Лабораторная работа №1

Математическое моделированеи

Александрова Ульяна Вадимовна

Содержание

# 1 Цель работы

Целью данное работы является изучение и освоение утилиты Git.

# 2 Теоретическое введение

Git — это распределённая система управления версиями, которая позволяет отслеживать изменения в файлах и координировать работу нескольких людей над проектом. Она была разработана Линусом Торвальдсом в 2005 году для управления разработкой ядра Linux.

Основные особенности Git:

1. **Распределённость**: Каждый разработчик имеет полную копию репозитория, включая всю историю изменений. Это позволяет работать автономно и синхронизировать изменения позже.
2. **Быстрота**: Git оптимизирован для высокой скорости выполнения операций.
3. **Эффективность хранения**: Git использует дельты и сжатие для эффективного хранения данных.
4. **Поддержка ветвления и слияния**: Git позволяет легко создавать и объединять ветки, что упрощает работу над новыми функциями и исправлением ошибок.

Основные команды Git:

* git init: Инициализация нового репозитория.
* git clone: Клонирование существующего репозитория.
* git add: Добавление изменений в индекс (стейджинг).
* git commit: Фиксация изменений в репозитории.
* git push: Отправка изменений в удалённый репозиторий.
* git pull: Получение изменений из удалённого репозитория и их слияние с локальной копией.

# 3 Выполнение лабораторной работы

## 3.1 Работа с git

### 3.1.1 1.1.1 Установка имени и электронной почты

У меня уже был подключен гитхаб на этой машине при помощи следующих команд:

git config --global user.name "Your Name"  
git config --global user.email "your\_email@whatever.com"

Как мы можем видеть, git работает успешно (рис. 1).



Рис. 1: Проверка работы утилиты на существующих папках

### 3.1.2 1.1.3 Установка отображения unicode

По умолчанию, git будет печатать не-ASCII символы в именах файлов в виде восьмеричных последовательностей \nnn. Чтобы избежать нечитаемых строк, я установила соответствующий флаг.

git config --global core.quotepath off

### 3.1.3 1.2.1 Создайте страницу «Hello, World»

Начну работу в пустом рабочем каталоге с создания пустого каталога с именем hello, затем войду в него и создам там файл с именем hello.html (рис. 2) (рис. 3).

echo "Hello, World!" > hello.html

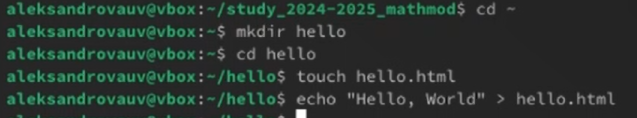


Рис. 2: Создание файла

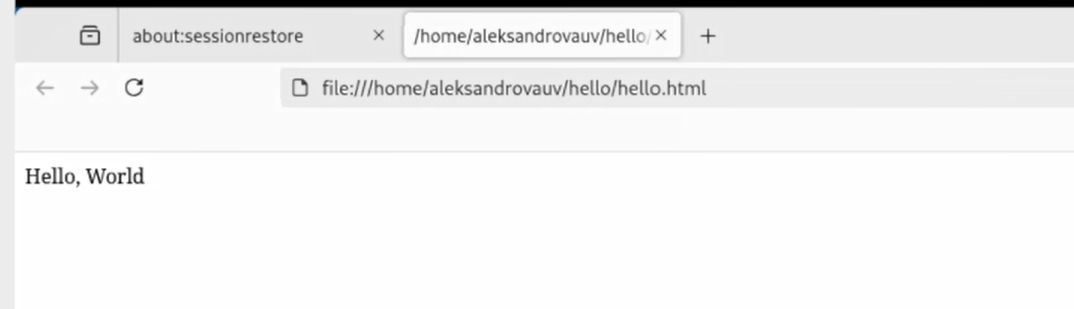


Рис. 3: Рабочий файл

### 3.1.4 1.2.2 Создание репозитория

Чтобы создать git репозиторий из этого каталога, выполняю команду git init (рис. 4).

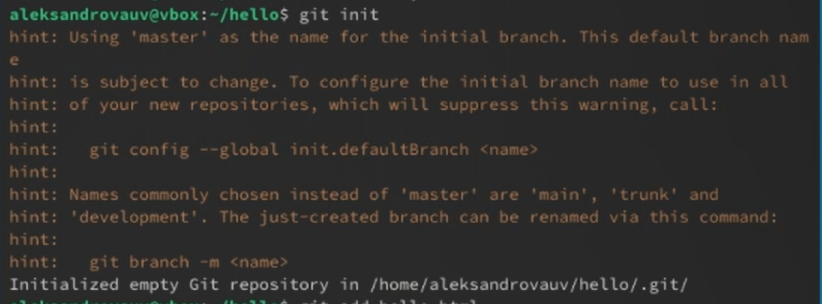


Рис. 4: Создание репозитория

### 3.1.5 1.2.3 Добавление файла в репозиторий и проверка состояния

Добавляю файл в репозиторий.

Используякоманду git status, проверияю текущее состояние репозитория.

git status

Коммитить нечего. Это означает, что в репозитории хранится текущее состояние рабочего каталога, и нет никаких изменений, ожидающих записи (рис. 5).

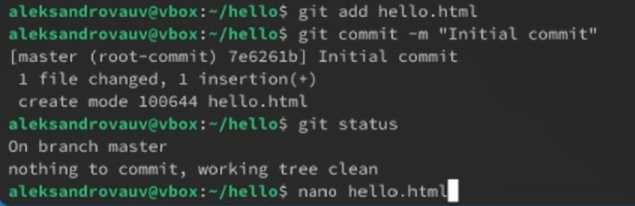


Рис. 5: Добавление файла в репозиторий и проверка состояния

Вот текст страниц 3-5 вашего PDF-документа в формате Markdown:

## 3.2 1.3 Внесение изменений

### 3.2.1 1.3.1 «Hello, World»

Добавляю HTML-теги к нашему приветствию. Изменяю содержимое файла hello.html на (рис. 6).

<h1>Hello, World!</h1>



Рис. 6: Работа файла

Проверяю состояние рабочего каталога.

## 3.3 1.4 Индексация изменений

Теперь выполняю команду git add, чтобы проиндексировать изменения (рис. 7).

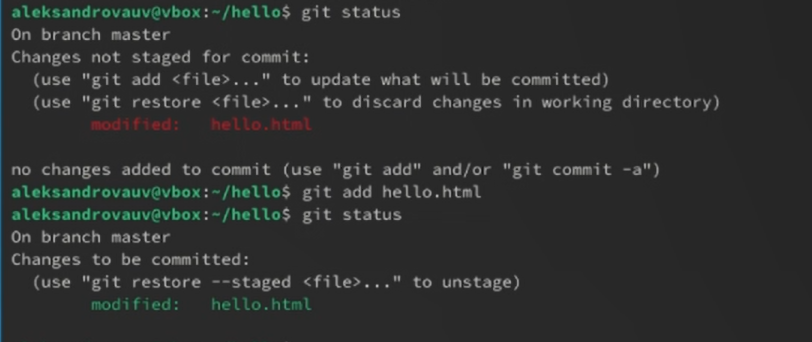


Рис. 7: Индексация

Изменения файла hello.html были проиндексированы. Это означает, что Git теперь знает об изменении, но изменение пока не записано в репозиторий. Следующий коммит будет включать в себя проиндексированные изменения.

