Отчёт по лабораторной работе №1

дисциплина: Математическое моделирование

Студент: Кузнецова София Вадимовна

Содержание

# Цель работы

Научиться работать с git.

# Выполнение лабораторной работы

Выполним следующие команды, чтобы git узнал наше имя и электронную почту.Настроим core.autocrlf с параметрами true и input (делает все переводы строк текстовых файлов в главном репозитории одинаковы). Что бы избежать нечитаемых строк, установим соответствующий флаг.

Имя и электронная почта

Имя и электронная почта

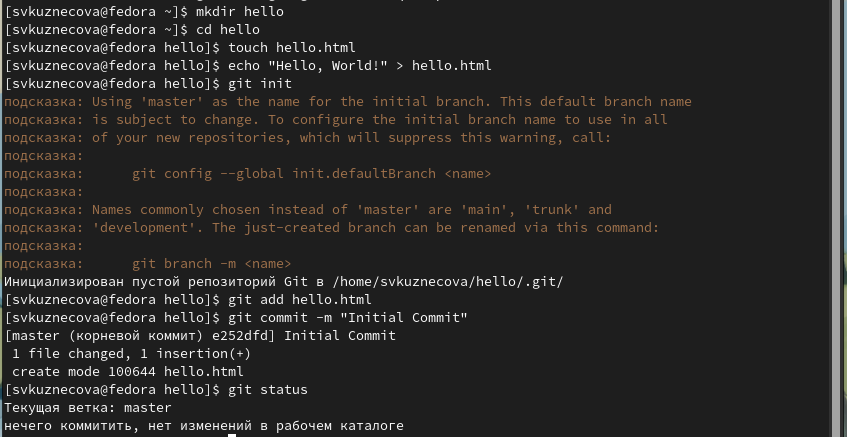
Сore.autocrlf с параметрами true и input

Сore.autocrlf с параметрами true и input

Флаг

Флаг

Начнём работу в пустом рабочем каталоге с создания пустого каталога с именем hello, затем войдём в него и создадим там файл с именем hello.html. Чтобы создать git репозиторий из этого каталога, выполним команду git init. Добавим файл в репозиторий и используем команду git status, чтобы проверить текущее состояние репозитория.

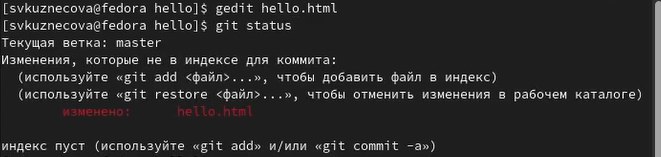


Создание файла hello.html

Добавим кое-какие HTML-теги к нашему приветствию. Изменим содержимое файла hello.html. Проверим состояние рабочего каталога.

Изменение файла hello.html

Изменение файла hello.html

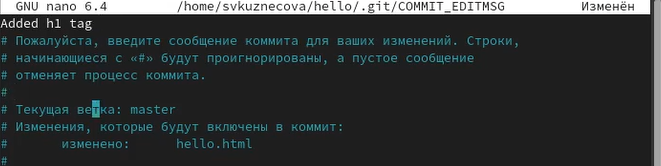


Состояние каталога

Сделаем коммит и проверим состояние. Открылся редактор. В первой строке введём комментарий: «Added h1 tag». Сохраним файл и выйдем из редактора. Теперь еще раз проверим состояние.

