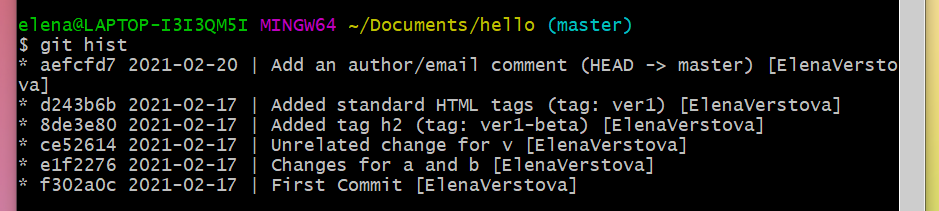


Рисунок 67 - Последний коммит изменен

## Перемещение файлов

Для перемещения файлов в пределах репозитория используются команды, показанные на рисунке 6. После выполнения данных команды git индексирует эти изменения (удаление файла hello.html и создание файла lib/hello.html).

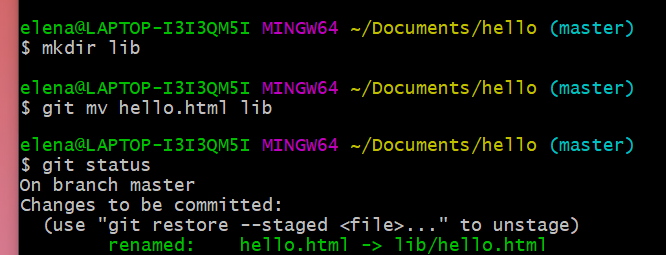


Рисунок 68 - Перемещение файлаgit

Далее надо осуществить коммит данного перемещения.

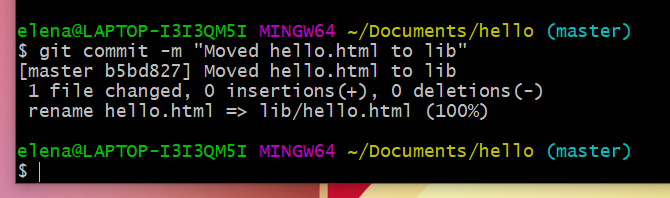


Рисунок 69 – Коммит перемещения

## Подробнее о структуре

Необходимо добавить еще один файл в репозиторий. Это будет файл index.html с кодом, показанным на рисунке 8.

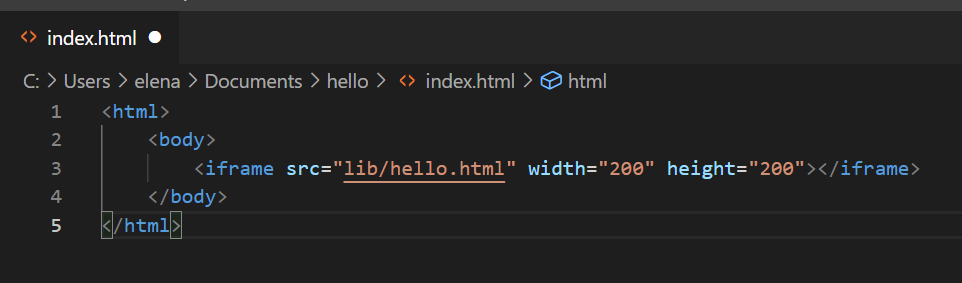
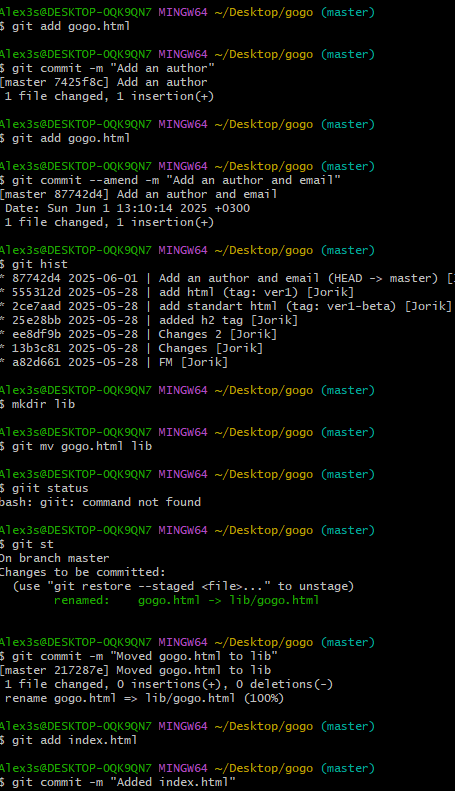


Рисунок 70 - Содержимое файла index.html

Далее нужно проиндексировать и закоммитить файл (рисунок 9). 

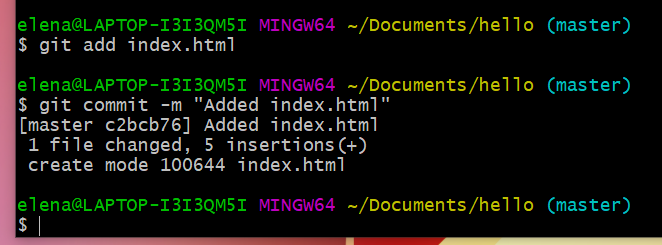


Рисунок 71 - Индексация и коммит

При открытии файла index.html будет виден кусок страницы hello.html (рисунок 10).

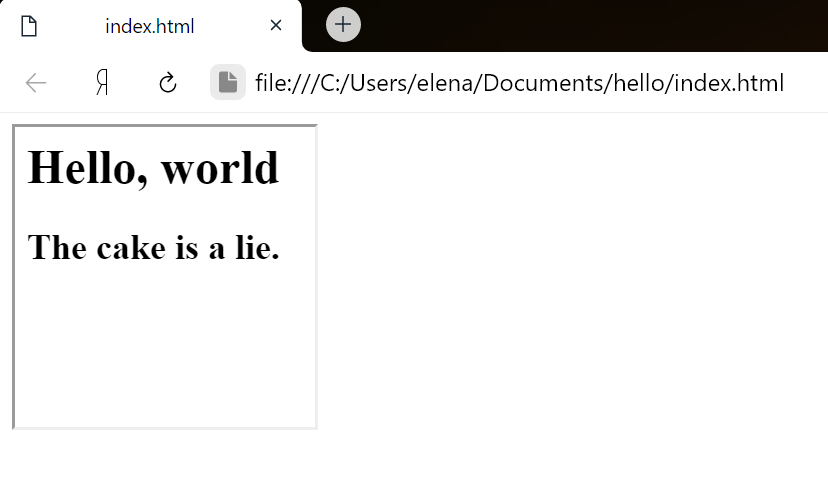


Рисунок 72 - Файл index.html, открытый в браузере

## Каталог .git

Чтобы посмотреть структуру каталога .git необходимо выполнить команду, показанную на рисунке 11.

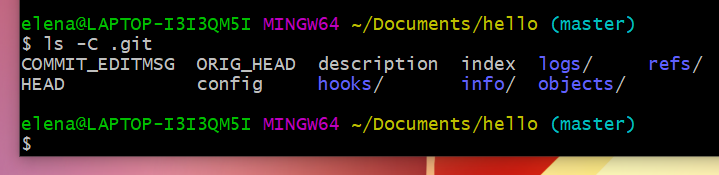


Рисунок 73 - Содержание каталога .git

При аналогичном просмотре каталога objects можно будет увидеть множество каталогов с именами из 2 символов (рисунок 12). Имена каталогов являются первыми двумя буквами хэша.

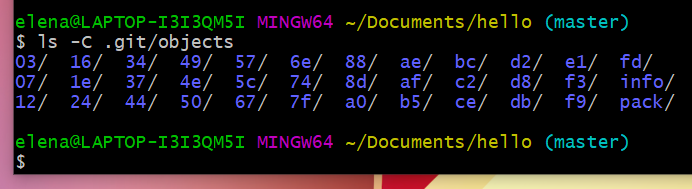


Рисунок 74 - Содержание каталога objects

При просмотре содержимого любого из каталогов будут показаны файлы, названия которых состоят из 38 символов (рисунок 13).

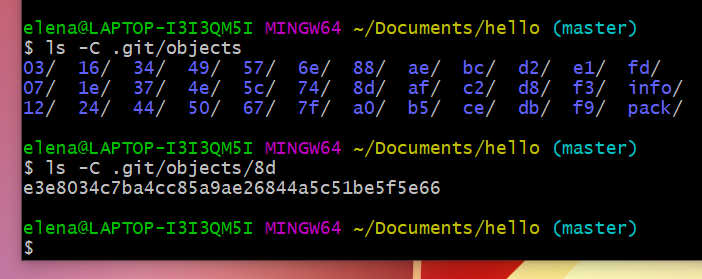


Рисунок 75 - Просмотр каталога 8d

Далее требуется просмотреть файл конфигурации с помощью команды cat (рисунок 14).