### 3.3.1 1.4.1 Коммит изменений

Делаю коммит и проверяю состояние (рис. 8).

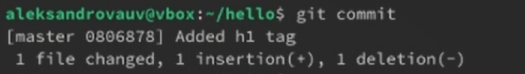


Рис. 8: Коммит

Рабочий каталог чистый, можно продолжить работу.

### 3.3.2 1.4.2 Добавление стандартных тегов страницы

Изменяю страницу «Hello, World», чтобы она содержала стандартные теги <html> и <body> (рис. 9).

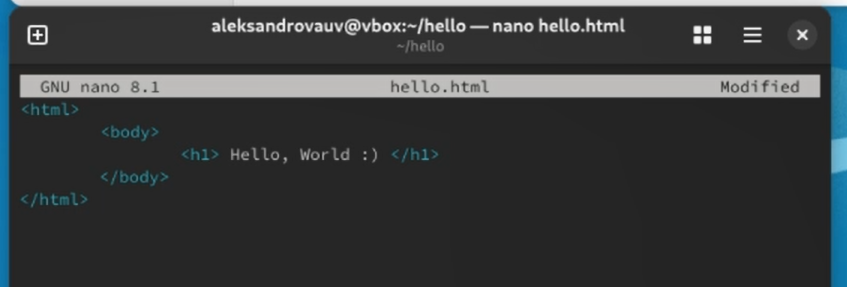


Рис. 9: Изменение файла

Теперь добавлю это изменение в индекс Git.

git add hello.html

Теперь добавляю заголовки HTML к странице «Hello, World» (рис. 10).

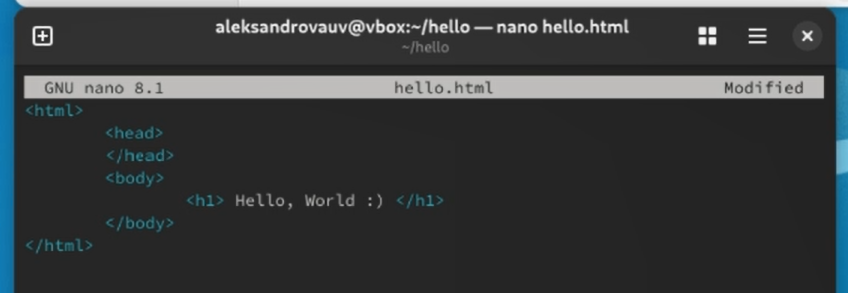


Рис. 10: Изменение файла

Проверяю текущий статус (рис. 11).



Рис. 11: Статус

hello.html указан дважды в состоянии. Первое изменение (добавление стандартных тегов) проиндексировано и готово к коммиту. Второе изменение (добавление заголовков HTML) является непроиндексированным. Если бы я сделала коммит сейчас, заголовки не были бы сохранены в репозиторий.

Коммичу проиндексированное изменение, а затем еще раз проверяю состояние (рис. 12).

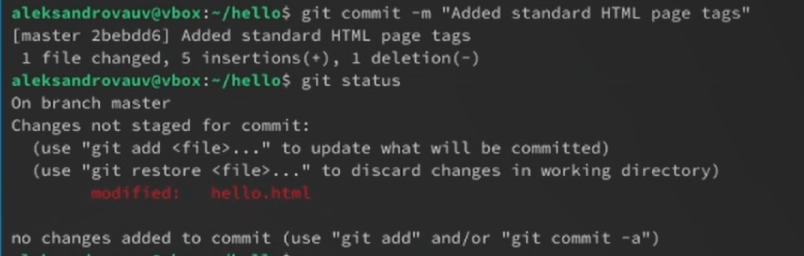


Рис. 12: Коммит

Состояние команды говорит о том, что hello.html имеет незафиксированные изменения, но уже не в буферной зоне.

### 3.3.3 1.4.3 История

Получим список произведенных изменений (рис. 13).

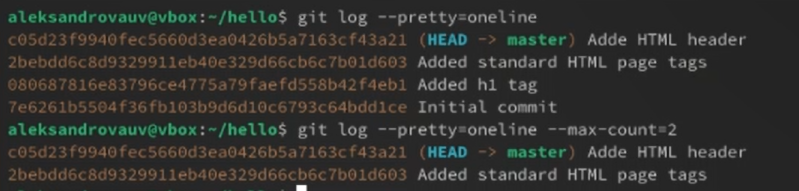


Рис. 13: Лог

### 3.3.4 1.4.4 Получение старых версий

Возвращаться назад в историю очень просто. Команда checkout скопирует любой снимок из репозитория в рабочий каталог (рис. 14).

Возврашение к предыдущей версии

Рис. 14: Возврашение к предыдущей версии

Просматриваю файл hello.html той версии и возвращаюсь к последней версии в ветке master (рис. 15).

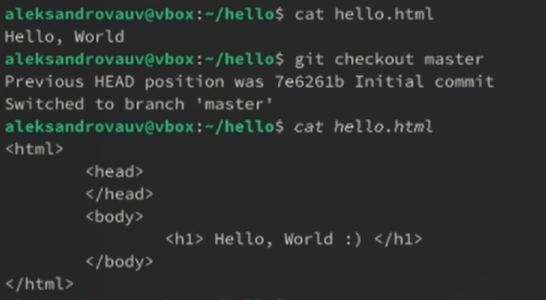


Рис. 15: Переход между версиями

### 3.3.5 1.4.5 Создание тегов версий

Называю текущую версию страницы hello первой (v1) (рис. 16).



Рис. 16: Тег ветки

Теперь текущая версия страницы называется v1.

#### 3.3.5.1 Теги для предыдущих версий и переключение между версиями

Создаю тег для предыдущей версии через команду:

git checkout v1^  
cat hello.html

Это версия с тегами <html> и <body>, но еще пока без <head>. Делаю ее версией v1-beta.

git tag v1-beta

Теперь пробую попереключаться между двумя отмеченными версиями (рис. 17).

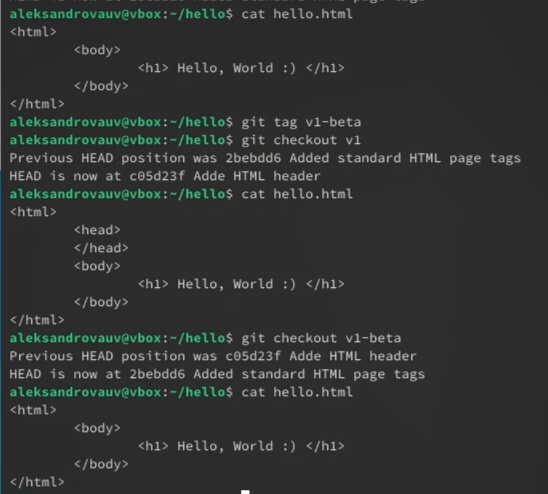


Рис. 17: Переключение между версиями

### 3.3.6 1.4.7 Просмотр тегов с помощью команды tag

Просматриваю теги (рис. 18).

