Отчёт по лабораторной работе №1

Система контроля версий Git

Дарья Эдуардовна Ибатулина

Содержание

# 1 Цель работы

Познакомиться с новыми командами Git, повторить уже изученный материал.

# 2 Задание

* Настроить Git;
* Создать тестовый репозиторий и поработать с ним.

# 3 Теоретическое введение

Системы контроля версий (Version Control System, VCS) применяются при работе нескольких человек над одним проектом. Обычно основное дерево проекта хранится в локальном или удалённом репозитории, к которому настроен доступ для участников проекта. При внесении изменений в содержание проекта система контроля версий позволяет их фиксировать, совмещать изменения, произведённые разными участниками проекта, производить откат к любой более ранней версии проекта, если это требуется.

В классических системах контроля версий используется централизованная модель, предполагающая наличие единого репозитория для хранения файлов. Выполнение большинства функций по управлению версиями осуществляется специальным сервером. Участник проекта (пользователь) перед началом работы посредством определённых команд получает нужную ему версию файлов. После внесения изменений, пользователь размещает новую версию в хранилище. При этом предыдущие версии не удаляются из центрального хранилища и к ним можно вернуться в любой момент. Сервер может сохранять не полную версию изменённых файлов, а производить так называемую дельта-компрессию — сохранять только изменения между последовательными версиями, что позволяет уменьшить объём хранимых данных.

Системы контроля версий поддерживают возможность отслеживания и разрешения конфликтов, которые могут возникнуть при работе нескольких человек над одним файлом. Можно объединить (слить) изменения, сделанные разными участниками (автоматически или вручную), вручную выбрать нужную версию, отменить изменения вовсе или заблокировать файлы для изменения. В зависимости от настроек блокировка не позволяет другим пользователям получить рабочую копию или препятствует изменению рабочей копии файла средствами файловой системы ОС, обеспечивая таким образом, привилегированный доступ только одному пользователю, работающему с файлом.

Системы контроля версий также могут обеспечивать дополнительные, более гибкие функциональные возможности. Например, они могут поддерживать работу с несколькими версиями одного файла, сохраняя общую историю изменений до точки ветвления версий и собственные истории изменений каждой ветви. Кроме того, обычно доступна информация о том, кто из участников, когда и какие изменения вносил. Обычно такого рода информация хранится в журнале изменений, доступ к которому можно ограничить.

В отличие от классических, в распределённых системах контроля версий центральный репозиторий не является обязательным.

Среди классических VCS наиболее известны CVS, Subversion, а среди распределённых — Git, Bazaar, Mercurial. Принципы их работы схожи, отличаются они в основном синтаксисом используемых в работе команд.

# 4 Выполнение лабораторной работы

У меня уже установлен гит, так как я работала с ним на предыдущих курсах, поэтому пропускаю шаг с настройкой и перехожу сразу к заданию параметров.

Настраиваю параметры установки окончаний строк. Если core.safecrlf установлен в true или warm, git проверяет, если преобразование является обратимым для текущей настройки *core.autocrlf*, *core.safecrlf true* - отвержение необратимого преобразования lf<->crlf (рис. [[1](#fig:001)]).

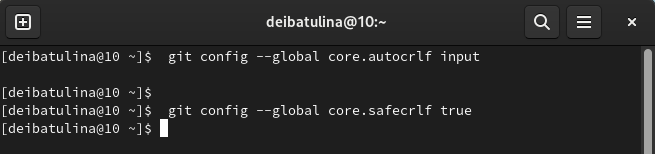


Figure 1: Настройка параметров установки окончаний строк

Чтобы избежать нечитаемых строк, установила соответствующий флаг (рис. [[2](#fig:002)]).

Figure 2: Установка флага

Figure 2: Установка флага

Начну работу в домашнем каталоге с создания пустого каталога с именем hello, затем захожу в него и создала там файл с именем hello.html (рис. [[3](#fig:003)]).



Figure 3: Создание директории и файла в ней

Чтобы создать git репозиторий из этого каталога, выполнила команду git init (рис. [[4](#fig:004)]).

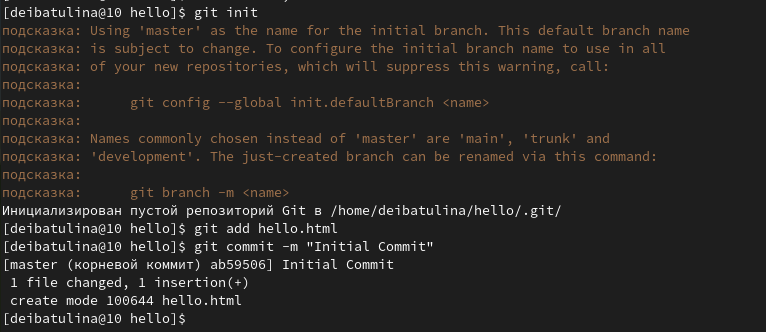


Figure 4: Создание Git-репозитория

Проверила текущее состояние репозитория (рис. [[5](#fig:005)]).

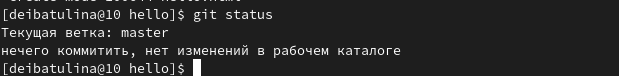


Figure 5: Проверка состояния Git-репозитория

Создала файл и заполнила его (рис. [[6](#fig:006)]).

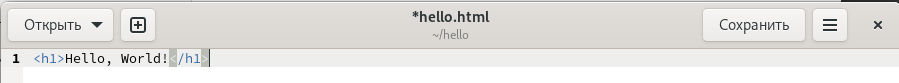


Figure 6: Создание фвйла и его заполнение

Проверила состояние рабочего каталога. git знает, что файл hello.html был изменен, но при этом эти изменения еще не зафиксированы в репозитории (рис. [[7](#fig:007)]).

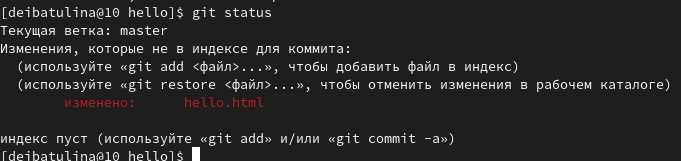


Figure 7: Проверка состояния репозитория

Добавила изменения в репозиторий (рис. [[8](#fig:008)]).

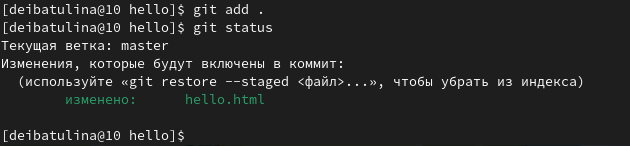


Figure 8: Добавление изменения в репозиторий

Сделала коммит и проверю позже состояние. Откроется редактор. В первой строке ввела комментарий: «Added h1 tag» (рис. [[9](#fig:009)]).

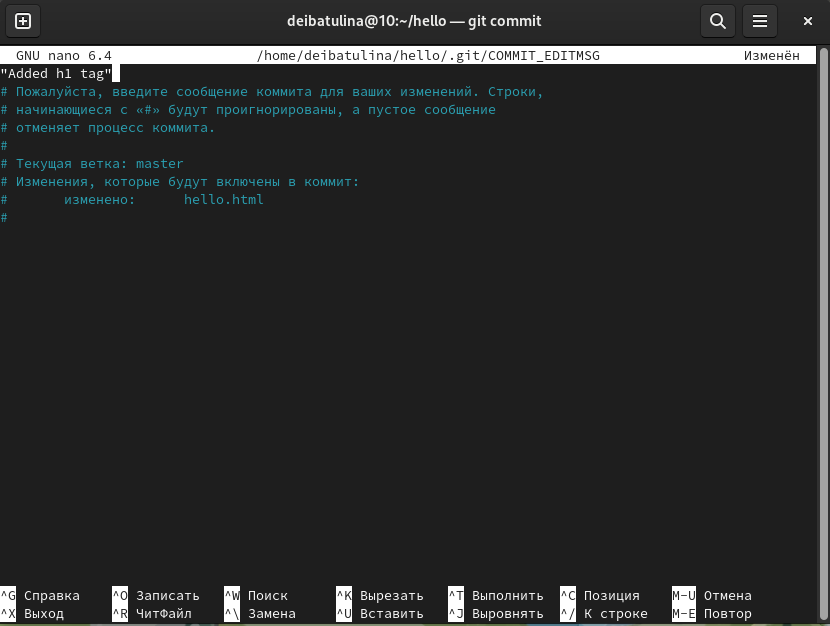


Figure 9: Коммит и редактирование коммита

Проверила состояние. Теперь все изменения проиндексированы (рис. [[10](#fig:010)]).