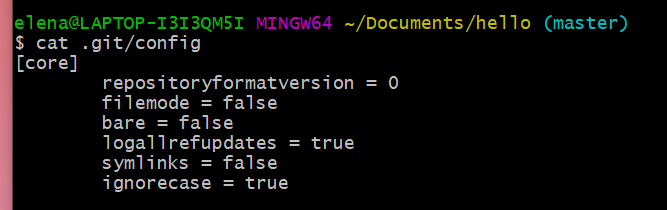


Рисунок 76 - Просмотр файла конфигурации

На рисунке 14 показан просмотр файлов в подкаталоге tags и веток в каталоге heads.

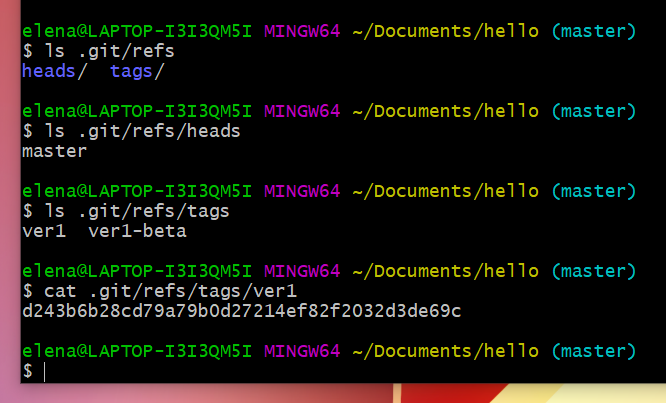


Рисунок 77 - Просмотр файлов и веток

Файл HEAD содержит ссылку на текущую ветку (рисунок 16).

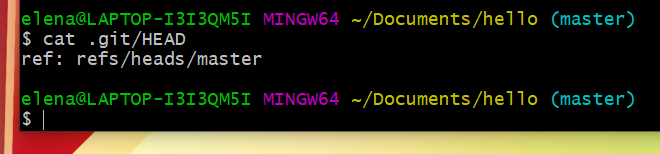


Рисунок 78 - Содержимое файла HEAD

## Работа с объектами git

Для начала необходимо просмотреть последний коммит (рисунок 17).

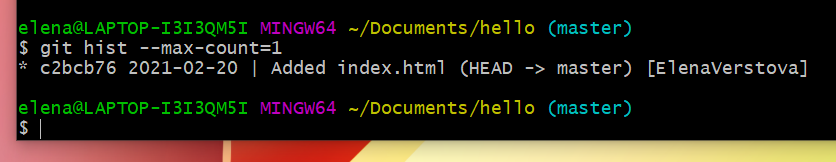


Рисунок 79 - Последний коммит

Далее надо использовать хэш последнего коммита, используя команды cat-file -p и cat-file -t (рисунок18) для просмотра объекта коммита. Также вместо длинных команд можно использовать сокращенные type и dump, если данные команды были заданы как алиасы.



Рисунок 80 - Просмотр объекта коммита

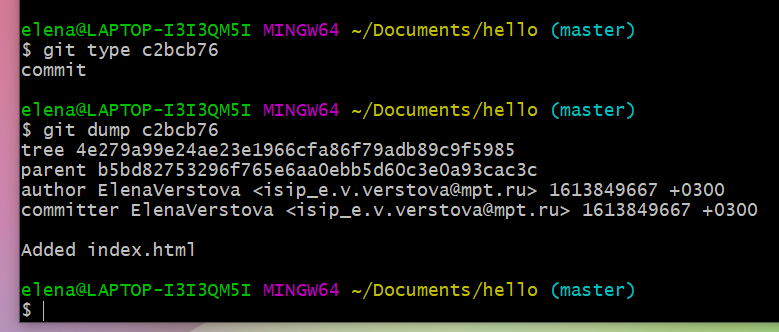


Рисунок 81 - Использование алиасов

Для просмотра дерева каталогов необходимо использовать его хэш (рисунок 20).

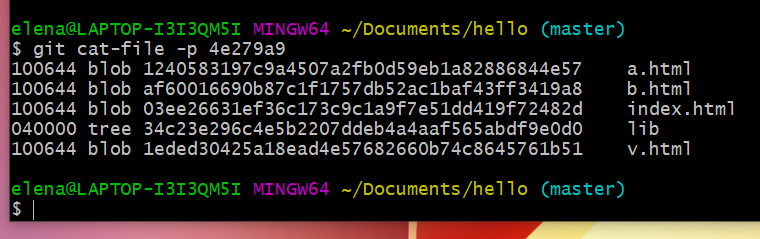


Рисунок 82 - Просмотр дерева каталогов

Затем нужно просмотреть каталог lib (рисунок 21).

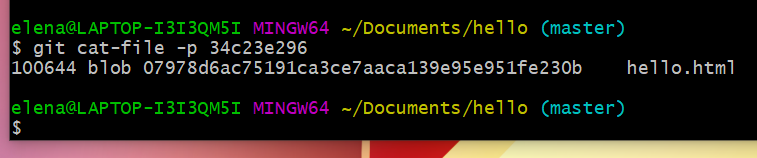


Рисунок 83 - Просмотр каталога lib

И затем требуется вывести содержимое файла hello.html (рисунок 22).

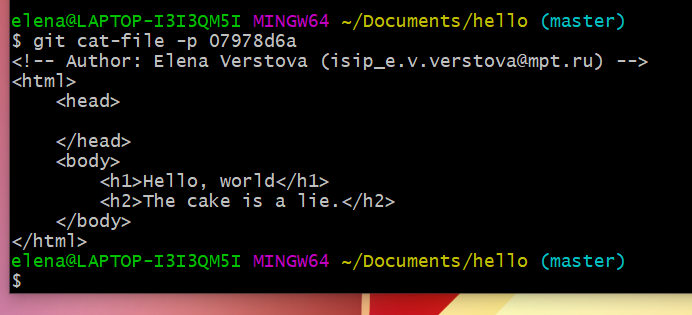


Рисунок 84 - Вывод содержимого файла hello.html

Аналогичным образом можно просмотреть содержимое файла, каким оно было в самом первом коммите, как показано на рисунке 23. Для этого требуется использовать лишь нужные хэши.

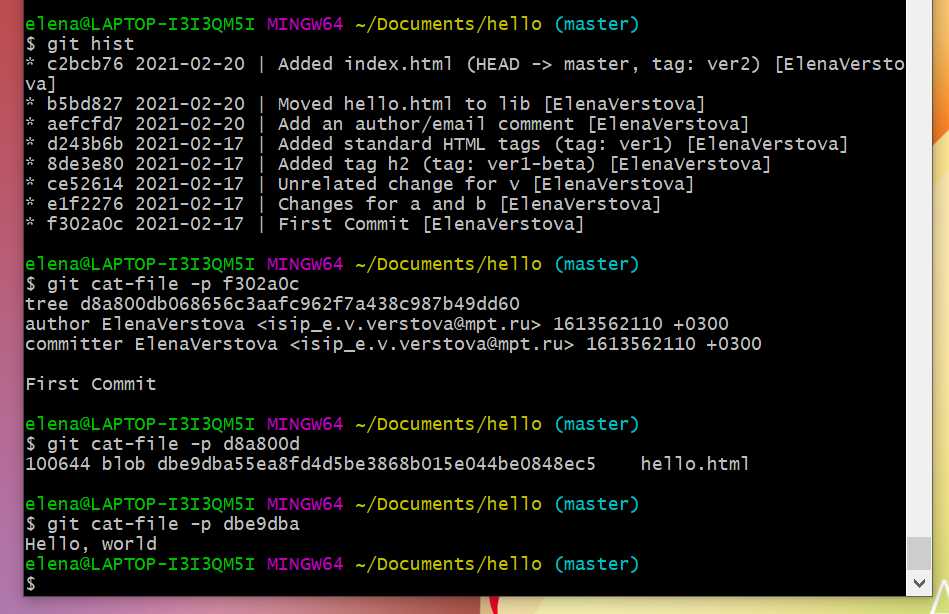


Рисунок 85 - Просмотр содержимого файла при первом коммите

Изображение выглядит как текст, снимок экрана, меню

Контент, сгенерированный ИИ, может содержать ошибки.

Изображение выглядит как текст, снимок экрана

Контент, сгенерированный ИИ, может содержать ошибки.

Изображение выглядит как текст, снимок экрана, дизайн

Контент, сгенерированный ИИ, может содержать ошибки.

Изображение выглядит как текст, снимок экрана

Контент, сгенерированный ИИ, может содержать ошибки.

# Работа с Git

## Создание ветки

Для начала необходимо создать ветку style с помощью команды git checkout -b style (рисунок 1).

Рисунок 86 - Создание новой ветки style

Затем нужно создать файл стилей (рисунок 2) и внести в него код, показанный на рисунке 3.