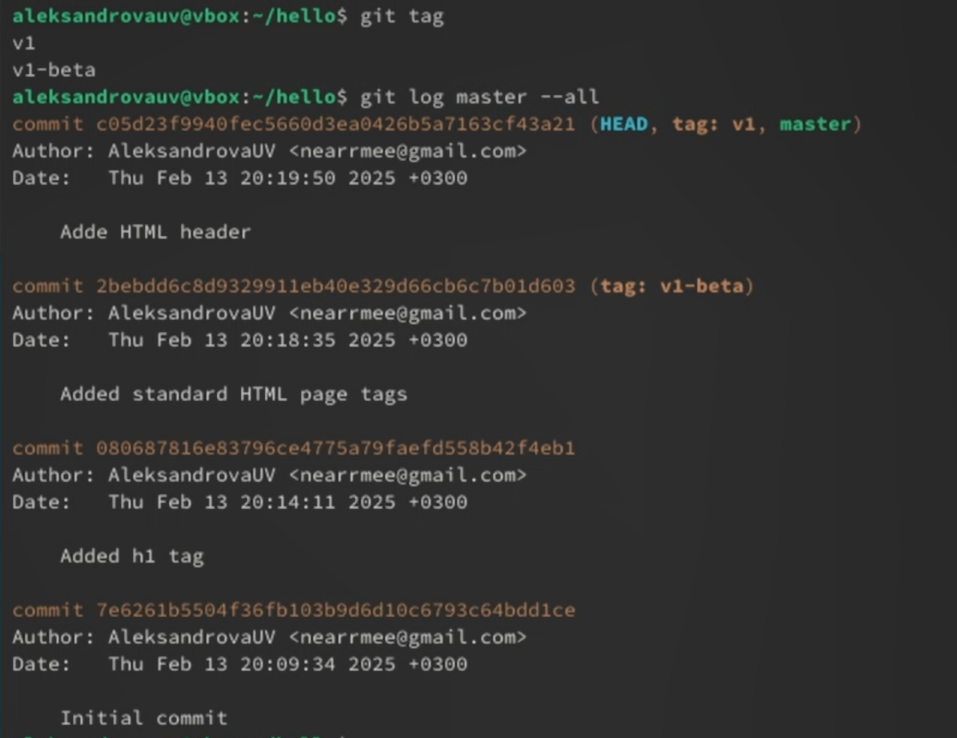


Рис. 18: Просмотр тегов

Мы можем видеть теги (v1 и v1-beta) в логе вместе с именем ветки (master). Кроме того, HEAD показывает коммит, на который мы переключились (на данный момент это v1-beta).

## 3.4 1.5 Отмена локальных изменений (до индексации)

### 3.4.1 1.5.1 Переключение на ветку master и изменение hello.html

Иногда случается, что мы изменили файл в рабочем каталоге, и хотим отменить последние коммиты (рис. 19).

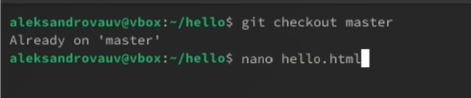


Рис. 19: Переключение

Вношу изменение в файл hello.html в виде нежелательного комментария.

<html>  
 <head></head>  
 <body>  
 <h1>Hello, World!</h1>  
 <!-- This is a bad comment. We want to revert it. -->  
 </body>  
</html>

## 3.5 1.6 Отмена проиндексированных изменений (перед коммитом)

### 3.5.1 1.6.1 Изменение файла

Вношу изменение в файл hello.html в виде нежелательного комментария.

<html>  
 <head>  
 <!-- This is an unwanted but staged comment -->  
 </head>  
 <body>  
 <h1>Hello, World!</h1>  
 </body>  
</html>

Проиндексирую это изменение.

git add hello.html

### 3.5.2 1.6.3 Сброс буферной зоны

Команда git reset сбрасывает буферную зону к HEAD. Это очищает буферную зону от изменений, которые мы только что проиндексировали. Команда git reset (по умолчанию) не изменяет рабочий каталог. Поэтому рабочий каталог все еще содержит нежелательный комментарий (рис. 20).

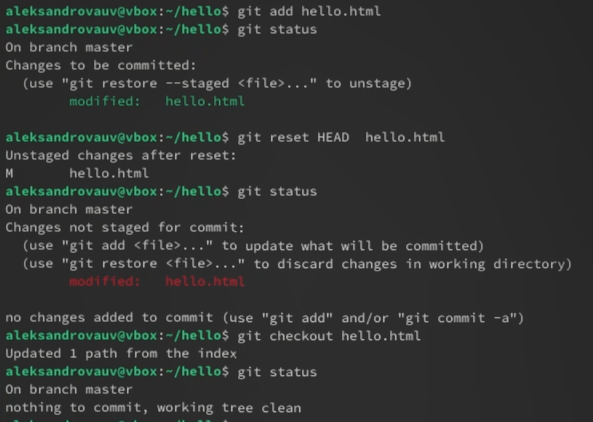


Рис. 20: Сброс буферной зоны

## 3.6 1.7 Отмена коммитов

### 3.6.1 1.7.1 Отмена коммитов

Изменяю файл hello.html на следующий.

<html>  
 <head></head>  
 <body>  
 <h1>Hello, World!</h1>  
 <!-- This is an unwanted but committed change -->  
 </body>  
</html>

Произвожу коммит и отменяю через git revert HEAD и просматриваю лог (рис. 21) (рис. 22).

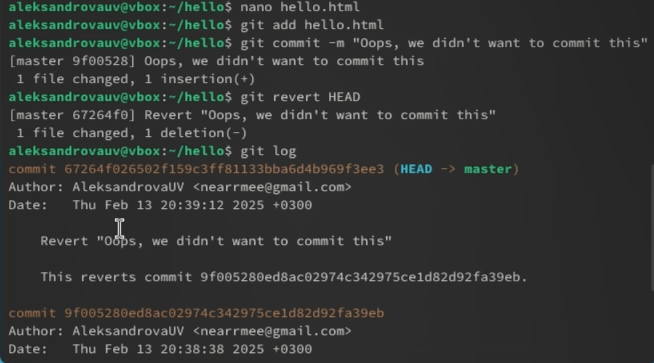


Рис. 21: Отмена коммита

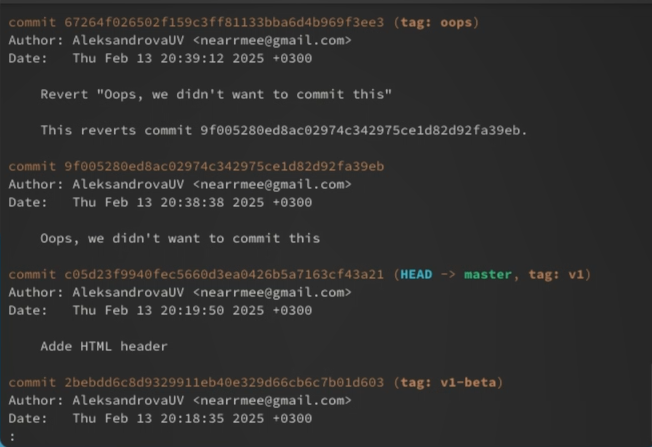


Рис. 22: Полный лог

## 3.7 1.9 Удаление тега oops

### 3.7.1 1.9.1 Удаление тега oops

Тег oops свою функцию выполнил. Удалю его и коммиты, на которые он ссылался, сборщиком мусора.

git tag -d oops  
git log --all

Тег «oops» больше не будет отображаться в репозитории (рис. 23).