Git commit

Git commit



Комментарий: «Added h1 tag»

Состояние каталога

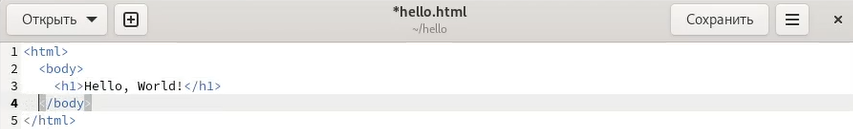
Состояние каталога

Изменим страницу «Hello, World», чтобы она содержала стандартные теги

и

. Теперь добавим это изменение в индекс git. Теперь добавим заголовки HTML (секцию

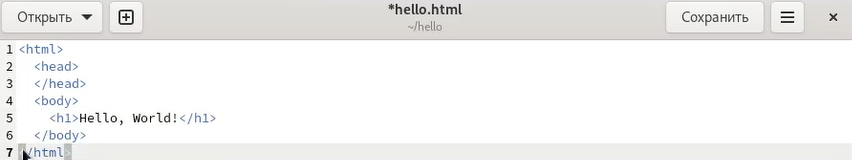
) к странице «Hello, World»



Добавим стандартные теги

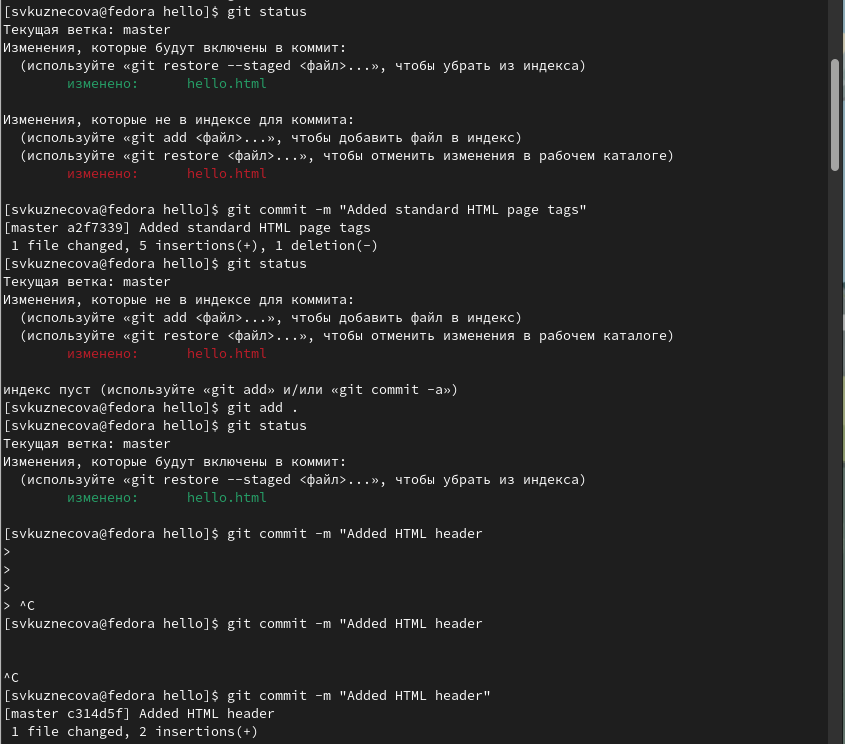
Добавление изменения

Добавление изменения



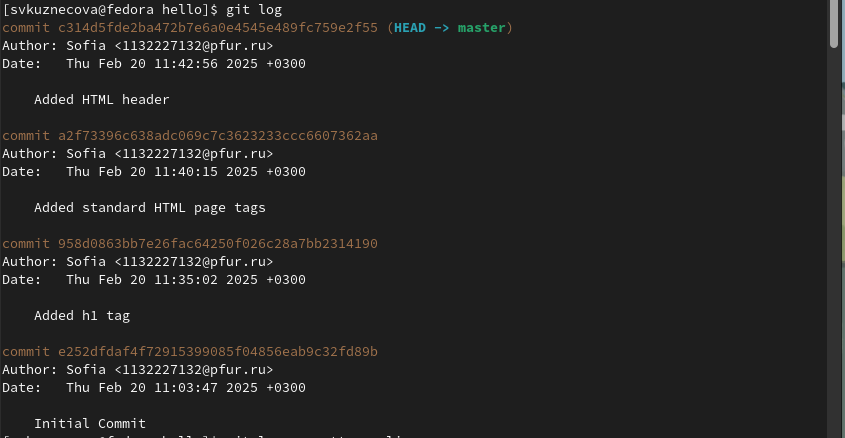
Добавим секцию

Проверим текущий статус. Произведём коммит проиндексированного изменения (значение по умолчанию), а затем еще раз проверим состояние. Теперь добавим второе изменение в индекс, а затем проверим состояние с помощью команды git status. Второе изменение было проиндексировано и готово к коммиту. Сделаем коммит второго изменения.