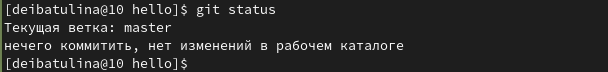


Figure 10: Проверка состояния

Редактирую файл hello.html и добавляю изменение в гит (рис. [[11](#fig:011)], [[12](#fig:012)]).

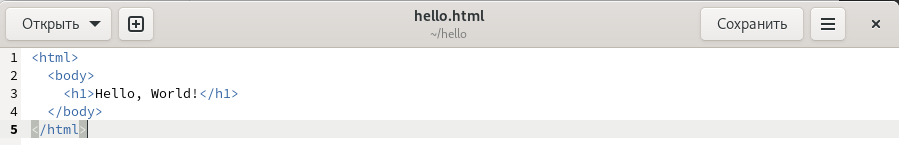


Figure 11: Редактирование файла

Figure 12: Добавление изменений

Figure 12: Добавление изменений

Вновь редактирую файл hello.html и проверяю состояние репозитория (рис. [[13](#fig:013)], [[14](#fig:014)]).

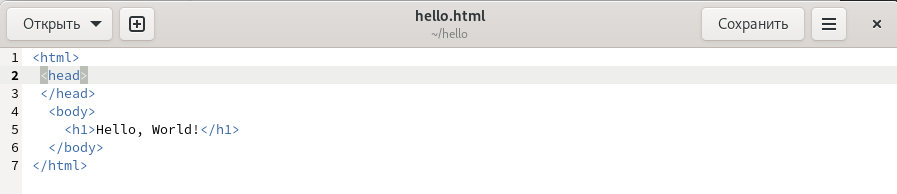


Figure 13: Редактирование файла

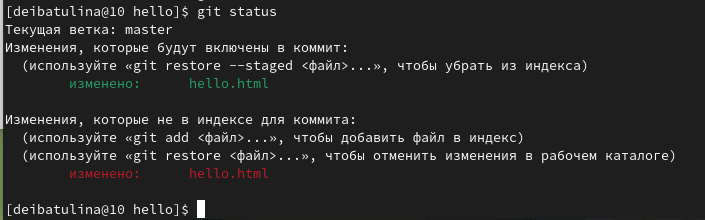


Figure 14: Проверка состояния

hello.html указан дважды в состоянии. Первое изменение (добавление стандартных тегов) проиндексировано и готово к коммиту.Второе изменение (добавление заголовков HTML) является непроиндексированным.

Теперь добавляю второе изменение в индекс, а затем проверяю состояние (рис. [[15](#fig:015)], [[16](#fig:016)], [[17](#fig:017)]). Сделала коммит второго изменения (добавление заголовков) (рис. [[18](#fig:018)]).

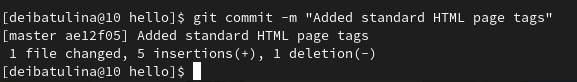


Figure 15: Коммит

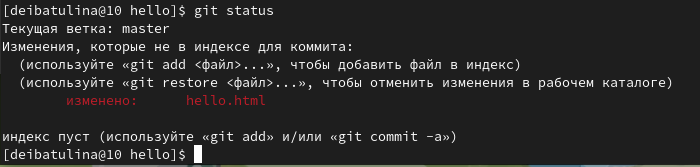


Figure 16: Проверка состояния

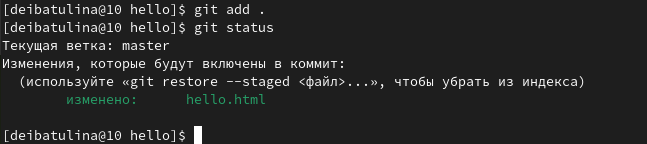


Figure 17: Добавление файлов, проверка состояния

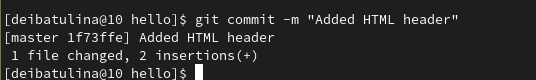


Figure 18: Коммит второго изменения

Получим список произведенных изменений (рис. [[19](#fig:019)]).

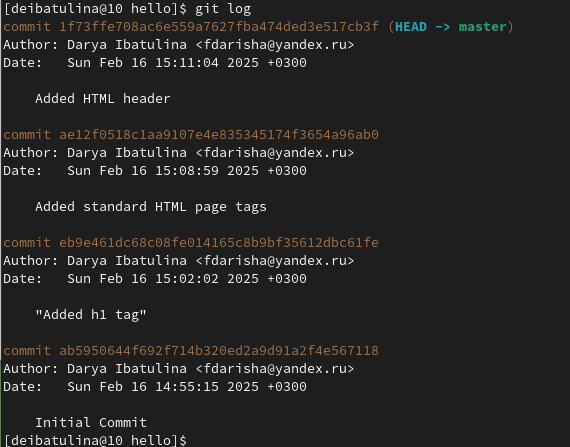


Figure 19: Список изменений - лог

Однострочный формат истории (рис. [[20](#fig:020)]).

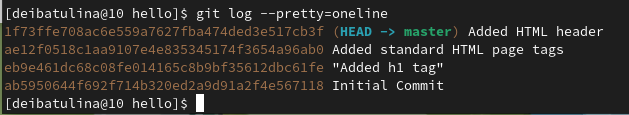


Figure 20: Формат истории - строка

Справочник по логу (рис. [[21](#fig:021)]).



Figure 21: Справочник по логу

Получаю хэши предыдущих версий, изучаю данные лога и нахожу хэш для первого коммита. Он должен быть в последней строке данных. Использую этот хэш-код (достаточно первых 7 знаков) в команде ниже.

git checkout <hash>

Затем проверяю содержимое файла hello.html.

Возвращаюсь к последней версии в ветке master и вывожу содержимое файла (рис. [[22](#fig:022)], [[23](#fig:023)]).

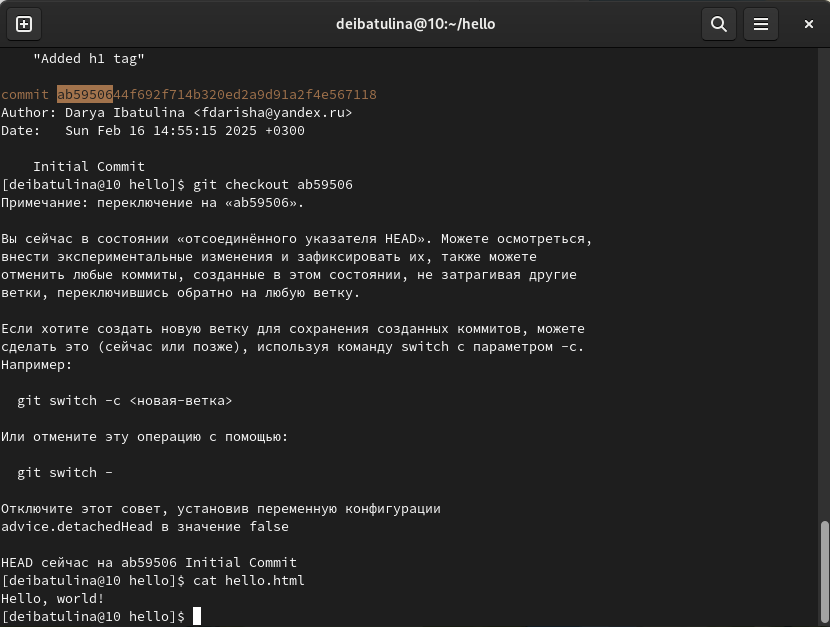


Figure 22: Переключение на ветку мастер и проверка содержимого файла

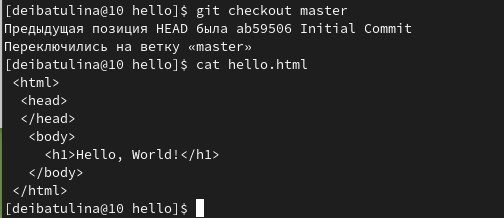


Figure 23: Возврат к предыдущей версии и вывод содержимого файла

Создаю теги и переключаюсь между ними. Вывожу логи и могу просмотреть теги (рис. [[24](#fig:024)], [[25](#fig:025)], [[26](#fig:026)]).