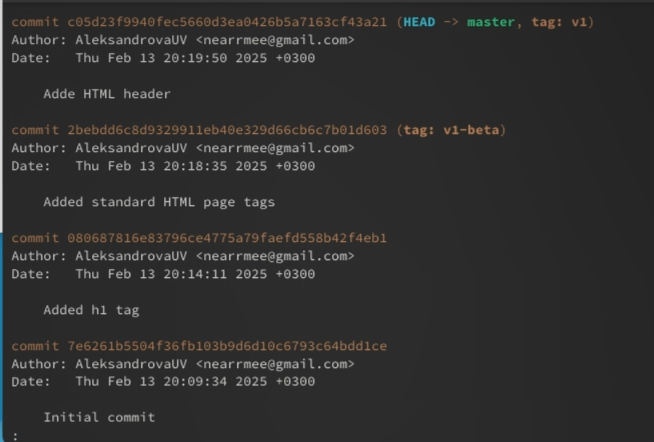


Рис. 23: Удаление тега

## 3.8 1.10 Внесение изменений в коммиты

### 3.8.1 1.10.1 Изменение страницы

Добавляю в страницу комментарий автора (рис. 24).

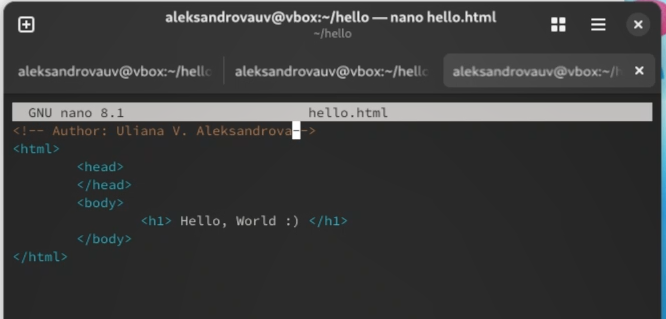


Рис. 24: Добавление авторства

Выполняю:

git add hello.html  
git commit -m "Add an author comment"

### 3.8.2 1.10.2 Необходим email

Добавляю email (рис. 25).

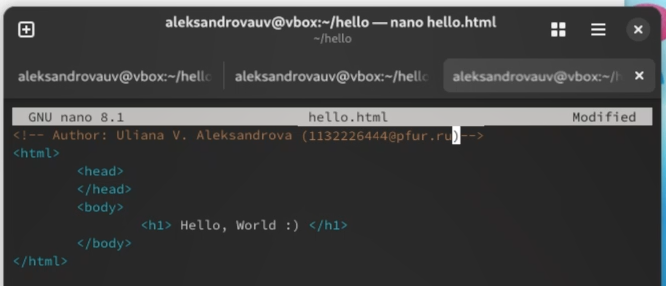


Рис. 25: Добавление почты

### 3.8.3 1.10.3 Измените предыдущий коммит

Коммичу (рис. 26).

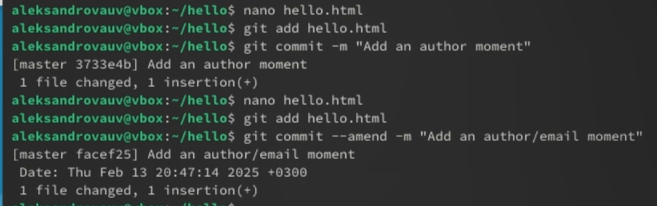


Рис. 26: Изменение предыдущего коммита

### 3.8.4 1.10.4 Просмотр истории

Выполняю (рис. 23).

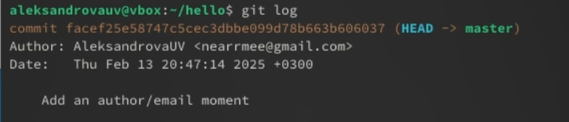


Рис. 27: Лог после изменения коммита

Мы можем увидеть, что оригинальный коммит «автор» заменен коммитом «автор/email». Этого же эффекта можно достичь путем сброса последнего коммита в ветке, и повторного коммита новых изменений.

## 3.9 1.11 Перемещение файлов

### 3.9.1 1.11.1 Перемещение в каталог lib

Создаю новый репозиторий и коммичу изменения (рис. 28).

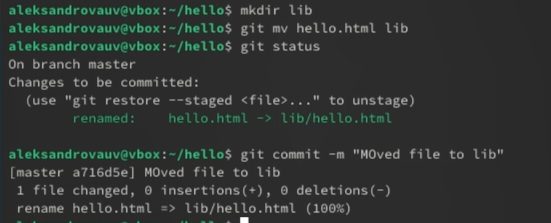


Рис. 28: Каталог lib

## 3.10 1.13 Подробнее о структуре

### 3.10.1 1.13.1 Добавление index.html

Добавим файл index.html в наш репозиторий.

<html>  
 <body>  
 <iframe src="lib/hello.html" width="200" height="200"></iframe>  
 </body>  
</html>

Добавляю файл и делаю коммит (рис. 29).

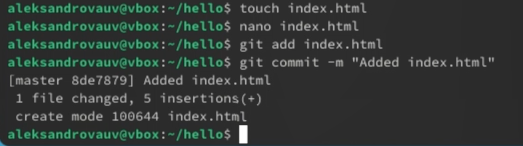


Рис. 29: index.html

Теперь при открытии index.html, вы должны увидеть кусок страницы hello в маленьком окошке.

### 3.10.2 1.14 Git внутри: Каталог .git

#### 3.10.2.1 1.14.1 Каталог .git

Выполняю команду (рис. 23):

ls -C .git

Это каталог, в котором хранится вся информация git.

ls -C .git/objects

Мы видим набор каталогов, имена которых состоят из 2 символов. Имена каталогов являются первыми двумя буквами хэша sha1 объекта, хранящегося в git.

ls -C .git/objects/<dir>

Смотрю в один из каталогов с именем из 2 букв. Мы видим файлы с именами из 38 символов. Это файлы, содержащие объекты, хранящиеся в git. Они сжаты и закодированы, поэтому просмотр их содержимого нам мало чем поможет.

cat .git/config

Это файл конфигурации, создающийся для каждого конкретного проекта. Записи в этом файле будут перезаписывать записи в файле .gitconfig вашего главного каталога, по крайней мере в рамках этого проекта.