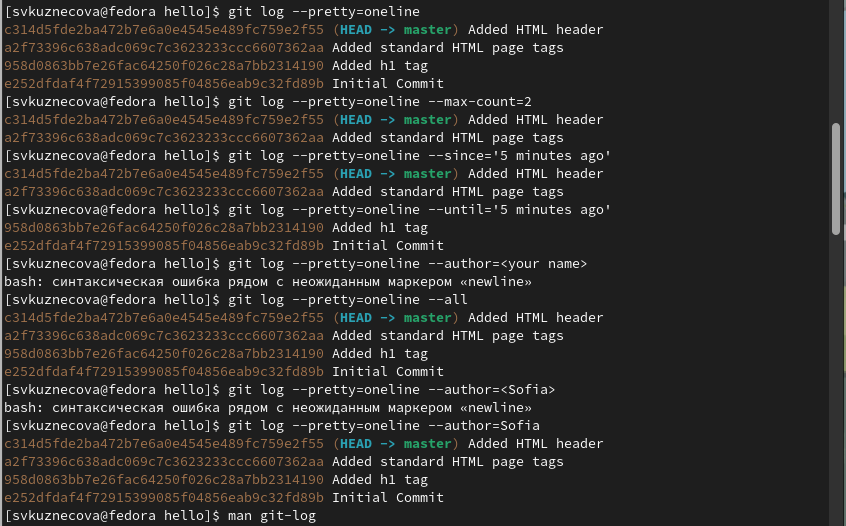


Коммит

Получим список произведенных изменений. Посмотрим однострочный формат истории. Также посмотрим много вариантов отображения лога и справочную страницу.