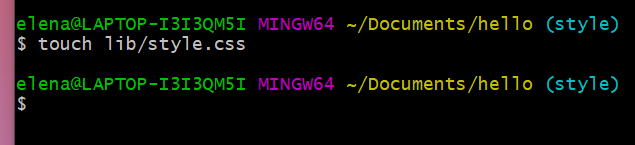


Рисунок 87 - Создание файла стилей

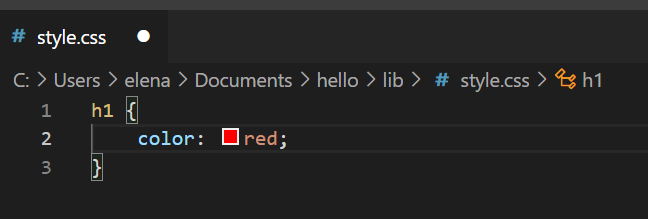
his

Рисунок 88 - Код style.css

После этого надо произвести индексацию и коммит (рисунок 4).

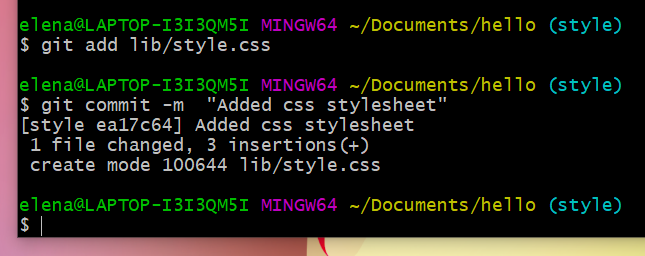


Рисунок 89 - Индексация и коммит нового файла

Далее требуется изменить основную страницу hello.html и закоммитить изменения (рисунки 5-6).

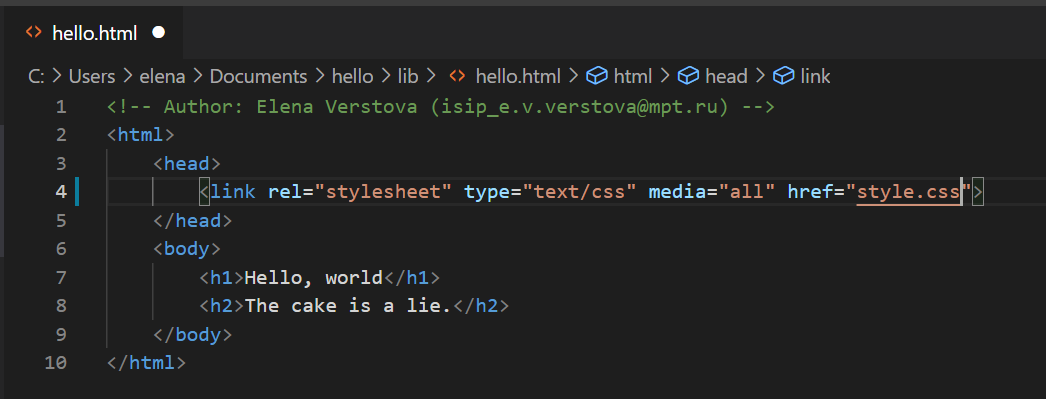


Рисунок 90 - Изменения в файле hello.html

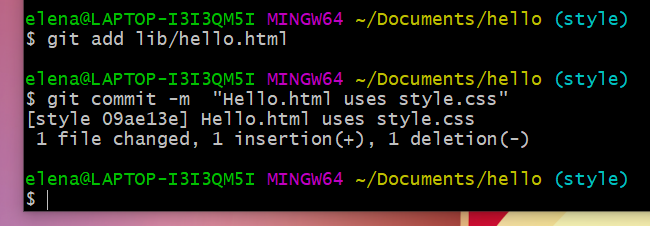


Рисунок 91 - Индексация и коммит

Далее аналогичные действия нужно осуществить с файлом index.html, как это показано на рисунках 7-8.

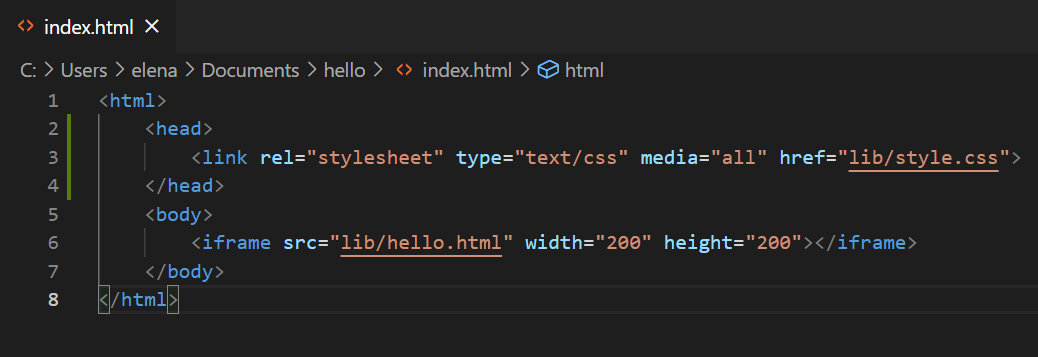


Рисунок 92 - Изменения в файле index.html

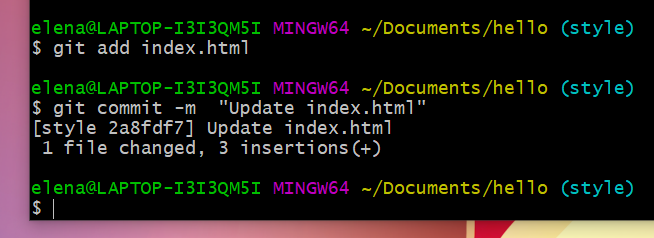


Рисунок 93 - Индексация и коммит

После выполнения предыдущих действий была создана новая ветка style с 3 коммитами.

## Навигация по веткам

При просмотре истории, как на рисунке 9, можно увидеть, что теперь в проекте 2 ветки.

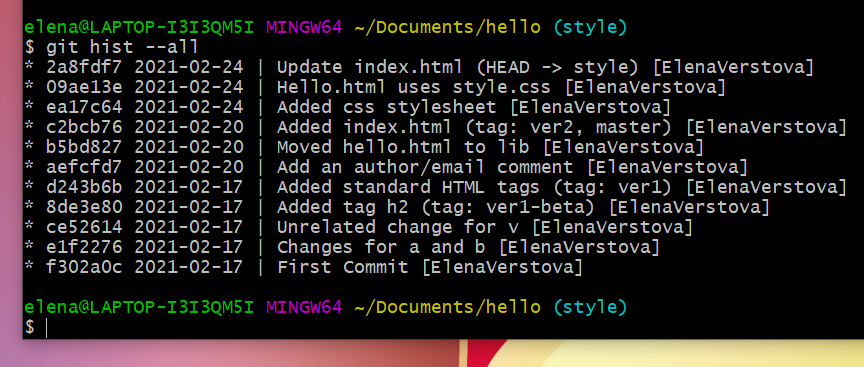


Рисунок 94 - Просмотр истории

Для переключения на ветку master используется команда git checkout master (рисунок 10). После переключения на нужную ветку при выводе файла hello.html можно увидеть, что изменения отсутствуют (по причине того, что они закоммичены в другой ветке).

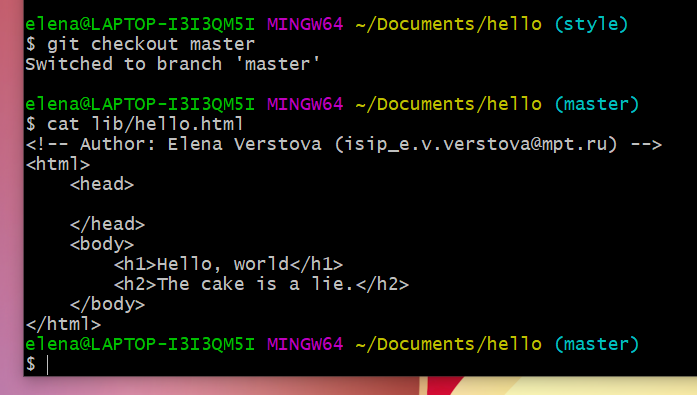


Рисунок 95 - Переключение на ветку master

При переключении на ветку style файл hello.html будет иметь другое содержание (рисунок 11).

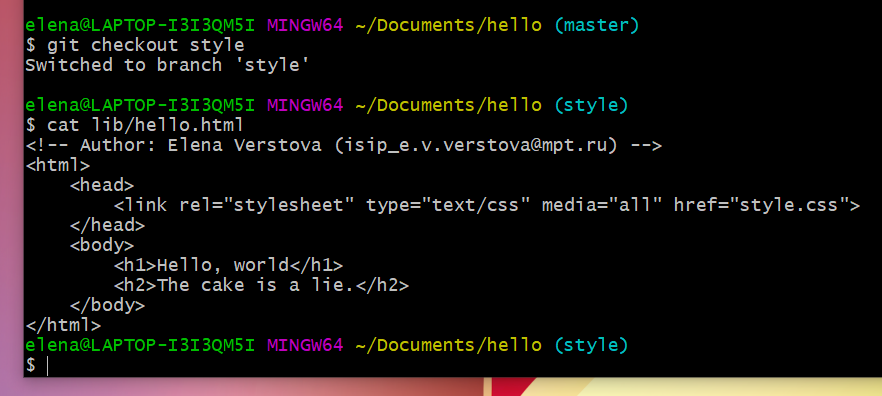


Рисунок 96 - Переключение на ветку style

## Изменения в ветке master

Необходимо переключиться на ветку master (рисунок 12) и добавить файл README (рисунки 13-14).