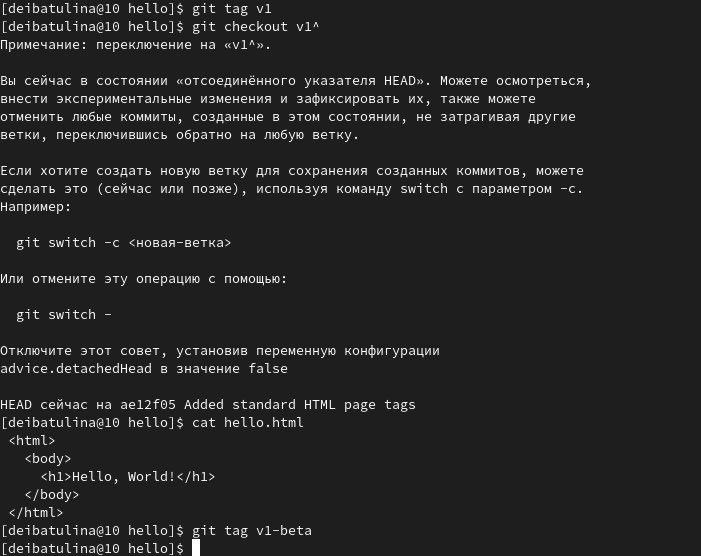


Figure 24: Переключение между тегами (1)

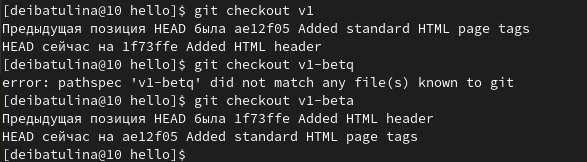


Figure 25: Переключение между тегами (2)

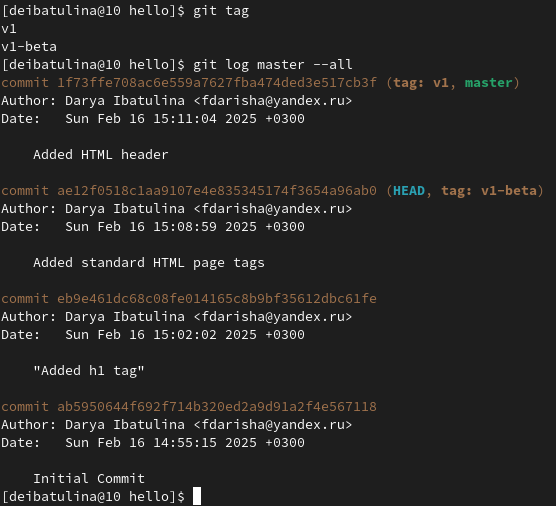


Figure 26: Переключение между тегами (3)

Пробую отменить ошибочно добавленные коммиты и проверяю логи, используя команду git revert. Всё получается (рис. [[27](#fig:027)], [[28](#fig:028)], [[29](#fig:029)], [[30](#fig:030)], [[31](#fig:031)], [[32](#fig:032)]? [[33](#fig:033)], [[34](#fig:034)], [[35](#fig:035)], [[36](#fig:036)], [[37](#fig:037)]).