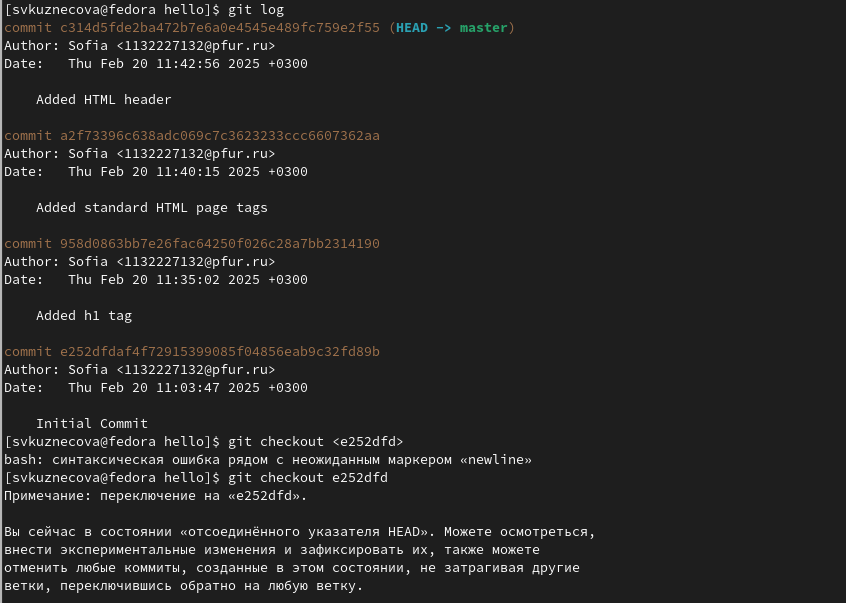


Список произведенных изменений

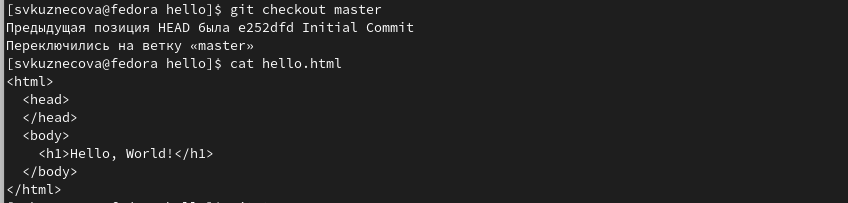


Однострочный формат истории. Много вариантов отображения лога. Справочная страница.

Получим хэши предыдущих версий. Изучим данные лога и найдём хэш для первого коммита. Используем этот хэш-код в команде ниже. Затем проверим содержимое файла hello.html. Вернёмся к последней версии в ветке master.

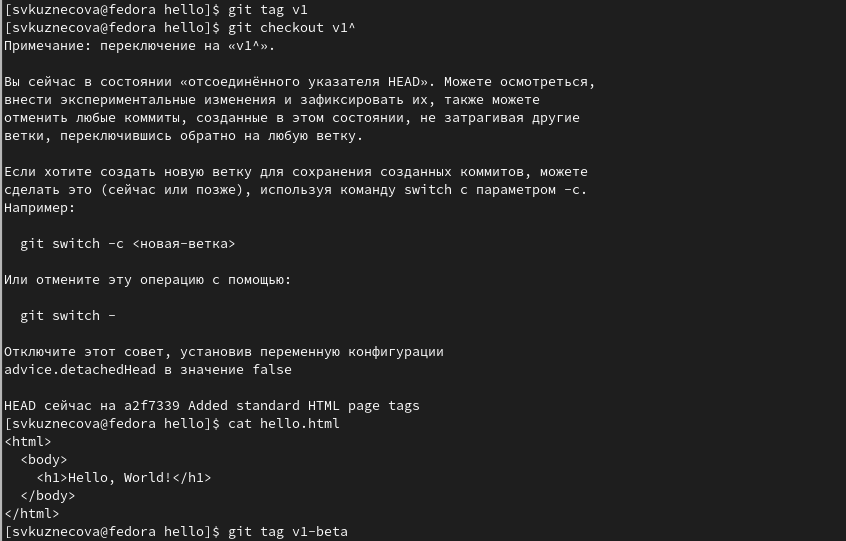


Хэши предыдущих версий

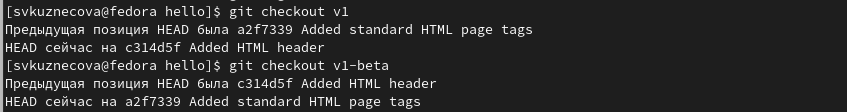


Хэш-код

Назовем текущую версию страницы hello первой (v1). Создадим тег первой версии. Для предыдущих версий создадим тег для версии, которая идет перед текущей версией и назовем его v1-beta. В первую очередь переключимся на предыдущую версию. Это версия c тегами html и body, но еще пока без head. Теперь сделаем ее версией v1-beta. Теперь попробуем попереключаться между двумя отмеченными версиями.