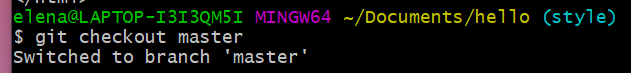


Рисунок 97 - Переключение на ветку master

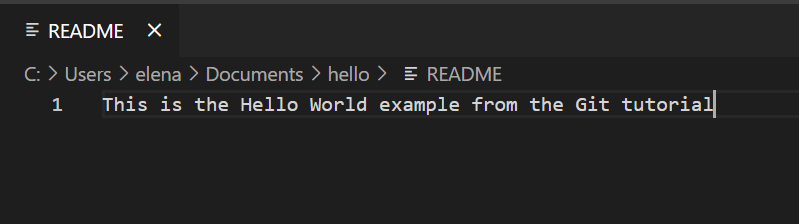


Рисунок 98 - Содержимое файла README

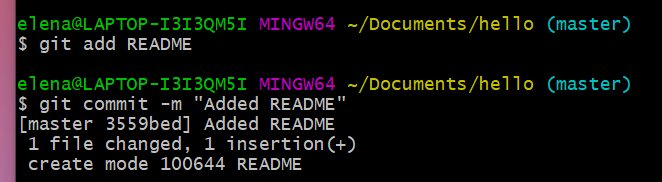


Рисунок 99 - Индексация и коммит

## Просмотр отличающихся веток

На рисунке 15 можно увидеть дерево коммитов.

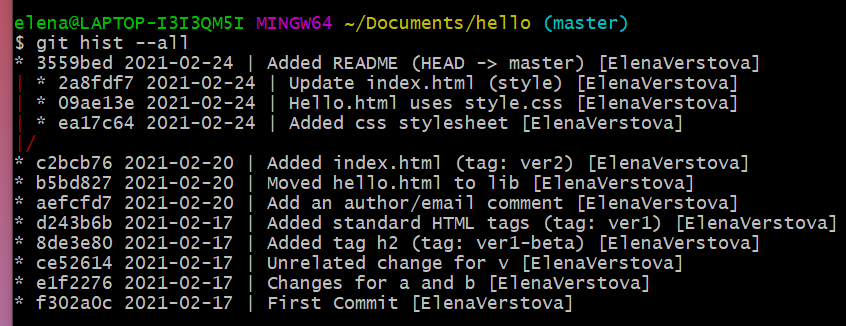


Рисунок 100 - Дерево коммитов

## Слияние

Слияние переносит изменения из двух веток в одну. Для слияния нужно перейти на ветку style и с помощью команды git merge master совместить ветки (рисунки 16-18).

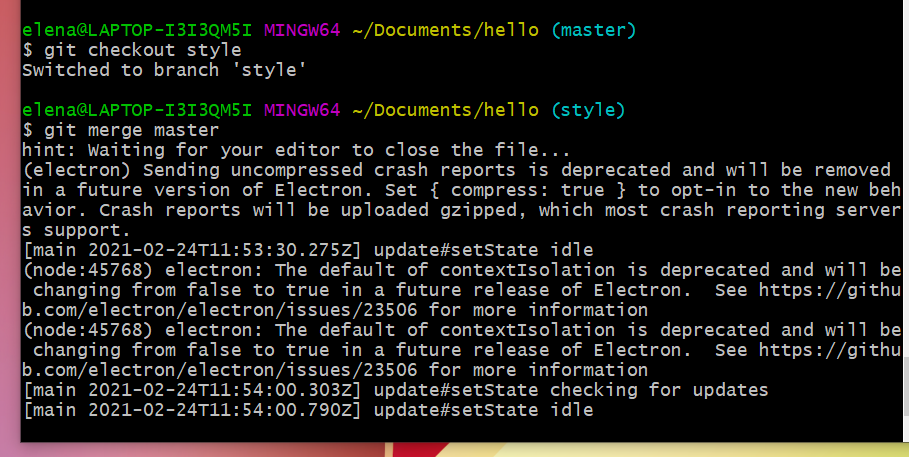


Рисунок 101 - Переключение на ветку style

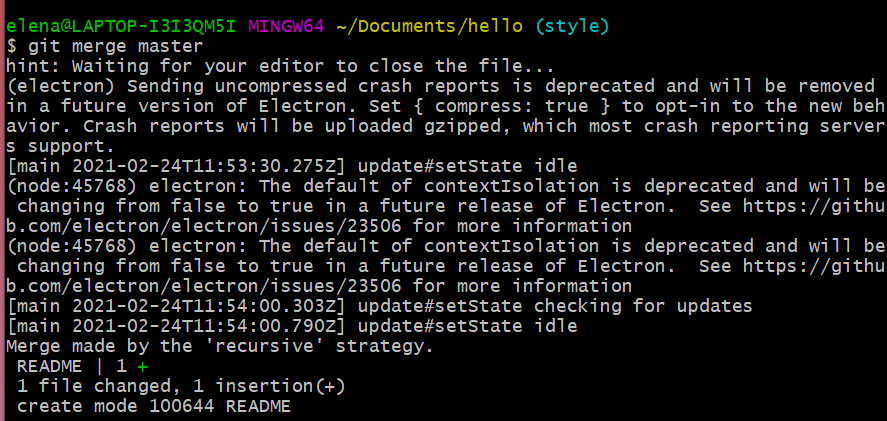


Рисунок 102 - Слияние с веткой master

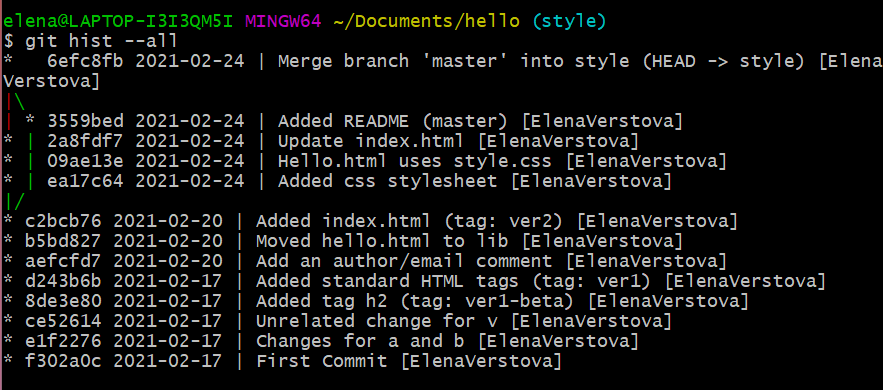


Рисунок 103 - Просмотр истории

## Создание конфликта

Для того, чтобы создать конфликт необходимо перейти в ветку master и внести изменения в файл hello.html (рисунки 19-21).

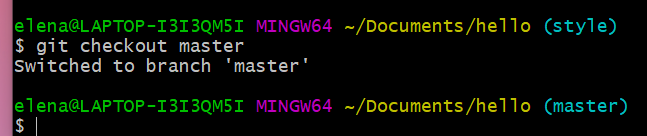


Рисунок 104 - Переход в ветку master

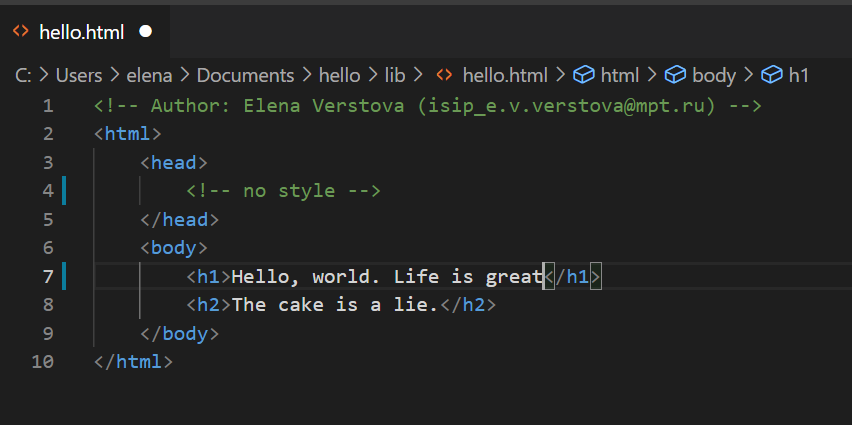


Рисунок 105 - Внесение изменений

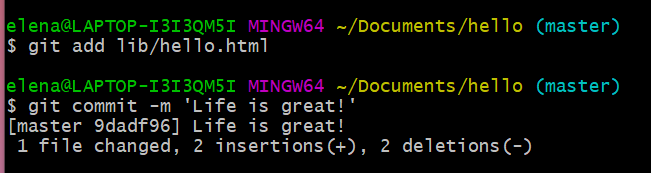


Рисунок 106 - Индексация и коммит

После выполнения предыдущих действий при просмотре веток можно будет увидеть конфликт в виде, как на рисунке 22.