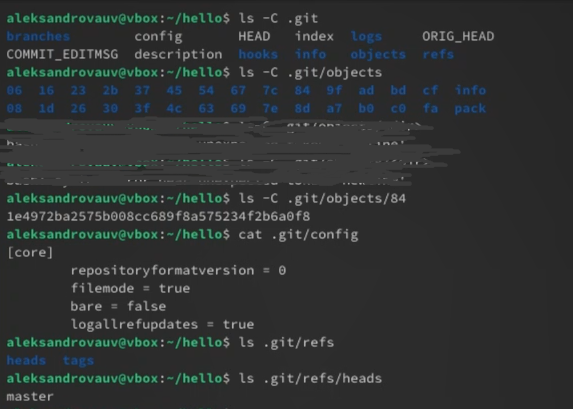


Рис. 30: Внутренний каталог

ls .git/refs  
ls .git/refs/heads  
ls .git/refs/tags  
cat .git/refs/tags/v1

Я узнаю файлы в подкаталоге тегов. Каждый файл соответствует тегу, ранее созданному с помощью команды git tag. Его содержание — это всего лишь хэш коммита, привязанный к тегу. Каталог heads практически аналогичен, но используется для веток, а не тегов. На данный момент у нас есть только одна ветка, так что все, что мы видим в этом каталоге – это ветка master (рис. 31).

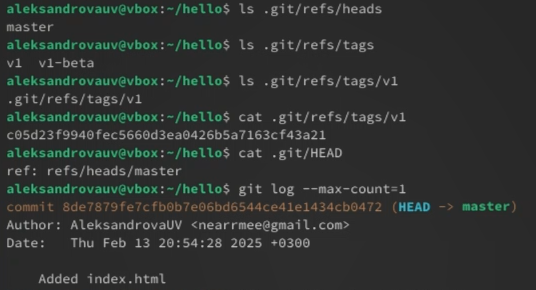


Рис. 31: сае

Файл HEAD содержит ссылку на текущую ветку, в данный момент это ветка master.

## 3.11 1.15 Работа непосредственно с объектами git

### 3.11.1 1.15.2 Вывод последнего коммита с помощью SHA1 хэша

Ищу последний коммит (рис. 32).

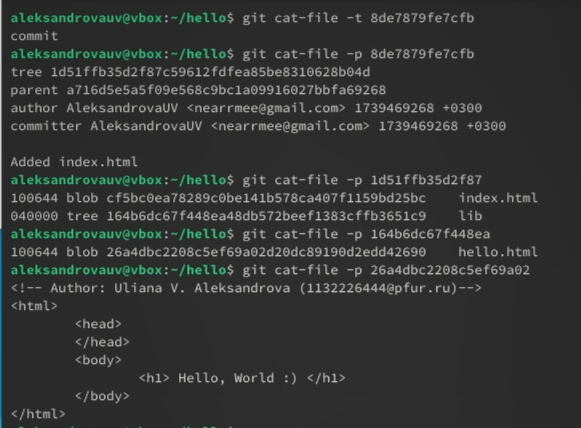


Рис. 32: работа с хэш-ключами

Мне удалось найти оригинальный файл hello.html с самого первого коммита вручную по ссылкам SHA1 хэша в последнем коммите.

## 3.12 1.16 Создание ветки

Пора сделать наш hello world более выразительным. Так как это может занять некоторое время, лучше переместить эти изменения в отдельную ветку, чтобы изолировать их от изменений в ветке master (рис. 33).

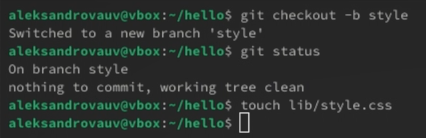


Рис. 33: переключение

### 3.12.1 1.16.2 Добавьте файл стилей style.css

Создаю файл lib/style.css:

h1 {  
 color: red;  
}

И выполняю:

git add lib/style.css  
git commit -m "Added css stylesheet"

### 3.12.2 1.16.3 Изменение основной страницы

Обновляю файл hello.html, чтобы использовать стили style.css (рис. 34) (рис. 35).



Рис. 34: Изменение файла

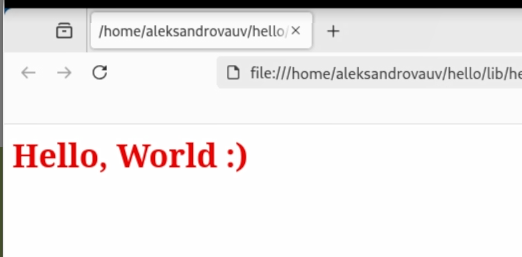


Рис. 35: Измененная страничка

Произвожу необходимые коммиты (рис. 36).

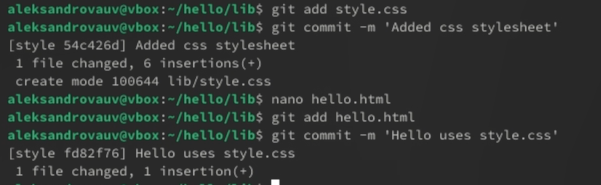


Рис. 36: Коммиты

### 3.12.3 1.16.4 Изменение index.html

Обновляю файл index.html, чтобы он тоже использовал style.css.

<html>  
 <head>  
 <link type="text/css" rel="stylesheet" media="all" href="lib/style.css" />  
 </head>  
 <body>  
 <iframe src="lib/hello.html" width="200" height="200"></iframe>  
 </body>  
</html>

И выполняю:

git add index.html  
git commit -m "Updated index.html"

## 3.13 1.17 Навигация по веткам

Просмотрю каталог (рис. 37).

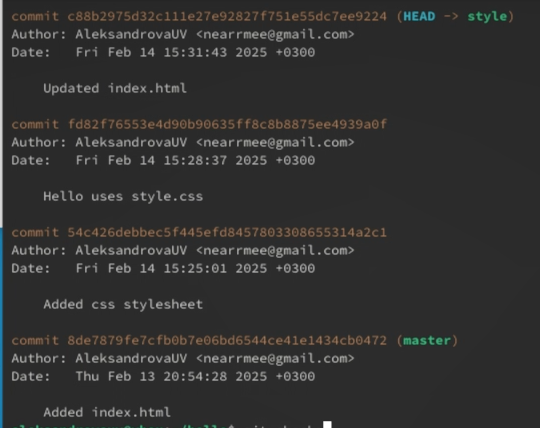


Рис. 37: Каталог

### 3.13.1 1.17.1 Переключение между ветками

Сейчас мы находимся на ветке master. Это заметно по тому, что файл hello.html не использует стили style.css.

Переключимся обратно. содержимое lib/hello.html подтверждает, что мы вернулись на ветку style (рис. 38).