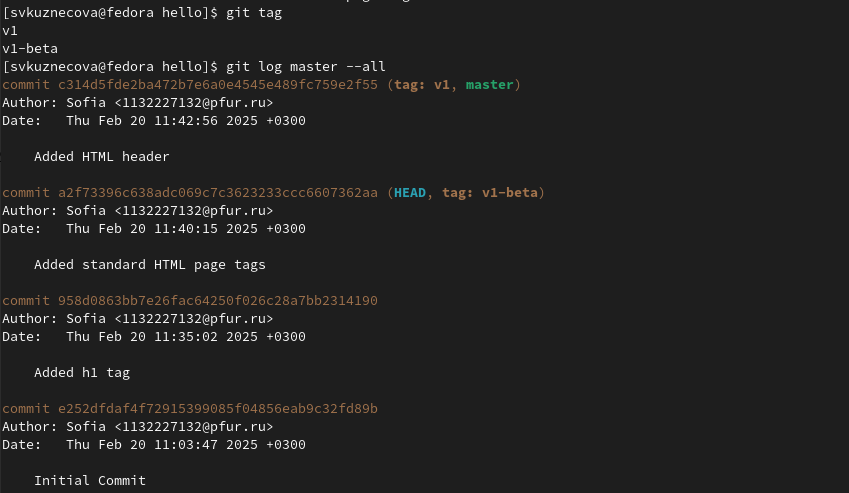


Теги



Перекоючение между двумя отмеченными версиями

Посмотрим, какие теги доступны, используя команду git tag. Также посмотрим теги в логе.



Теги в логе

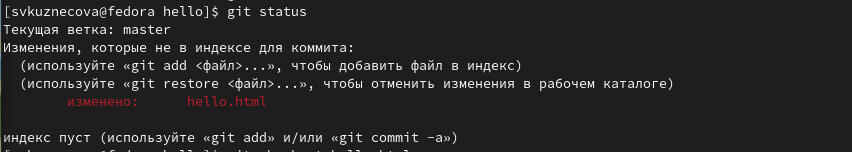
Убедимся, что мы находитесь на последнем коммите ветки master, прежде чем продолжить работу. Внесём изменение в файл hello.html в виде нежелательного комментария. Проверим состояние рабочего каталога. Используем команду git checkout для переключения версии файла hello.html в репозитории.

Коммите ветки master

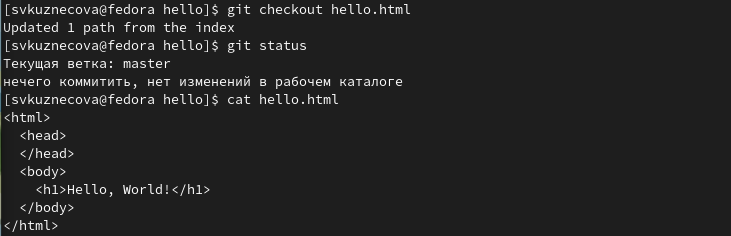
Коммите ветки master



Нежелательный комментарий

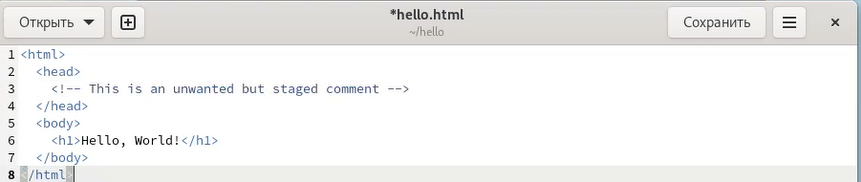


Состояние каталога



Команда git checkout

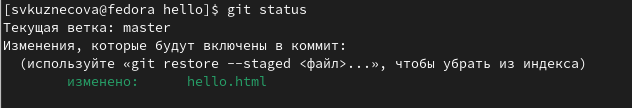
Внесём изменения в файл hello.html в виде нежелательного комментария. Проиндексируем это изменение. Проверим состояние нежелательного изменения. Переключимся на версию коммита. Переключимся на версию коммита и проверим статус.