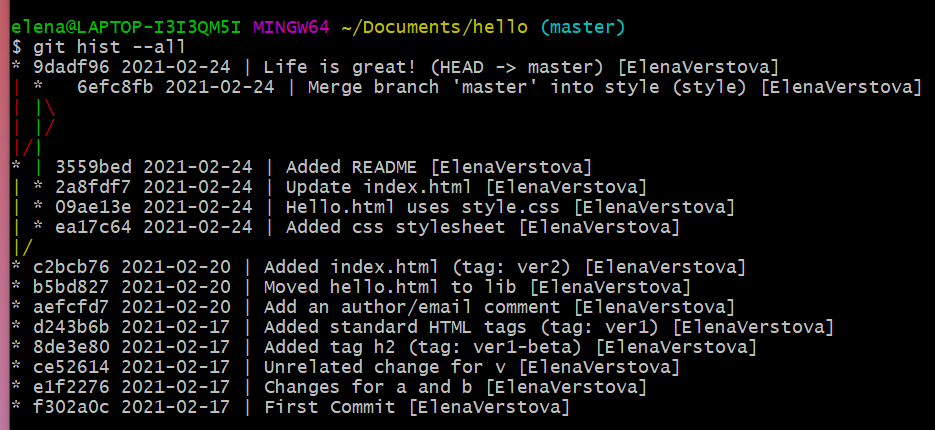


Рисунок 107 - Конфликт изменений

## Разрешение конфликтов

При попытке объединить ветку style с master будет показана ошибка из-за конфликта, как показано на рисунке 23.

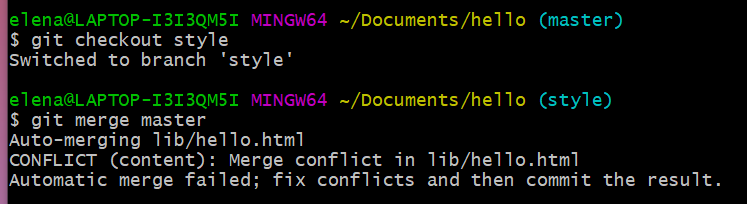


Рисунок 108 - Ошибка при слиянии

При открытии файла hello.html конфликт будет показан (рисунок 24).

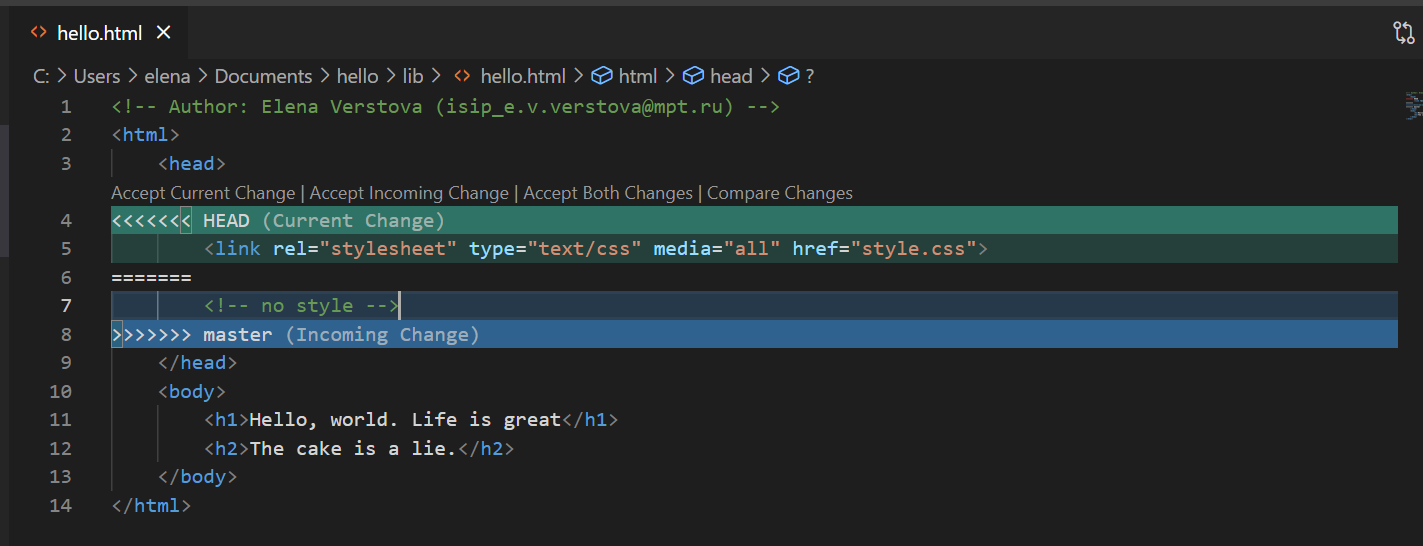


Рисунок 109 - Просмотр файла hello.html при наличии конфликта

Чтобы решить конфликт, нужно внести изменения вручную (рисунок 25).

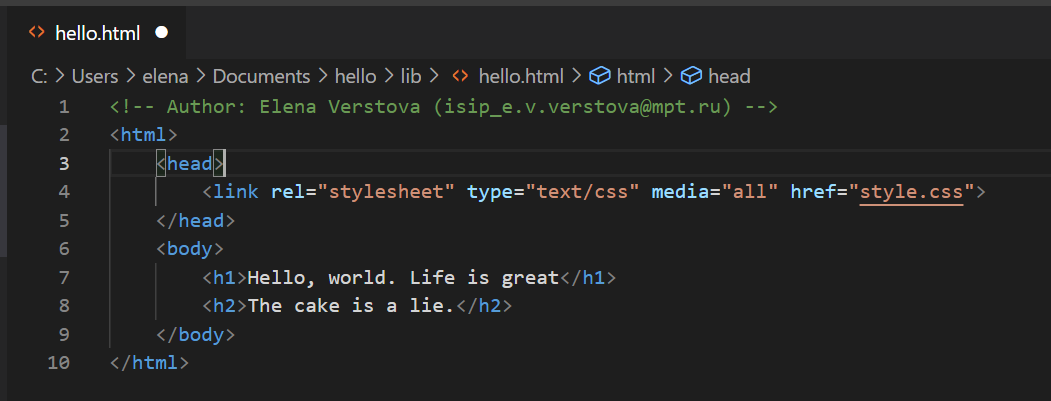


Рисунок 110 - Решение конфликта вручную

Затем следует произвести индексацию и коммит (рисунок 26).

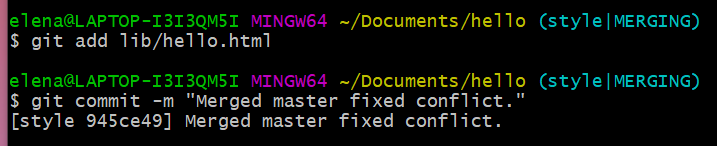


Рисунок 111 - Индексация и коммит

Изображение выглядит как текст, снимок экрана, дизайн

Контент, сгенерированный ИИ, может содержать ошибки.

Изображение выглядит как текст, снимок экрана, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.

Изображение выглядит как текст, снимок экрана, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.

# Работа с Git

## Сброс ветки style

Для сброса ветки необходимо применить команду reset --hard до требуемой точки (рисунки 1-2).

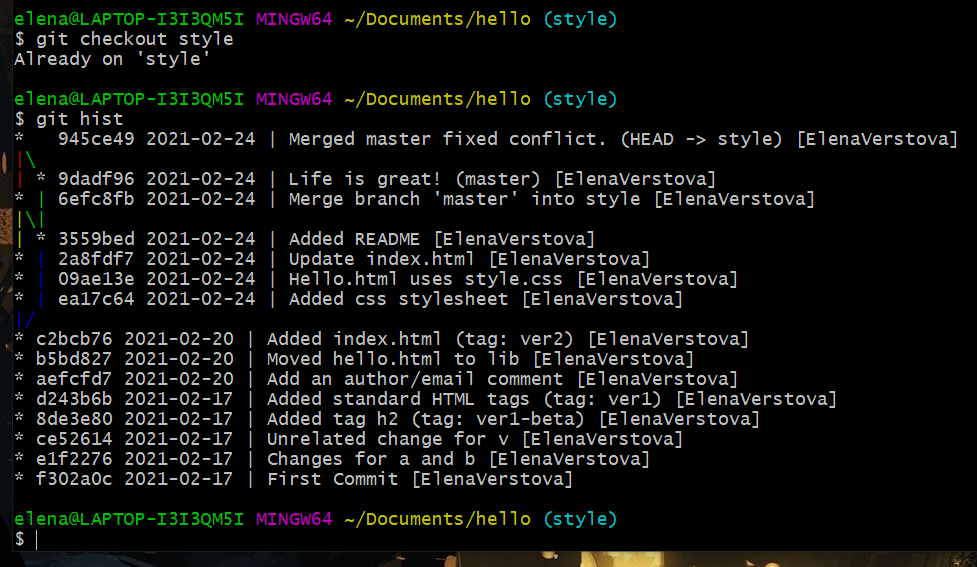


Рисунок 112 - Просмотр истории

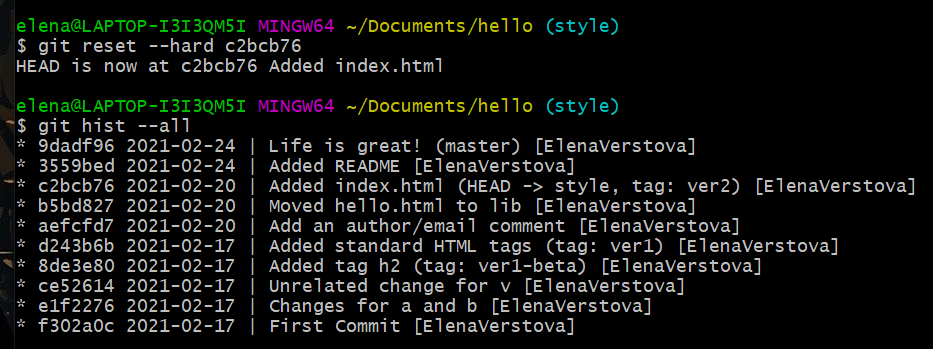


Рисунок 113 - Сброс ветки style

## Сброс ветки master

Аналогичные действия нужно произвести и для ветки master (рисунки 3-4).