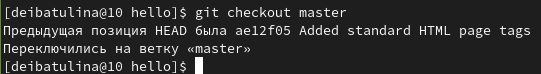


Figure 27: Шаг 1

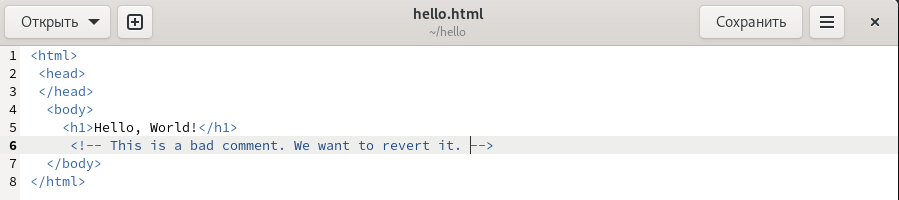


Figure 28: Шаг 2

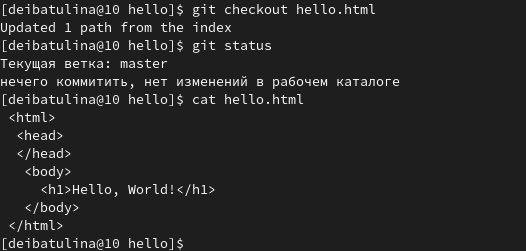


Figure 29: Шаг 3

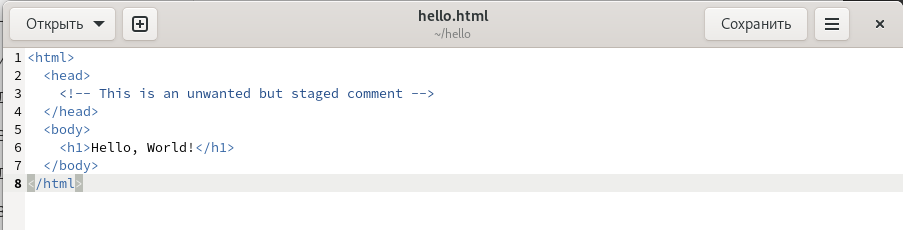


Figure 30: Шаг 4

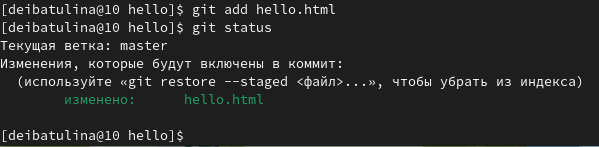


Figure 31: Шаг 5

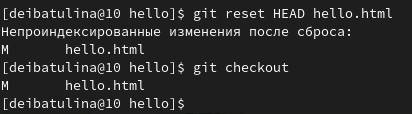


Figure 32: Шаг 6

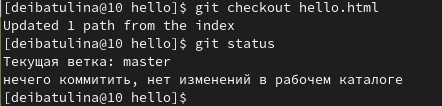


Figure 33: Шаг 7

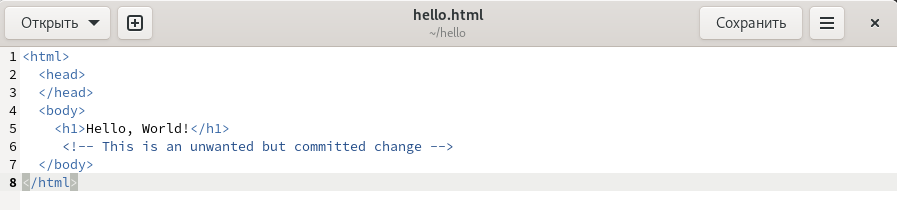


Figure 34: Шаг 8

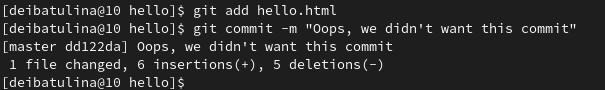


Figure 35: Шаг 9

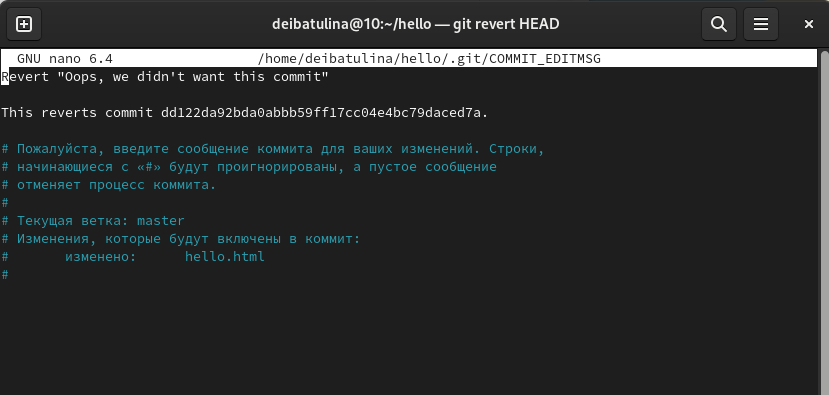


Figure 36: Шаг 10

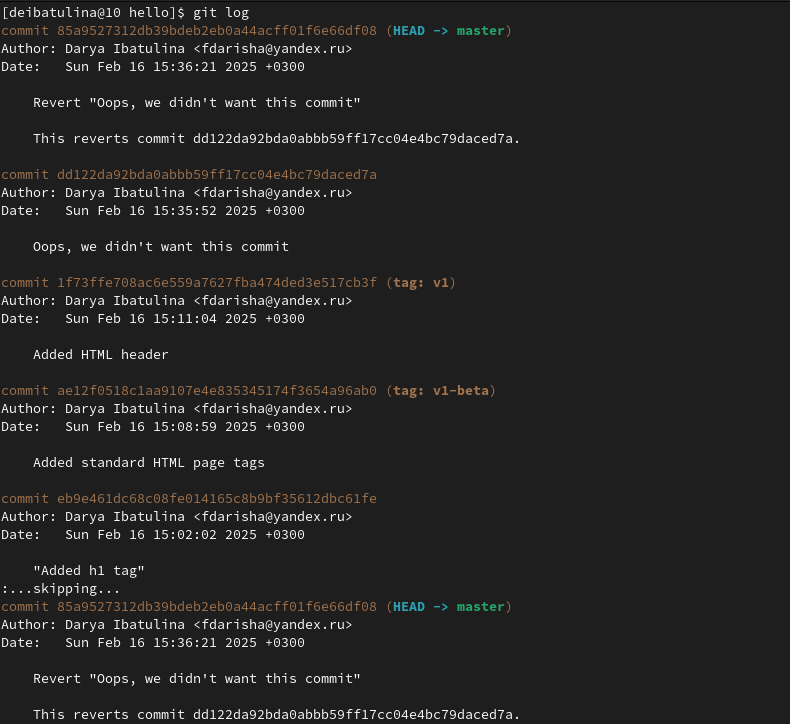


Figure 37: Шаг 11

Чтобы коммиты удалялись и из истории ветки, существует более мощная команда git reset (рис. [[38](#fig:038)], [[39](#fig:039)], [[40](#fig:040)]).

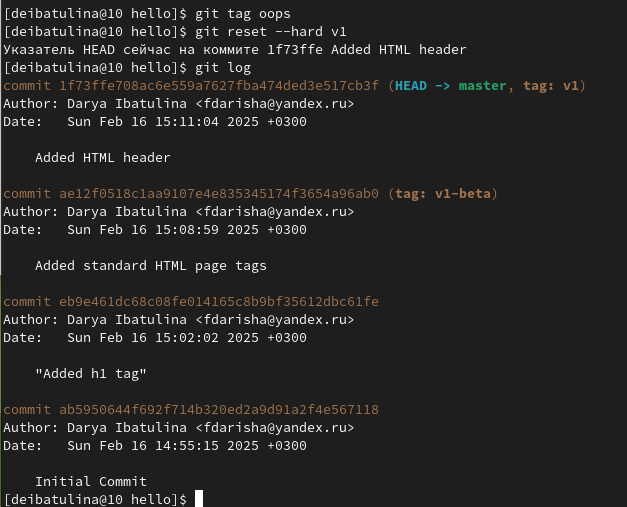


Figure 38: Шаг 1

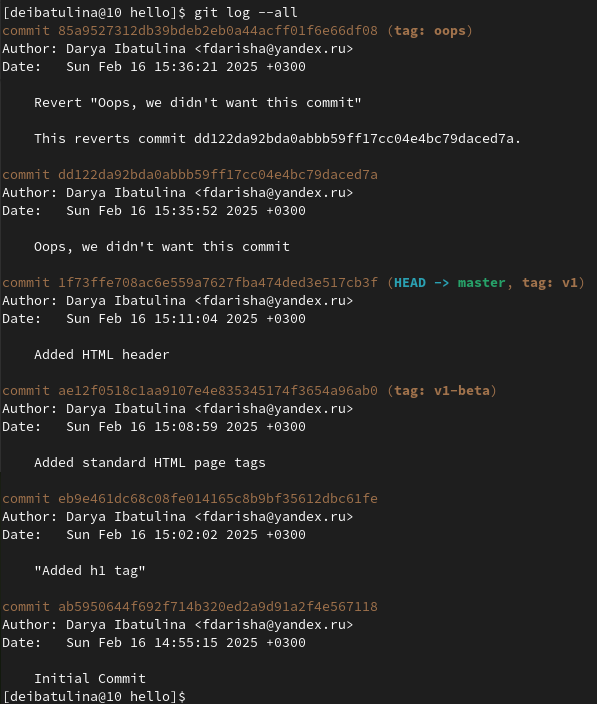


Figure 39: Шаг 2

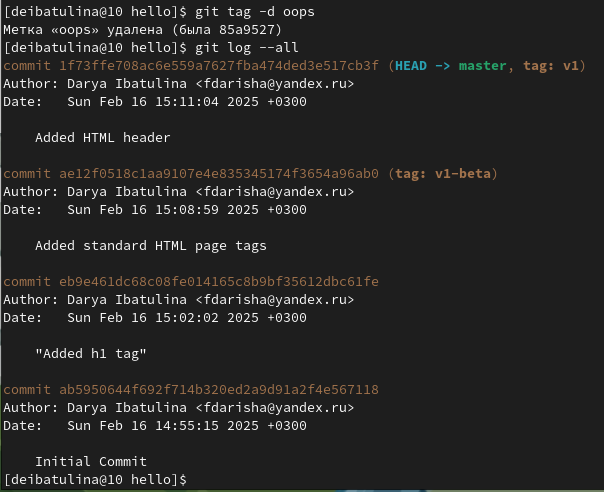


Figure 40: Шаг 3

Добавление автора и электронной почты в файл приветствия (рис. [[41](#fig:041)], [[42](#fig:042)], [[43](#fig:043)], [[44](#fig:044)], [[45](#fig:045)]).

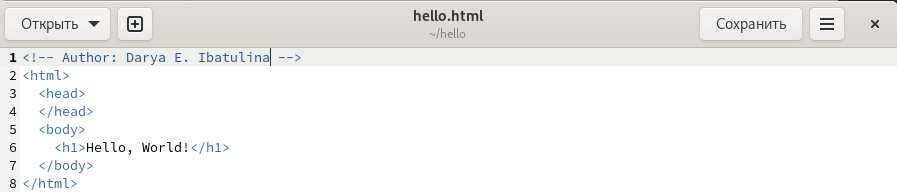


Figure 41: Шаг 1

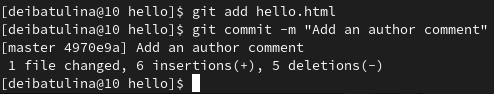


Figure 42: Шаг 2

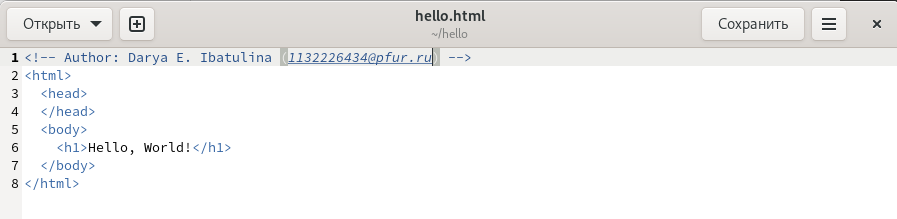


Figure 43: Шаг 3

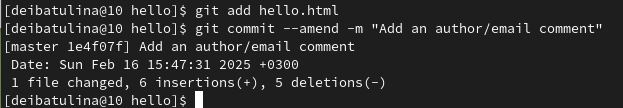


Figure 44: Шаг 4

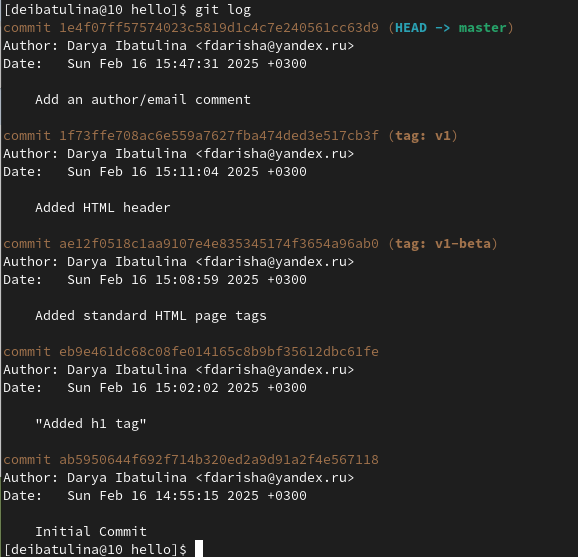


Figure 45: Шаг 5

Создаем новый каталог и копируем в него содержимое hello, отправляем в репозиторий (рис. [[46](#fig:046)], [[47](#fig:047)]).