Нежелательный комментарий

Нежелательный комментарий

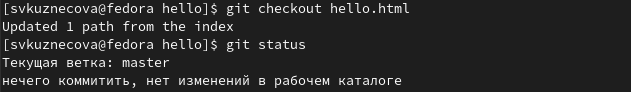
Нежелательный комментарий



Состояние нежелательного изменения

Отмена индексации изменения

Отмена индексации изменения

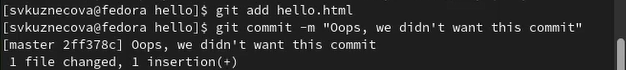


Версия коммита

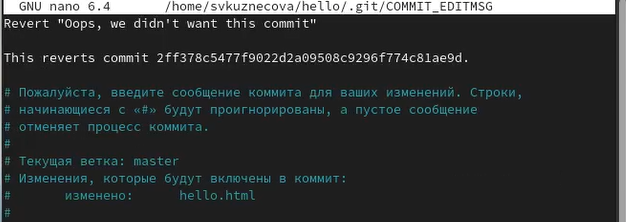
Изменим файл hello.html на следующий. Выполняем следующие команды.Отменим коммит. Проверим лога для просмотра нежелательные и отмененные коммиты в наш репозиторий.



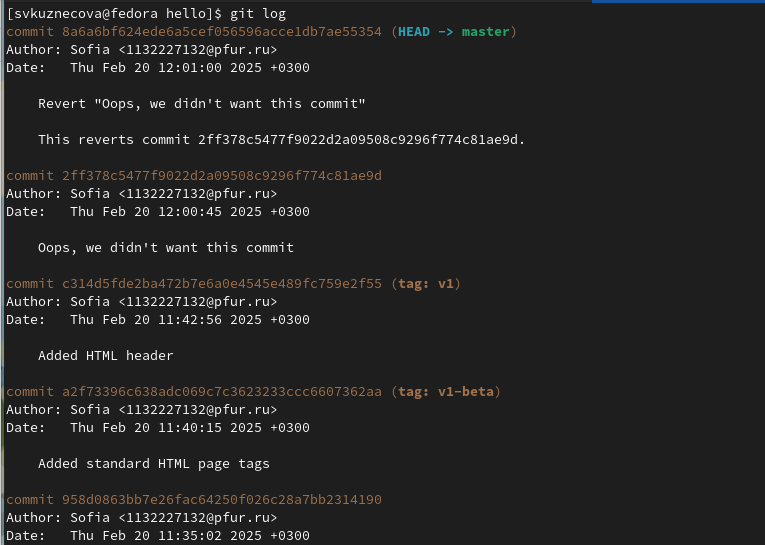
Изменение файла hello.html



Выполнение команд

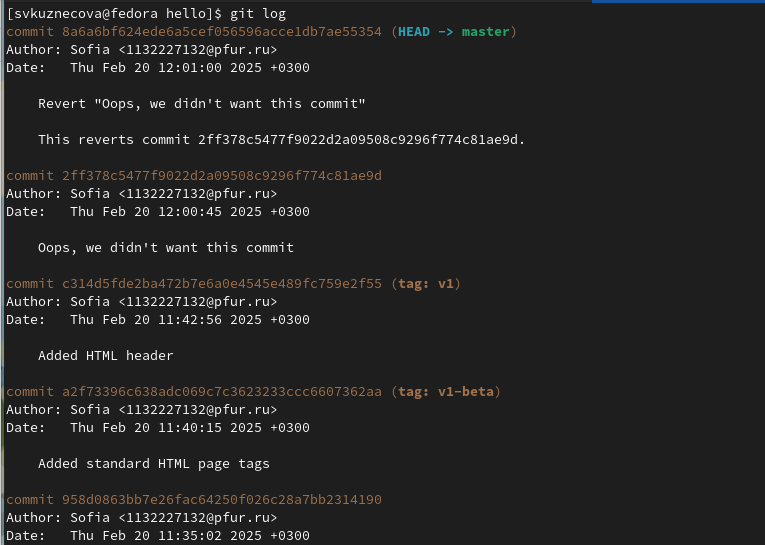


Отмена коммита



Проверка лога

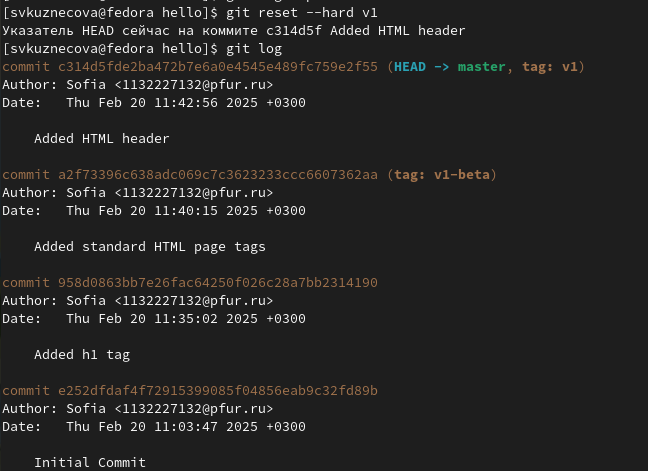
Сделаем быструю проверку истории коммитов. Отметим последний коммит тегом, чтобы потом можно было его найти. Глядя на историю лога, мы видим, что коммит с тегом «v1» является коммитом, предшествующим ошибочному коммиту. Сбросим ветку до этой точки.Посмотрим на все коммиты.