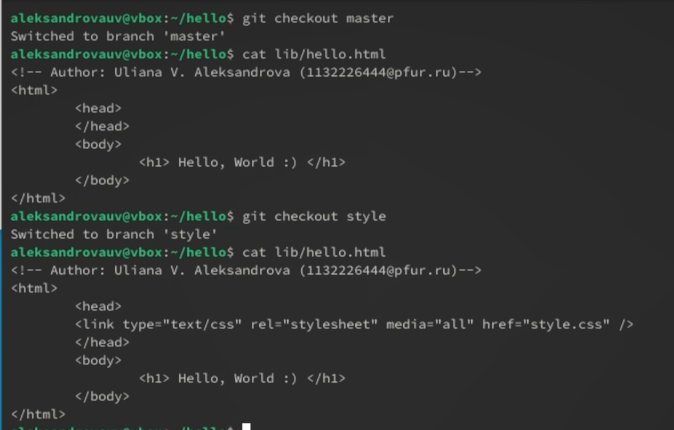


Рис. 38: Переключение между ветками

##.18 Изменения в ветке master

### 3.13.2 1.18.1 Файл README в ветке master

Создаю файл README.md (рис. 39).



Рис. 39: Создание README

### 3.13.3 1.19.1 Просмотр отличающихся веток

### 3.13.4 1.19.2 Текущие ветки

Теперь у нас в репозитории есть две отличающиеся ветки (рис. 40).

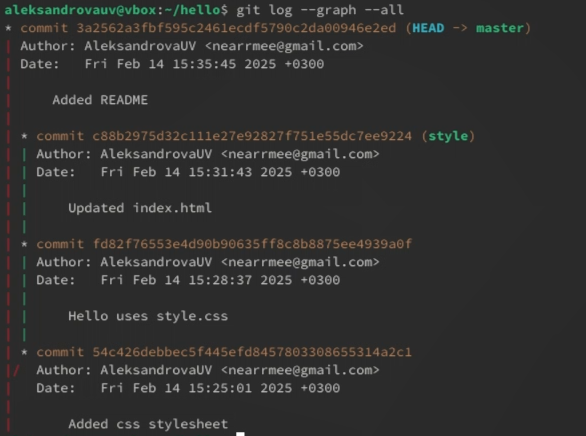


Рис. 40: Новая ветка

Добавление опции --graph в git log вызывает построение дерева коммитов с помощью простых ASCII символов. Мы видим обе ветки (style и master), и то, что ветка master является текущей HEAD. Общим предшественником обеих веток является коммит «Added index.html». Опция --all гарантированно означает, что мы видим все ветки. По умолчанию показывается только текущая ветка.

## 3.14 1.20 Слияние

### 3.14.1 1.20.1 Слияние веток

Слияние переносит изменения из двух веток в одну. Сольем master с style (рис. 41) (рис. 42).

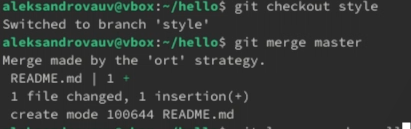


Рис. 41: Слияние веток

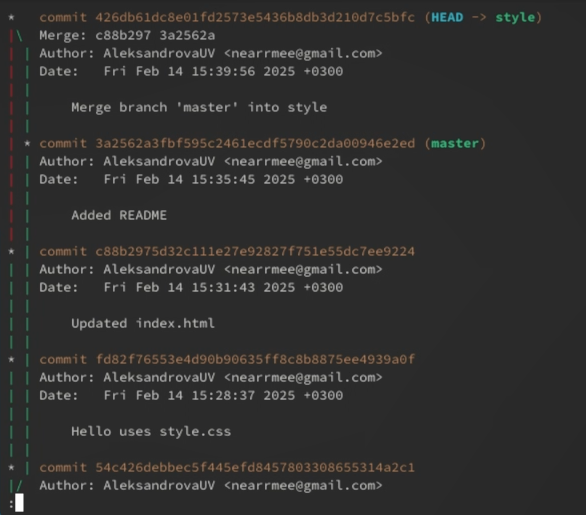


Рис. 42: Слияние веток лог

## 3.15 1.21 Создание конфликта

### 3.15.1 1.21.1 Вернитесь в master и создайте конфликт

Вернитесь в ветку master и внесу изменения в документ (рис. 43).

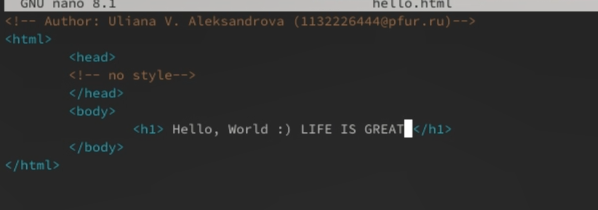


Рис. 43: Изменение файла

Отправлю изменения:

git add lib/hello.html  
git commit -m 'Life is great'

### 3.15.2 1.21.2 Просмотр веток

После коммита «Added README» ветка master была объединена с веткой style, но в настоящее время в master есть дополнительный коммит, который не был слит с style. Последнее изменение в master конфликтует с некоторыми изменениями в style (рис. 44).

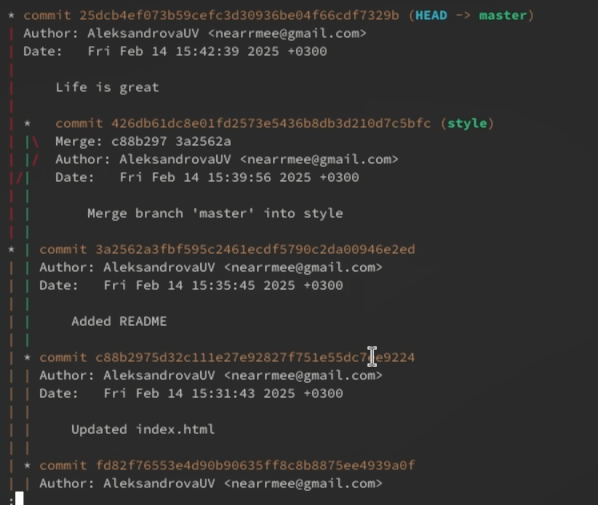


Рис. 44: Конфликт веток

## 3.16 1.22 Разрешение конфликтов

### 3.16.1 1.22.1 Слияние master с веткой style

Теперь вернусь к ветке style и попытаюсь объединить ее с новой веткой master.

git checkout style  
git merge master

В файлеlib/hello.html мы видим (рис. 45).

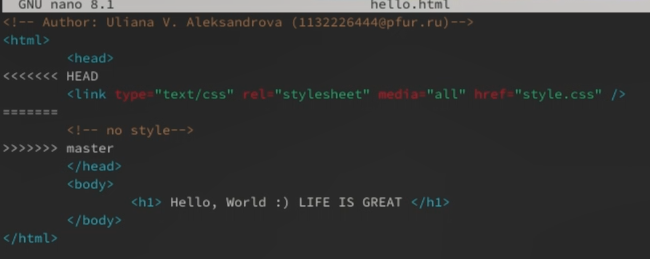


Рис. 45: Файл

Первый раздел — версия текущей ветки (style). Второй раздел — версия ветки master.