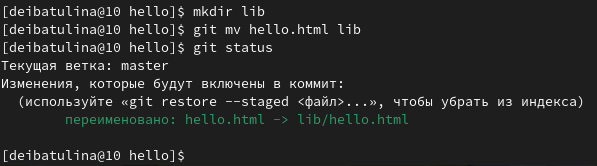


Figure 46: Копирование

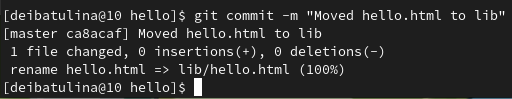


Figure 47: Отправка изменений

Создаем новый файл, отправляем в репозиторий (рис. [[48](#fig:048)], [[49](#fig:049)]).



Figure 48: Создание и редактирование файла



Figure 49: Отправка изменений

Теперь при открытии index.html, я вижу кусок страницы hello в маленьком окошке (рис. [-[50](#fig:050)).

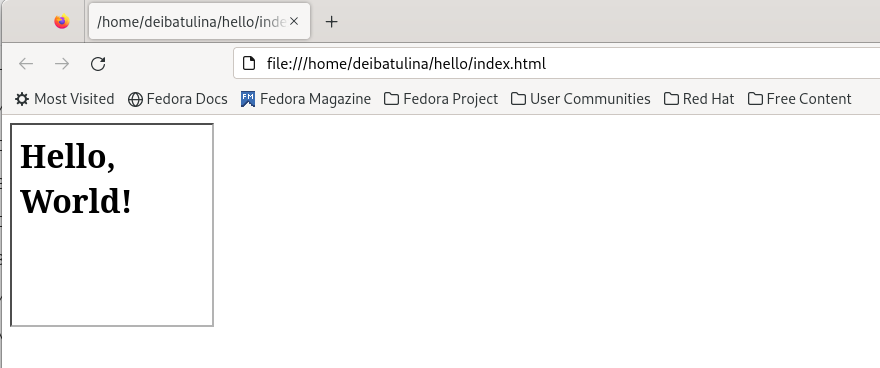


Figure 50: Создание и редактирование файла

Углубляюсь в базу данных объектов, смотрю файлы конфига, ветки и теги, файлы HEAD (рис. [[51](#fig:051)], [[52](#fig:052)], [[53](#fig:053)], [[54](#fig:054)], [[55](#fig:055)]).

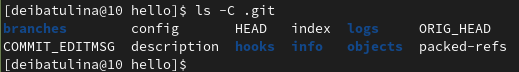


Figure 51: Шаг 1

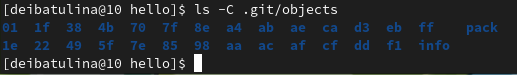


Figure 52: Шаг 2

Figure 53: Шаг 3

Figure 53: Шаг 3

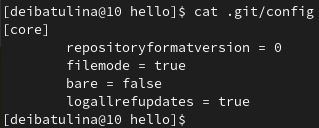


Figure 54: Шаг 4

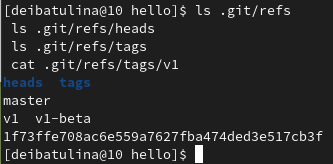


Figure 55: Шаг 5

Работаю непосредственно с объектами git (рис. [[56](#fig:056)], [[57](#fig:057)], [[58](#fig:058)], [[59](#fig:059)], [[60](#fig:060)], [[61](#fig:061)]).

Figure 56: Шаг 1

Figure 56: Шаг 1

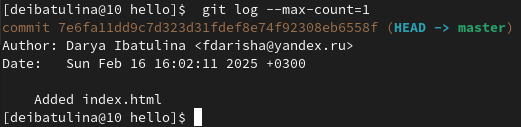


Figure 57: Шаг 2

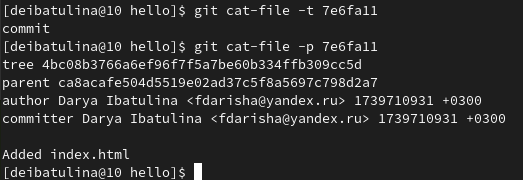


Figure 58: Шаг 3

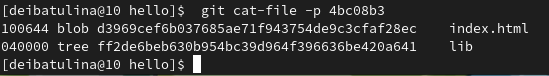


Figure 59: Шаг 4

Figure 60: Шаг 5

Figure 60: Шаг 5

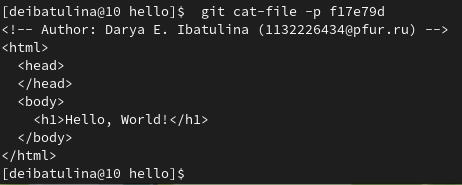


Figure 61: Шаг 6

Добавляю файл стилей и новую ветку style (рис. [[62](#fig:062)], [[63](#fig:063)], [[64](#fig:064)], [[65](#fig:065)], [[66](#fig:066)]).

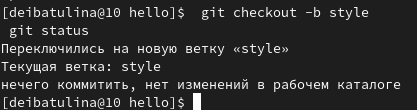


Figure 62: Шаг 1



Figure 63: Шаг 2

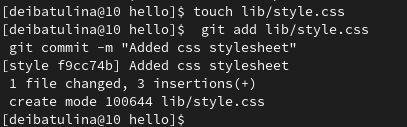


Figure 64: Шаг 3

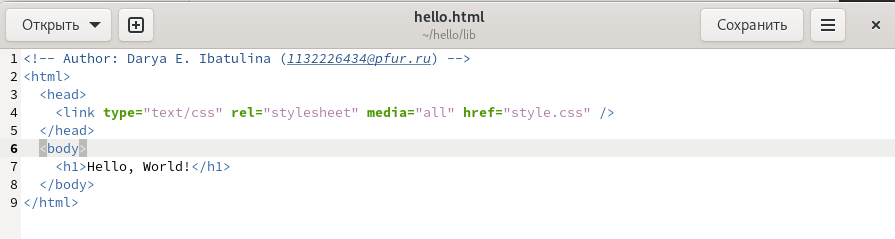


Figure 65: Шаг 4

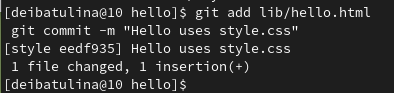


Figure 66: Шаг 5

Изменяю файлы так, чтобы использовались стили style.css (рис. [[67](#fig:067)], [[68](#fig:068)], [[69](#fig:069)], [[70](#fig:070)], [[71](#fig:071)], [[72](#fig:072)], [[73](#fig:073)]).