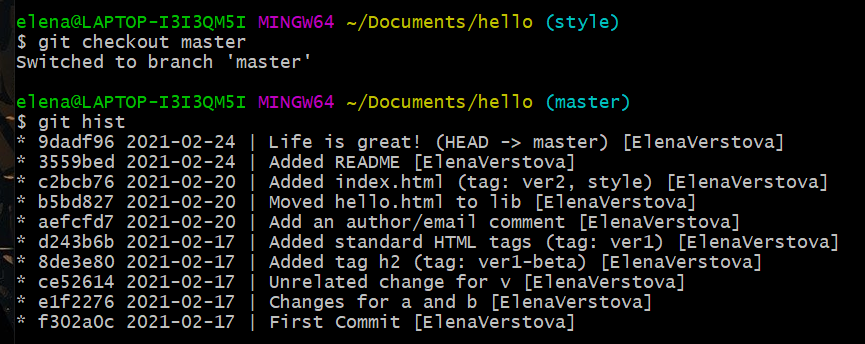


Рисунок 114 - Переключение на master и просмотр истории

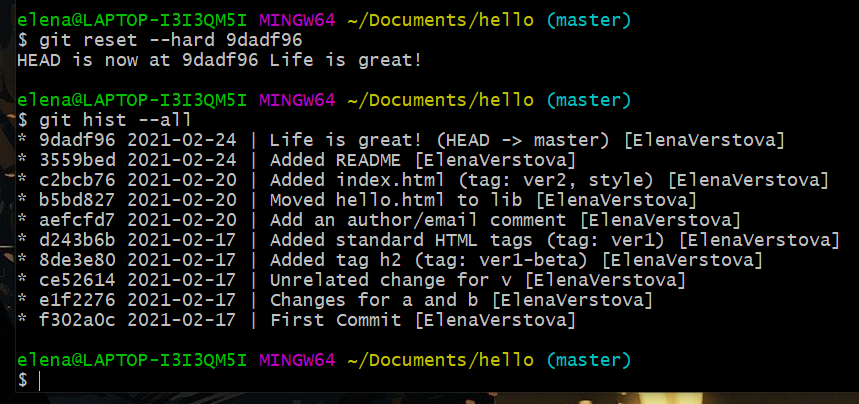


Рисунок 115 - Сброс ветки master

## Перебазирование

Команду rebase можно использовать вместо команды merge (рисунок 5).

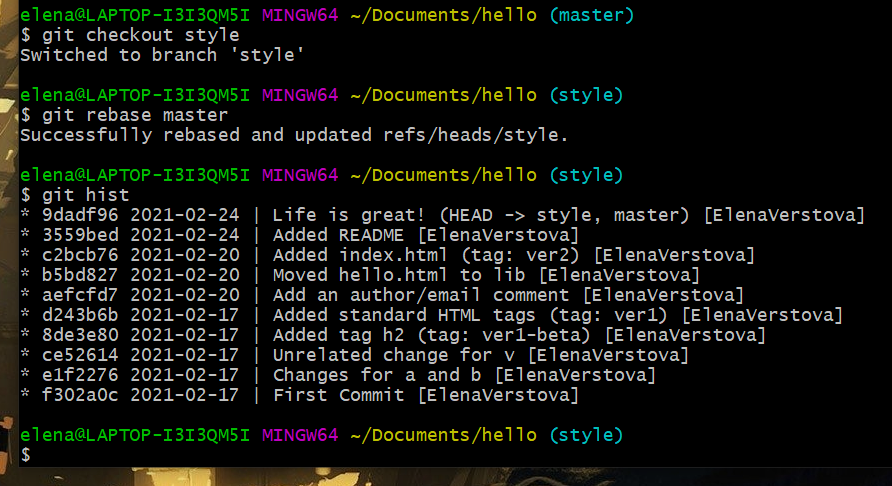


Рисунок 116 - Перебазирование веток

## Слияние в ветку master

Далее требуется произвести слияние веток с помощью merge (рисунки 6-7).

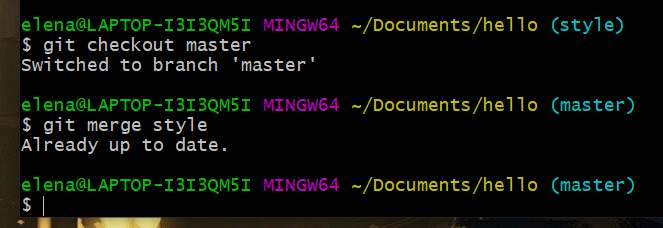


Рисунок 117 - Слияние веток

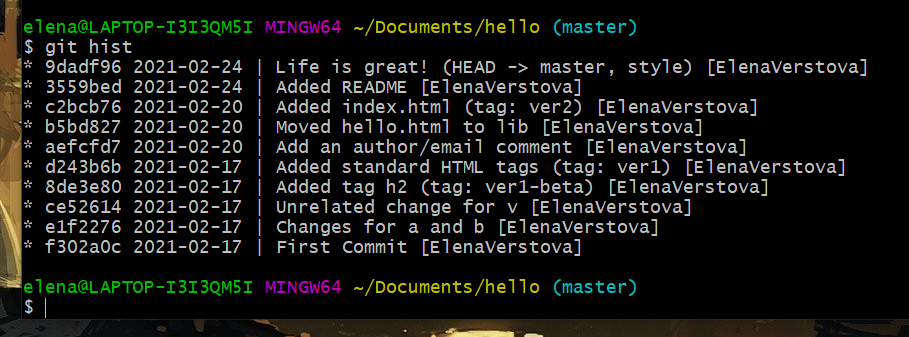


Рисунок 118 - Просмотр истории

## Клонирование репозиториев

Далее требуется научиться делать копии репозиториев. Для этого необходимо перейти в рабочий каталог и затем использовать команду git clone. Все данные действия показаны на рисунке 8.

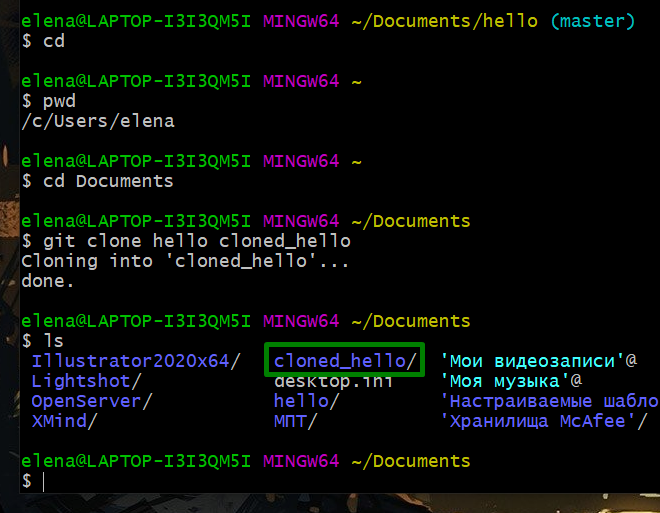


Рисунок 119 - Переход в рабочий каталог и его клонирование

## Просмотр клонированного репозитория

После этого можно просмотреть клонированный репозиторий (рисунки 9-10).