### 3.16.2 1.22.2 Решение конфликта

Для решения конфликта, вношу изменения (рис. 46).

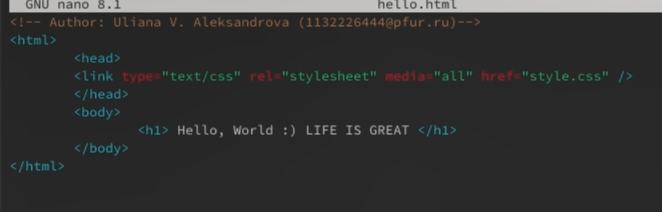


Рис. 46: Изменение файла

И выполняю коммит:

git add lib/hello.html  
git commit -m "Merged master fixed conflict."

## 3.17 1.23 Сброс ветки style

### 3.17.1 1.23.1 Сброс ветки style

Вернемся на ветке style к точке перед тем, как мы слили ее с веткой master. Мы можем сбросить ветку к любому коммиту. По сути, это изменение указателя ветки на любую точку дерева коммитов. В этом случае мы хотим вернуться в ветке style в точку перед слиянием с master. Нам необходимо найти последний коммит перед слиянием.

Мы видим, что коммит «Updated index.html» был последним на ветке style перед слиянием. Сбросим ветку style к этому коммиту (рис. 47).

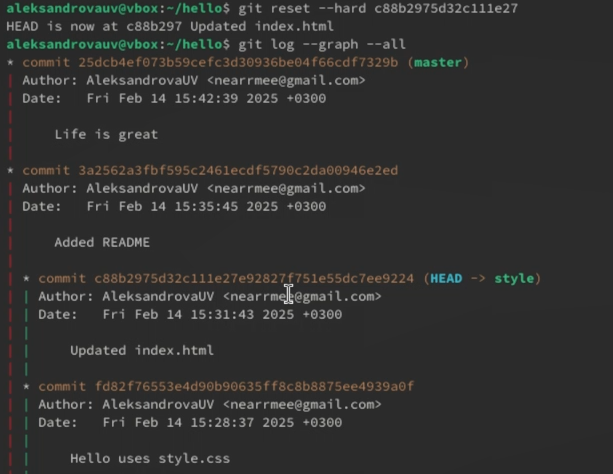


Рис. 47: Сброс ветки

## 3.18 1.24 Сброс ветки master

### 3.18.1 1.24.1 Сброс ветки master

Добавив интерактивный режим в ветку master, мы внесли изменения, конфликтующие с изменениями в ветке style. Вернусь в ветке master в точку перед внесением конфликтующих изменений (рис. 48).



Рис. 48: Ветка master

Коммит «Added README» идет непосредственно перед коммитом конфликтующего интерактивного режима. Мы сбросим ветку master к коммиту «Added README» (рис. 49).

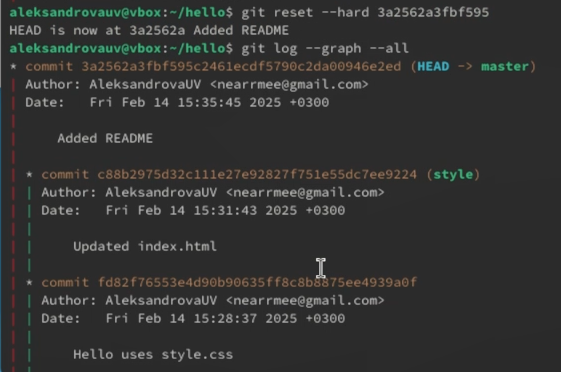


Рис. 49: Сброс ветки master

Лог выглядит, как будто репозиторий был перемотан назад во времени к точке до какого-либо слияния.

## 3.19 1.25 Перебазирование

Используем команду rebase вместо команды merge (рис. 50).

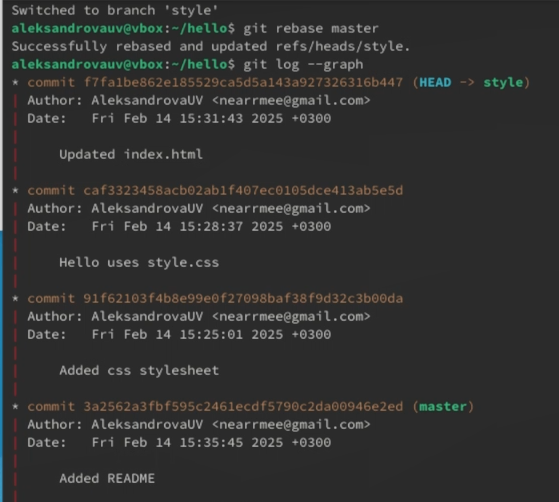


Рис. 50: Перебазирование

## 3.20 1.26 Слияние в ветку master

Мы поддерживали соответствие ветки style с веткой master (с помощью rebase), теперь давайте солью изменения style в ветку master (рис. 51).

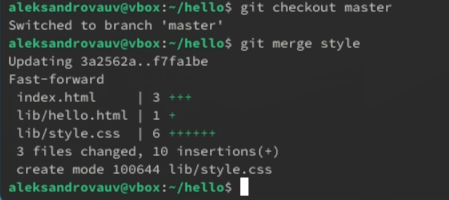
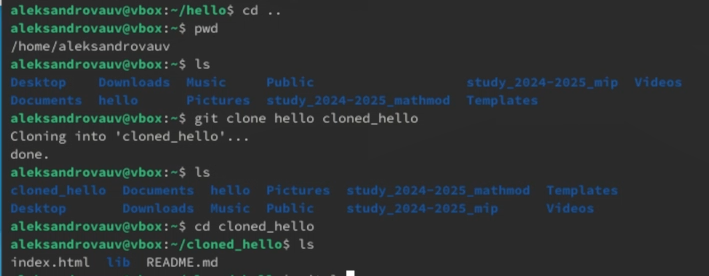


Рис. 51: Слияние в ветку master

## 3.21 1.27 Клонирование репозиториев

### 3.21.1 1.27.1 Клонирование репозиториев

Делаю клон своего рабочего каталога hello в домашнем репозитории (рис. **¿fig:064?**).



## 3.22 1.28 Работа с клонированный репозиторием

### 3.22.1 1.28.1 Лог

Просмотрим лог (рис. 52).