Figure 67: Шаг 1

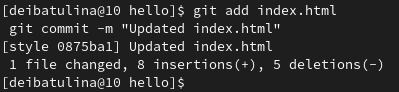


Figure 68: Шаг 2

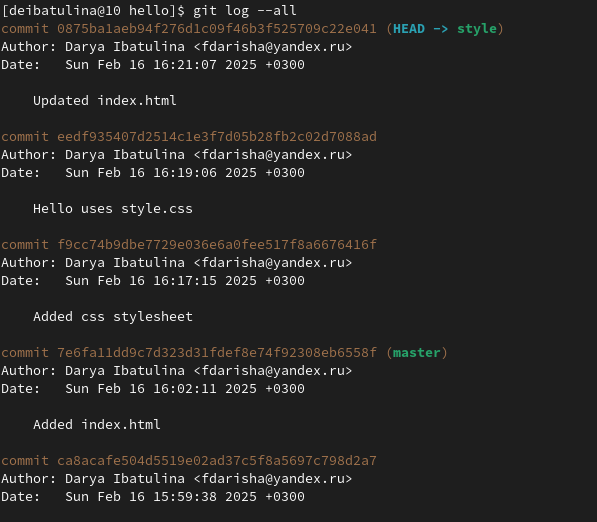


Figure 69: Шаг 3

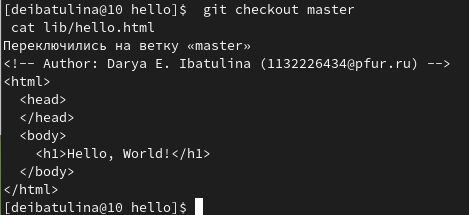


Figure 70: Шаг 4

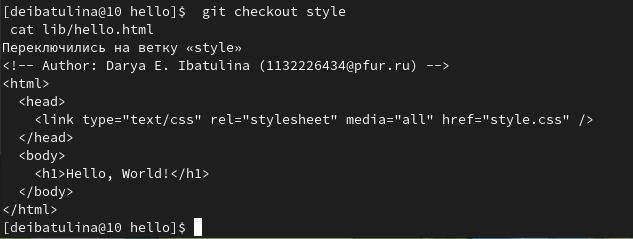


Figure 71: Шаг 5

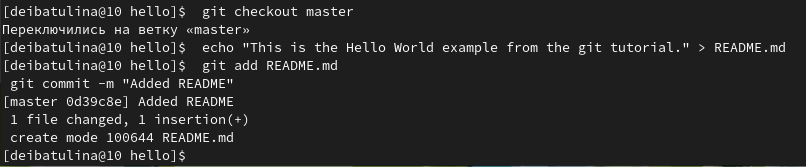


Figure 72: Шаг 6

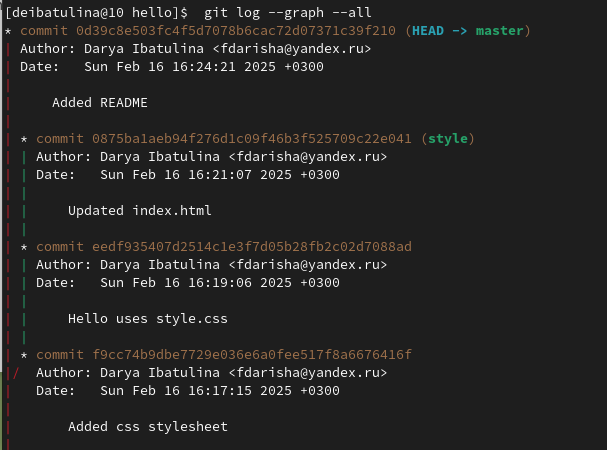


Figure 73: Шаг 7

Выполняю слияние веток. Появляются конфликты, которые необходимо разрешить. Успешно справляюсь с данной задачей (рис. [[74](#fig:074)], [[75](#fig:075)], [[76](#fig:076)], [[77](#fig:077)], [[78](#fig:078)], [[79](#fig:079)], [[80](#fig:080)], [[81](#fig:081)], [[82](#fig:082)], [[83](#fig:083)], [[84](#fig:084)], [[85](#fig:085)], [[86](#fig:086)], [[87](#fig:087)], [[88](#fig:088)], [[89](#fig:089)]).