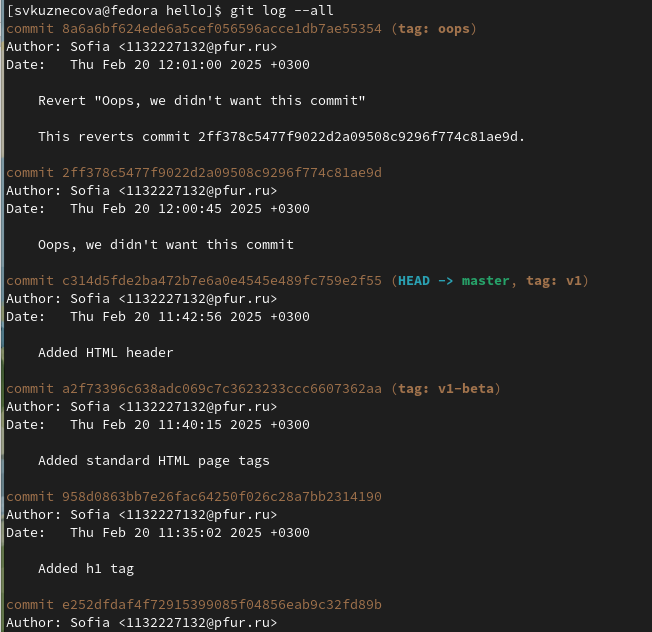
История коммитов

Отметка последнего коммита тегом

Отметка последнего коммита тегом

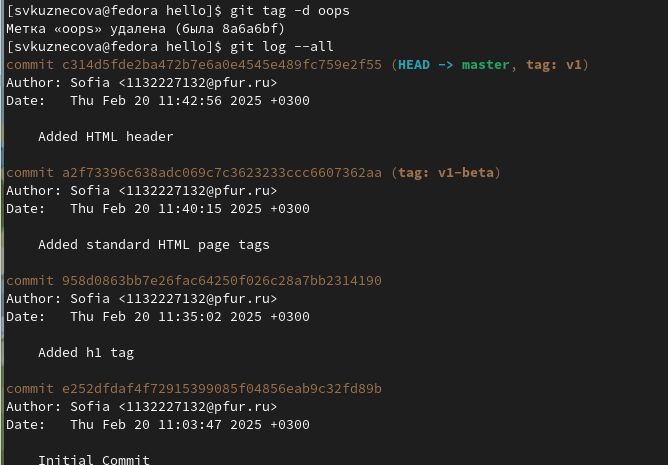


Сброс ветки



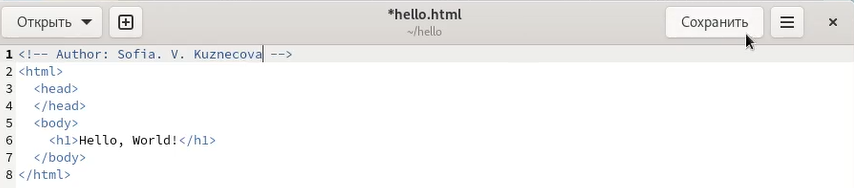
Просмотр коммитов

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

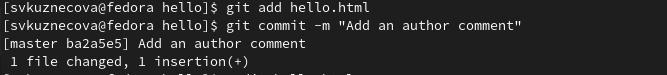


Тег oops

Добавим в страницу комментарий автора (вставьте свою фамилию). Сделаем коммит. Обновим страницу hello, включив в неё email. Изменим предыдущий коммит, включив в него адрес электронной почты. Сделаем быструю проверку нашей истории коммитов.



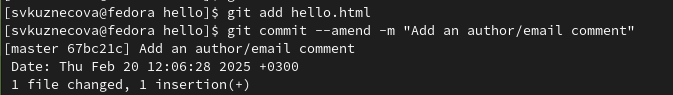