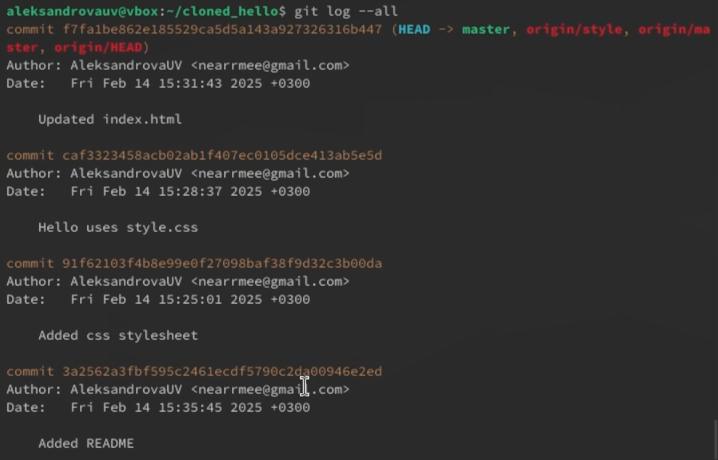


Рис. 52: Лог после клонирования

Мы видим список всех коммитов в новый репозиторий.

## 3.23 1.29 Что такое origin?

Рассмотрим, что такое origin (рис. 53).

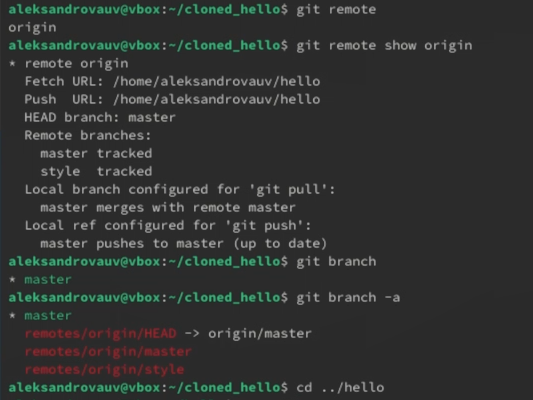


Рис. 53: Слияние в ветку master

Удаленные репозитории обычно размещаются на отдельной машине, возможно, централизованном сервере. Однако, как мы видим здесь, они могут с тем же успехом указывать на репозиторий на той же машине.

## 3.24 1.31 Изменение оригинального репозитория

### 3.24.1 1.31.1 Внесение изменения в оригинальный репозиторий hello

Сейчас мы находимся в репозитории hello. Вношу следующие изменения в файл README.md:

This is the Hello World example from the git tutorial.

Теперь в оригинальном репозитории есть более поздние изменения, которых нет в клонированной версии. Далее мы извлечем и сольем эти изменения в клонированный репозиторий.

### 3.24.2 1.31.2 Извлечение изменений

Извлекаю изменения (рис. 54).

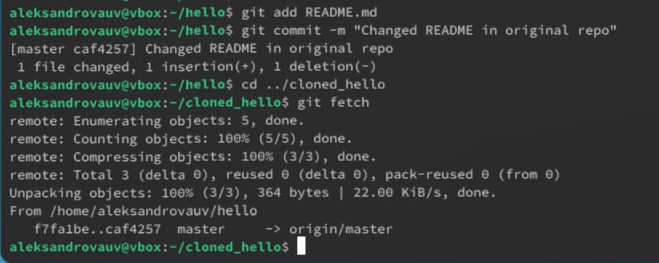


Рис. 54: Извлечение изменений

Мы можем продемонстрировать, что клонированный файл README.md не изменился. Поэтому я сливаю изменения в локальную ветку master. Теперь изменения есть (рис. 55).

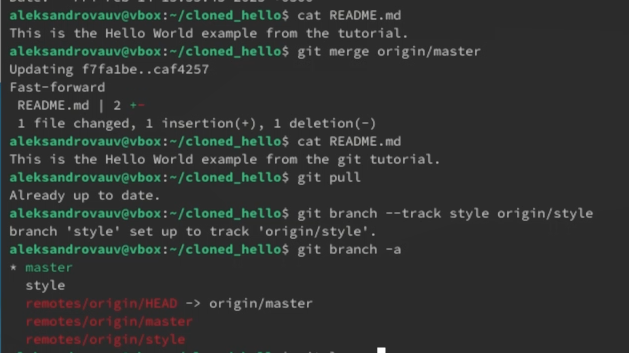


Рис. 55: Изменения

## 3.25 1.33 Добавление ветки наблюдения

### 3.25.1 1.33.1 Добавление локальной ветки, которая отслеживает удаленную

Добавляю и открываю лог (рис. 56).

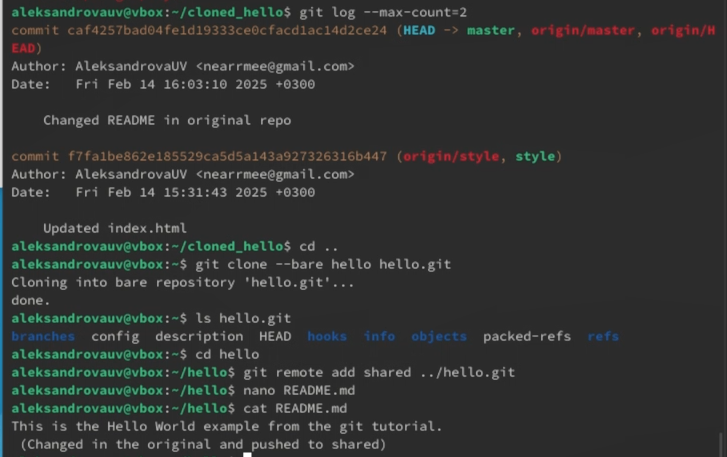


Рис. 56: Изменения

## 3.26 1.35 Создайте чистый репозиторий

Выполняю:

cd ..  
git clone --bare hello hello.git  
ls hello.git

## 3.27 1.36 Добавление удаленного репозитория

Теперь добавляю репозиторий hello.git к оригинальному репозиторию.

cd hello  
git remote add shared ../hello.git

## 3.28 1.37 Отправка изменений

Редактирую файл README.md и делаю коммит.

Файл README.md:

This is the Hello World example from the git tutorial. (Changed in the original and pushed to shared)

git checkout master  
git add README  
git commit -m "Added shared comment to readme"

Теперь отправляю изменения в общий репозиторий:

git push shared master

## 3.29 1.38 Извлечение общих изменений

Извлекаю общие изменения (рис. 57).

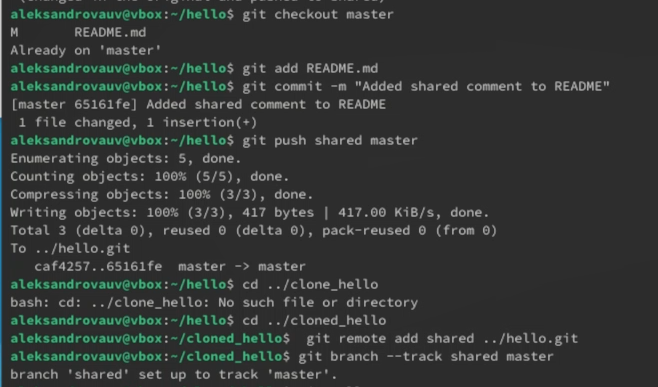


Рис. 57: Изменения

# 4 Выводы

Я освоила работу в Git.