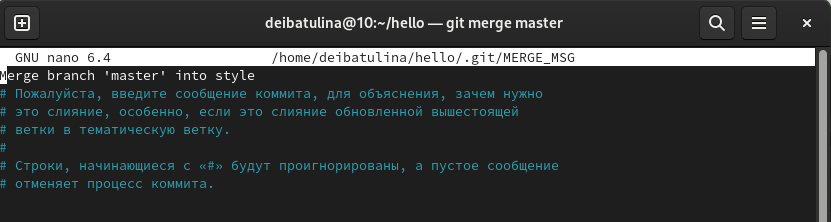


Figure 74: Шаг 1

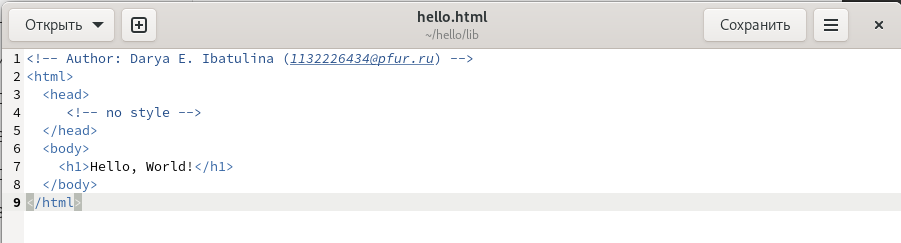


Figure 75: Шаг 2

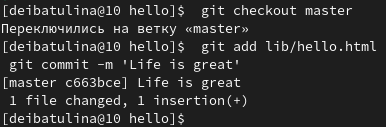


Figure 76: Шаг 3

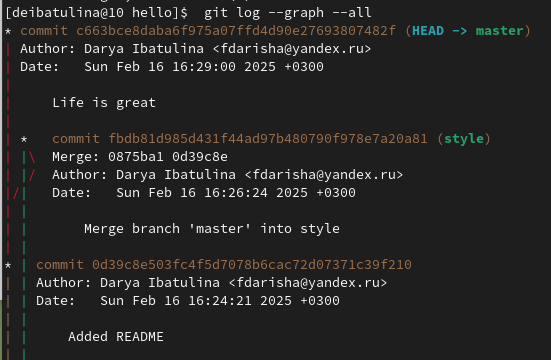


Figure 77: Шаг 4

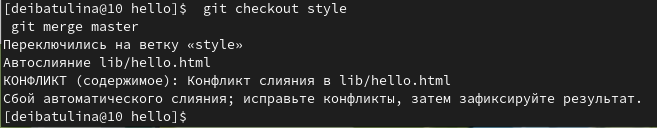


Figure 78: Шаг 5



Figure 79: Шаг 6



Figure 80: Шаг 7

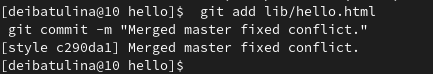


Figure 81: Шаг 8

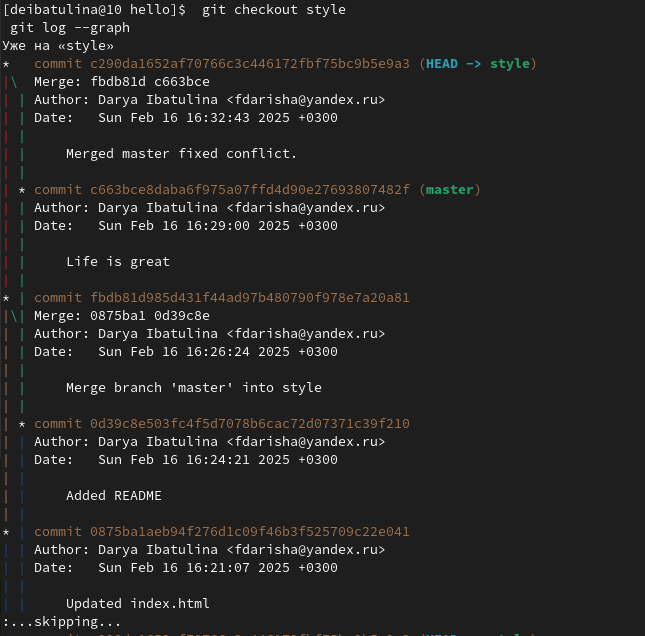


Figure 82: Шаг 9

Figure 83: Шаг 10

Figure 83: Шаг 10

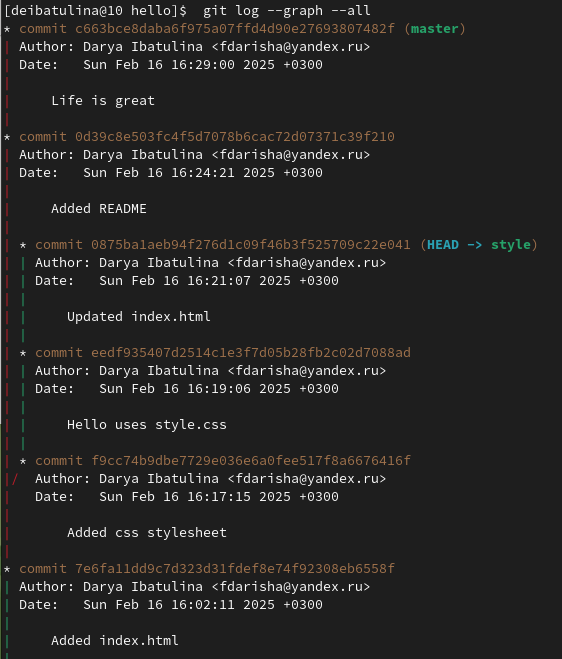


Figure 84: Шаг 11

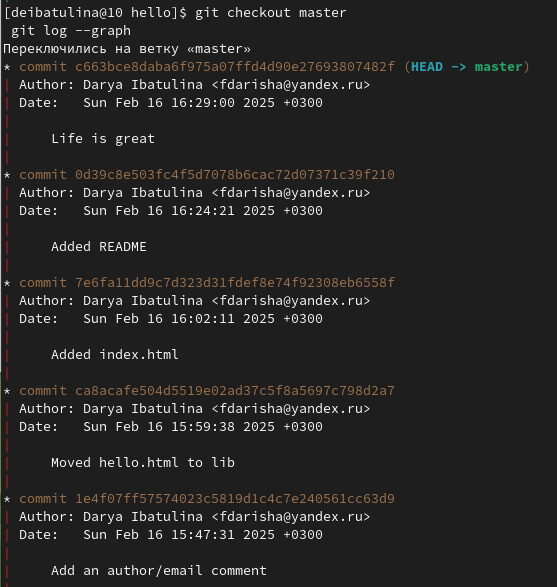


Figure 85: Шаг 12

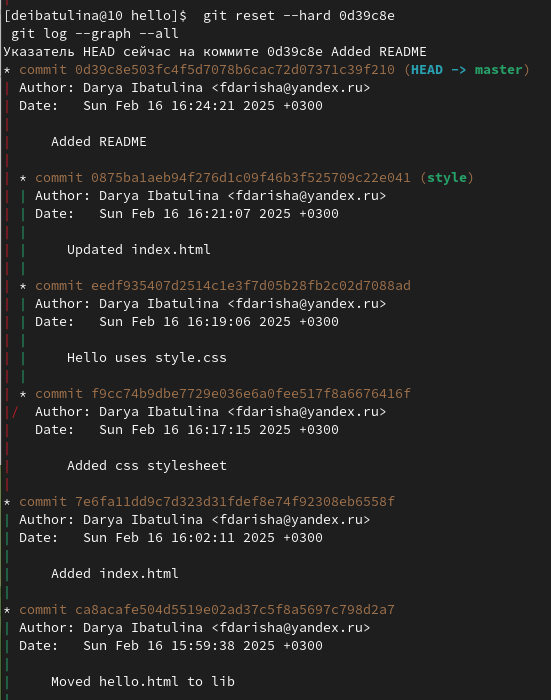


Figure 86: Шаг 13

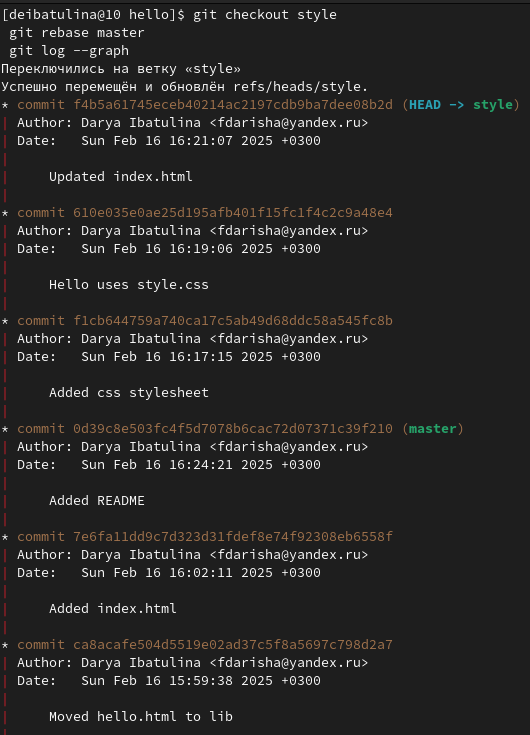


Figure 87: Шаг 14

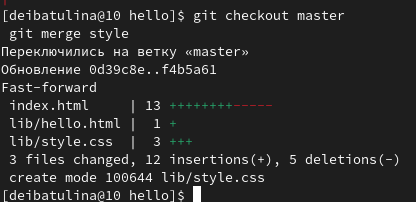


Figure 88: Шаг 15

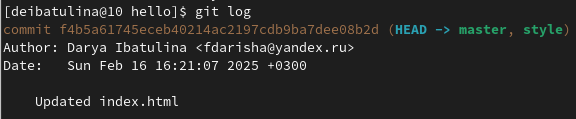


Figure 89: Шаг 16

Клонируем один репозиторий в другой созданный (рис. [[90](#fig:090)], [[91](#fig:091)], [[92](#fig:092)], [[93](#fig:093)]).

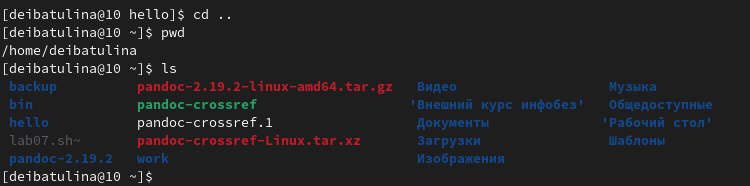


Figure 90: Шаг 1

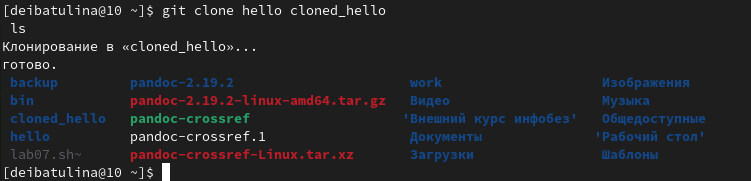


Figure 91: Шаг 2

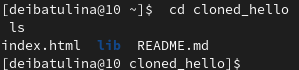


Figure 92: Шаг 3

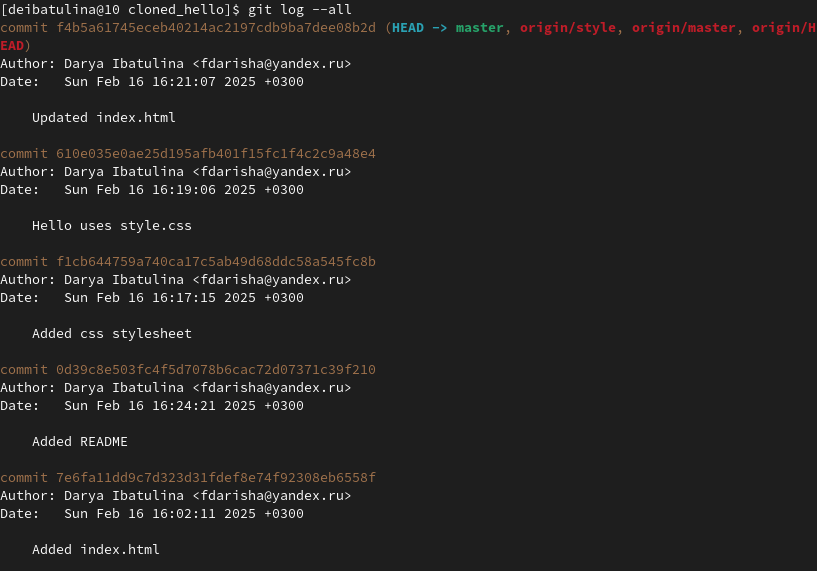


Figure 93: Шаг 4

Мы видим, что клонированный репозиторий знает об имени по умолчанию удаленного репозитория. Давайте посмотрим, можем ли мы получить более подробную информацию об имени по умолчанию. Удаленные репозитории обычно размещаются наотдельной машине, возможно, централизованном сервере. Однако, как мы видим здесь, они могут с тем же успехом указывать на репозиторий на той же машине. Нет ничего особенного в имени «origin», однако существует традиция использовать «origin» в качестве имени первичного централизованного репозитория (если таковой имеется) рис. [[94](#fig:094)]).

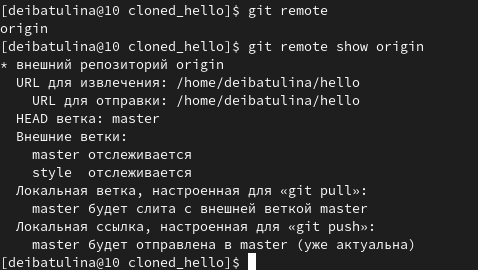


Figure 94: git remote, origin

Просмотрим все ветки (в том числе и удаленные) рис. [[95](#fig:095)]).

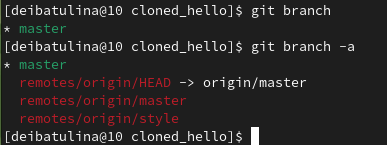


Figure 95: Просмотр всех веток (в том числе и удаленных)

Клонируем один репозиторий в другой созданный (рис. [[96](#fig:096)], [[97](#fig:097)], [[98](#fig:098)], [[99](#fig:099)], [[100](#fig:100)], [[101](#fig:101)]).