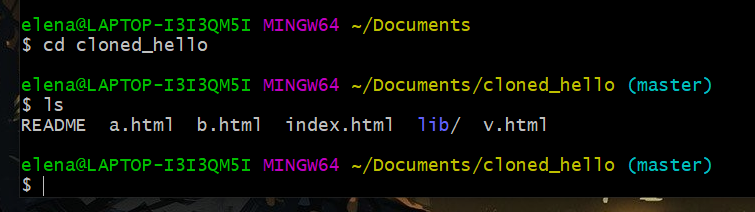


Рисунок 120 - Просмотр содержимого клонированного репозитория

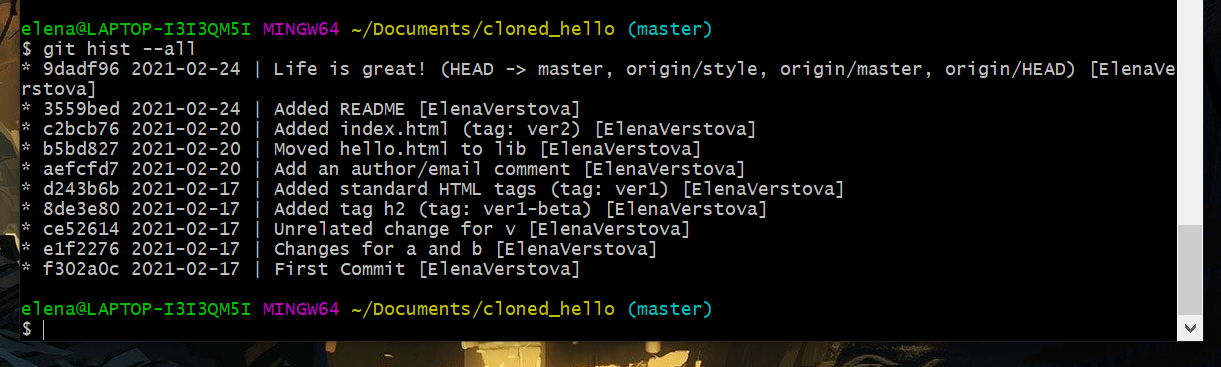


Рисунок 121 - Просмотр логов клонированного каталога

## Origin

Origin – имя по умолчанию. Просмотр данных о нем возможен с помощью команд, показанных на рисунке 11.

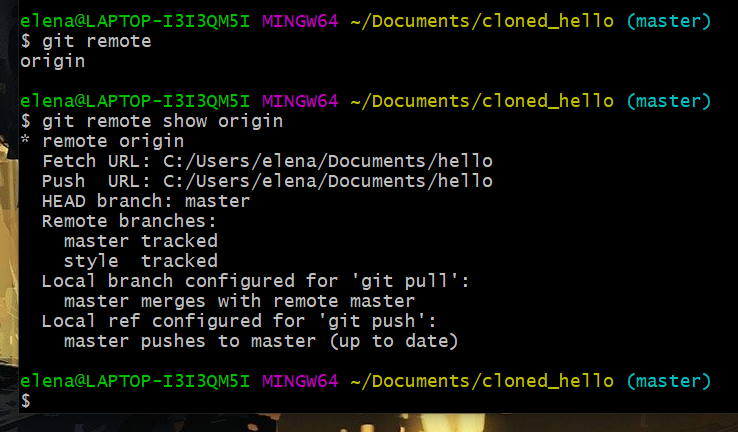


Рисунок 122 - Просмотр данных об origin

## Удаленные ветки

Для просмотра удаленных веток используется команда git branch -a (рисунок 12).

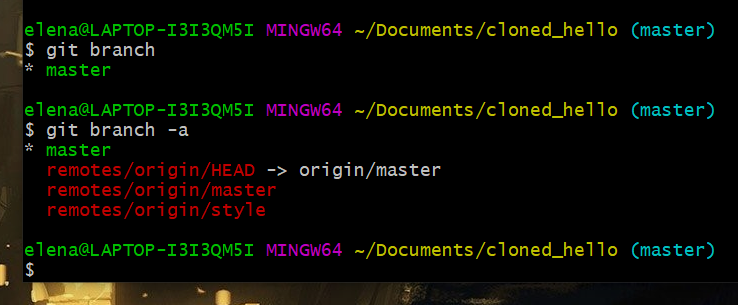


Рисунок 123 - Просмотр удаленных веток

Изображение выглядит как текст, снимок экрана

Контент, сгенерированный ИИ, может содержать ошибки.

Изображение выглядит как текст, снимок экрана, программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.clea

# Работа с Git

Сначала необходимо внести изменения в оригинальный репозиторий. Для этого нужно перейти в данный репозиторий (рисунок 1).

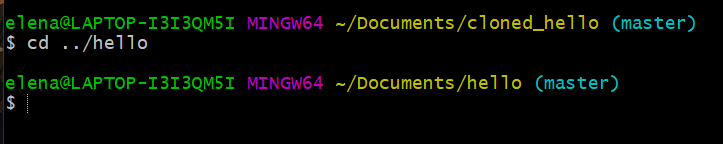


Рисунок 124 - Переход в оригинальный репозиторий

Далее надо внести изменения в файл README (рисунок 2) и затем произвести индексацию и коммит (рисунок 3).

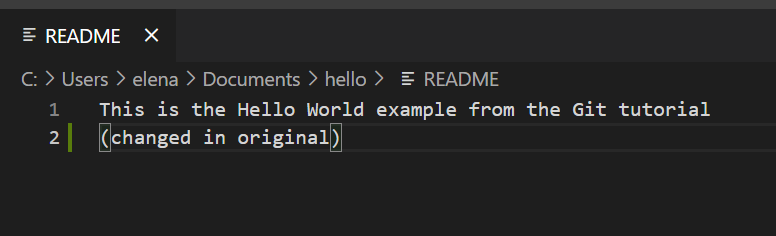


Рисунок 125 - Изменения в файле README

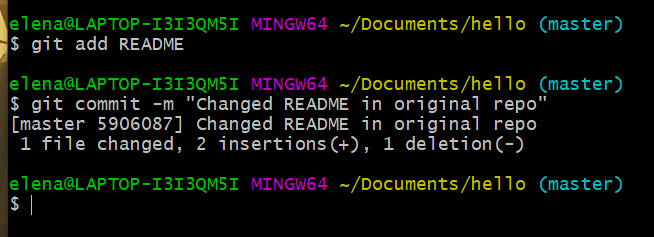


Рисунок 126 - Индексация и коммит новых изменений

Далее требуется перейти в клонированный репозиторий и извлечь изменения с помощью команды git fetch (рисунок 4) и просмотреть историю (рисунок 5).

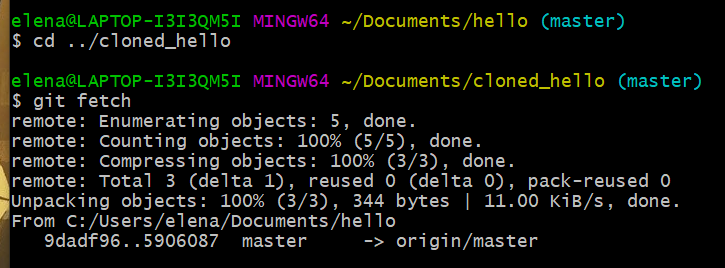


Рисунок 127 - Извлечение изменений

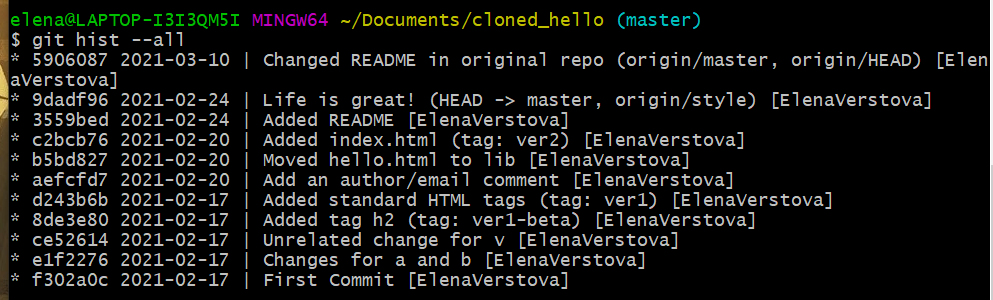


Рисунок 128 - Просмотр истории

При попытке вывести содержимое файла README можно увидеть, что изменения не были внесены (рисунок 6).

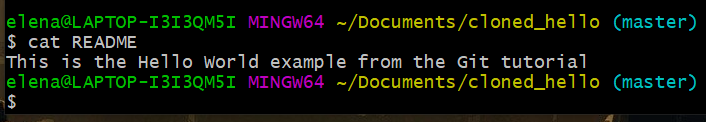


Рисунок 129 - Вывод содержимого файла README

Далее нужно слить извлеченные изменения в ветку master (рисунок 7).

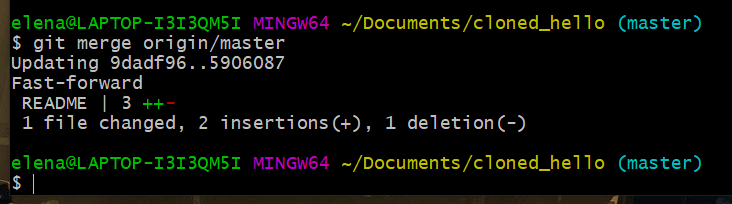


Рисунок 130 - Слияние изменений

И после выполнения предыдущего действия при выводе README можно будет увидеть последние изменения (рисунок 8).