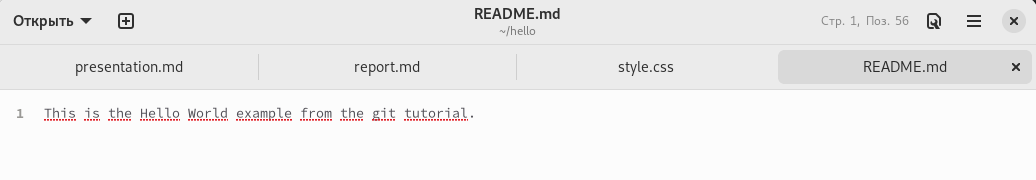


Figure 96: Шаг 1

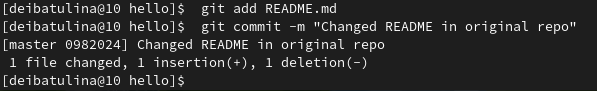


Figure 97: Шаг 2

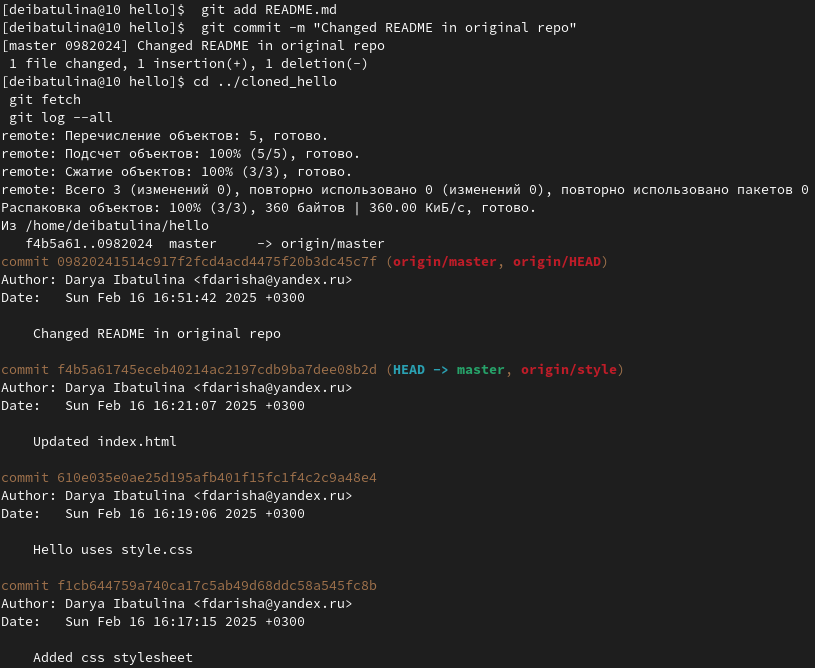


Figure 98: Шаг 3

Figure 99: Шаг 4

Figure 99: Шаг 4

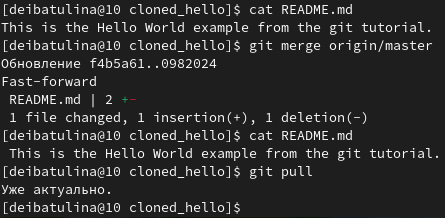


Figure 100: Шаг 5

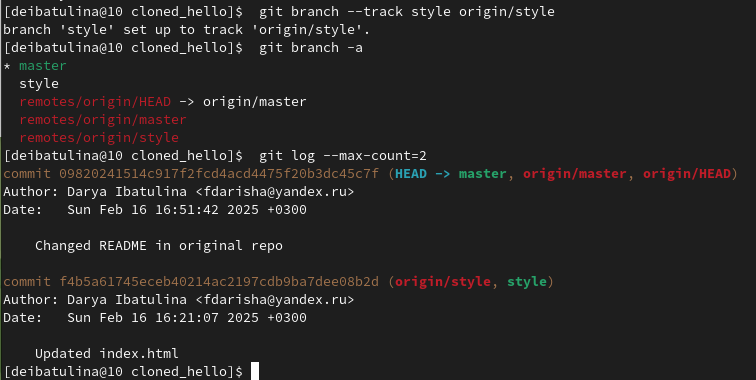


Figure 101: Шаг 6

Создадим чистый репозиторий. Сейчас мы находимся в рабочем (домашнем) каталоге. Как правило, репозитории, оканчивающиеся на .git, являются чистыми репозиториями. Мы видим,что в репозитории hello.git нетрабочего каталога.По сути, это есть не что иное,как каталог.git нечистого репозитория (рис. [[102](#fig:102)]).



Figure 102: Чистый репозиторий

Так как чистые репозитории, как правило, расшариваются на каком-нибудь сетевом сервере,нам необходимо отправить наши изменения в другие репозитории. Начнем с создания изменения для отправки. Отредактирую файл README.md и сделаю коммит (рис. [[103](#fig:103)], [[104](#fig:104)], [[105](#fig:105)]).

Figure 103: Переход в каталог hello

Figure 103: Переход в каталог hello

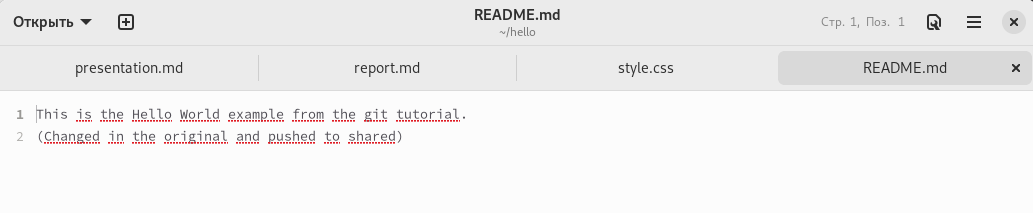


Figure 104: Файл README.md

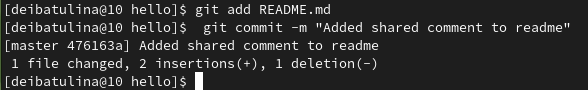


Figure 105: Коммит

Отправляю изменения в общий репозиторий (рис. [[106](#fig:106)]).

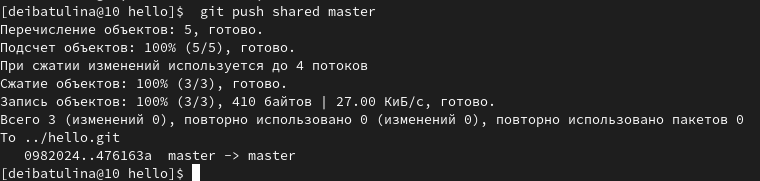


Figure 106: Отправка изменений в общий репозиторий

Переключаемся в клонированный репозиторий и извлекаем изменения из общего репозитория (рис. [[107](#fig:107)]).

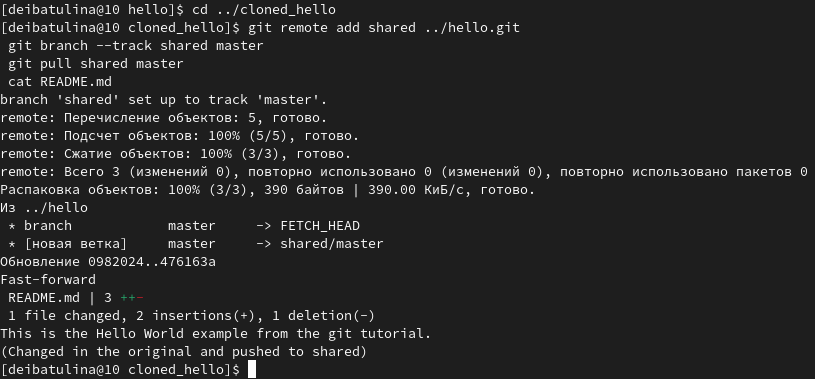


Figure 107: Извлечение изменений из общего репозитория в клонированный

# 5 Выводы

В результате проделывания лабораторной работы я разобралась с тем, как работает гит и его команды. Научилась отменять коммиты не только из репозитория, но и из истории веток. Вспомнила основные команды гита, уже изученные мною, и применила полученные знания на практике, создав тестовый репозиторий.

# Список литературы

1. Кулябов Д.С., Королькова А.В. Руководство к лабораторной работе №1. Математическое моделирование. - 2025. — 31 с.