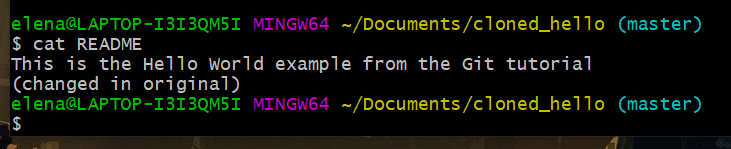


Рисунок 131 - Вывод содержимого файла README

Также существует команда, объединяющая функции git fetch и git merge, которая показана на рисунке 9.

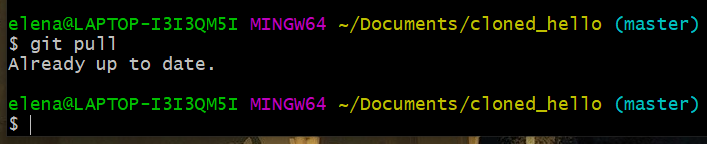


Рисунок 132 - Команда git pull

Далее требуется добавить локальную ветку, которая будет отслеживать удаленную ветку (рисунок 10).

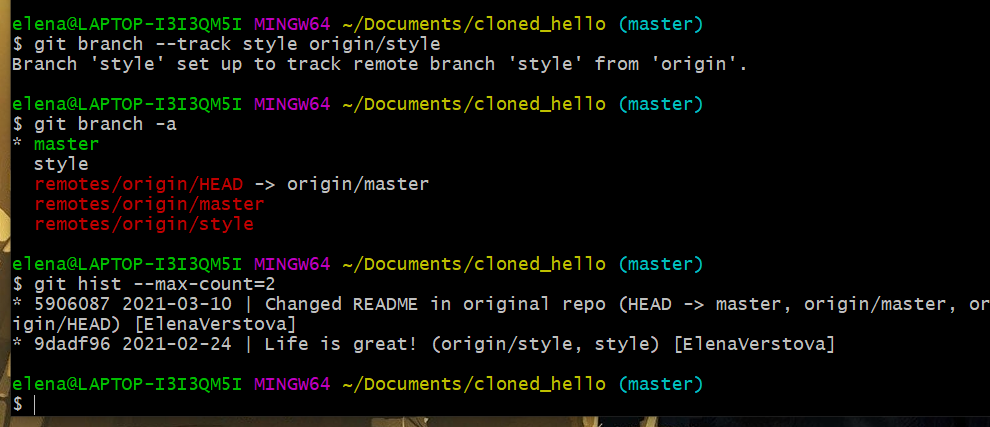


Рисунок 133 - Добавление локальной ветки

Далее необходимо создать чистый репозиторий (рисунок 11).

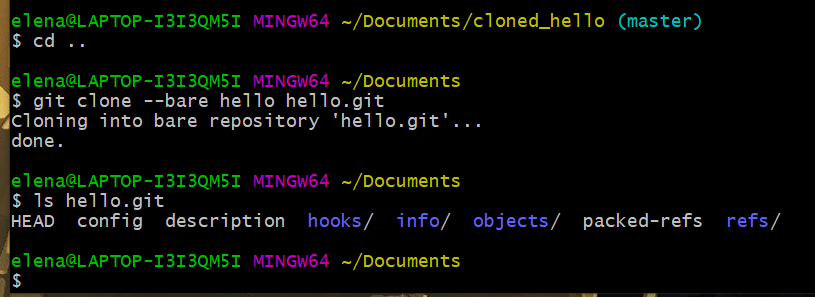


Рисунок 134 - Создание чистого репозитория

Для добавления удаленного репозитория используется команда, показанная на рисунке 12.

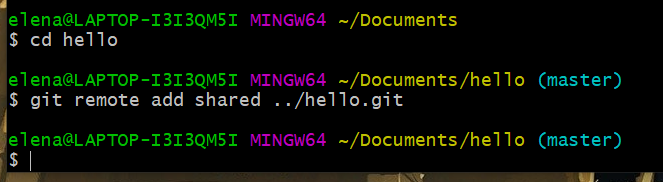


Рисунок 135 - Добавление удаленного репозитория

Затем требуется научиться отправлять изменения в удаленный репозиторий. Для этого сначала надо внести изменения, проиндексировать и произвести коммит (рисунок 13-14).

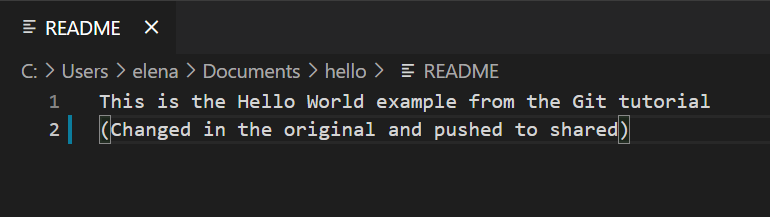


Рисунок 136 - Внесение изменений в файл

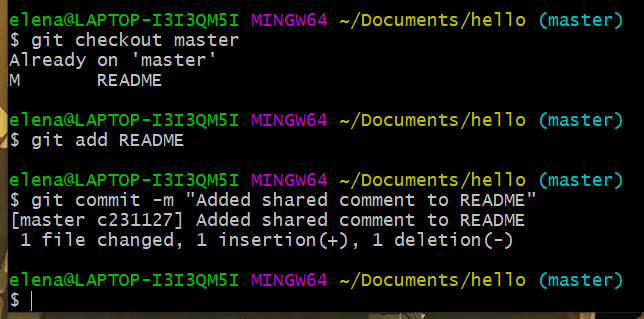


Рисунок 137 - Индексация и коммит

Далее надо отправить изменения в общий репозиторий, используя команду git push shared master (рисунок 15).

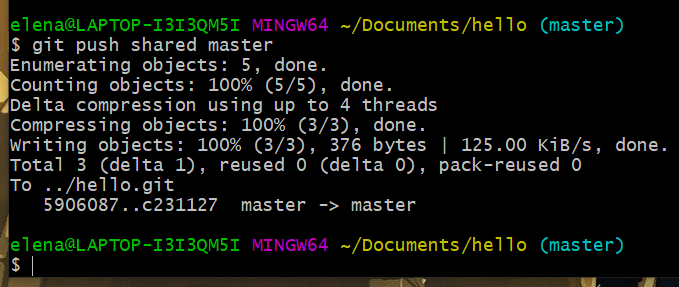


Рисунок 138 - Отправка изменений в общий репозиторий

Для извлечения общих изменений нужно перейти в клонированный каталог и выполнить перечень команд (рисунок 16).

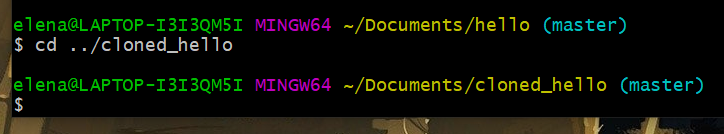


Рисунок 139 - Переход в клонированный репозиторий

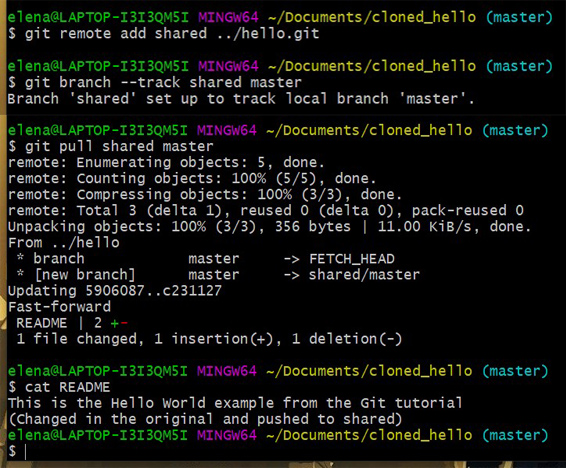


Рисунок 140 - Команды, извлекающие общие изменения

Для настройки git сервера нужно выполнить команду, показанную на рисунке 18. Затем в другом окне можно проверить работу сервера, сделав копию проекта hello (рисунок 18).

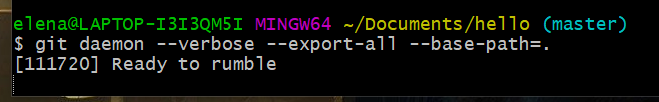


Рисунок 141 - Настройка сервера

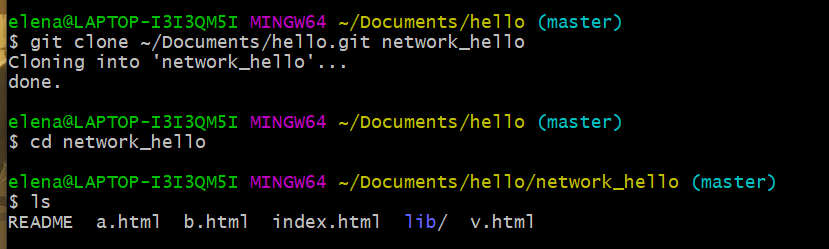


Рисунок 142 - Клонирование проекта

Изображение выглядит как текст, снимок экрана

Контент, сгенерированный ИИ, может содержать ошибки.

Изображение выглядит как текст, снимок экрана, дизайн

Контент, сгенерированный ИИ, может содержать ошибки.

Изображение выглядит как текст, снимок экрана, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.