Комментарий автора



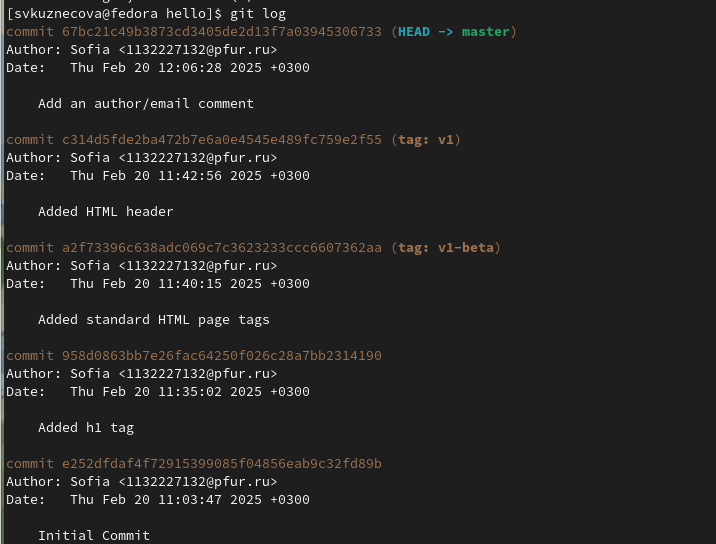
Коммит



Комментарий email

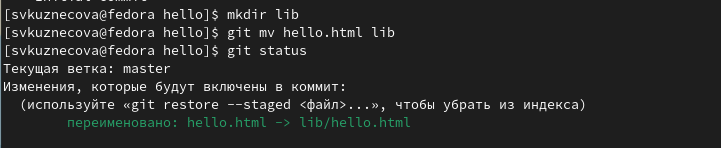


Изменения предыдущего коммита

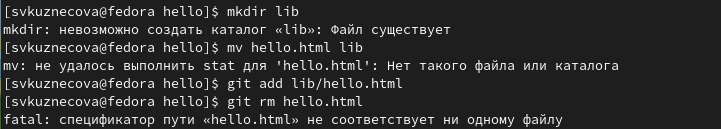


История коммитов

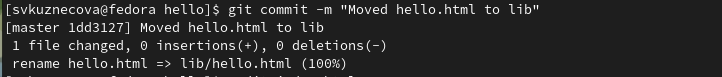
Создадим структуру репозитория. И перенесем страницу в каталог lib. Следующий набор команд идентичен последним действиям. Работы здесь побольше, но результат тот же. Сделаем коммит этого перемещения.



Каталог lib



Идентичный набор команд

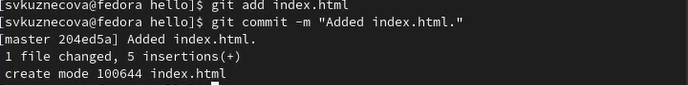


Коммит перемещения

Добавим файл index.html в наш репозиторий. Сделаем коммит.

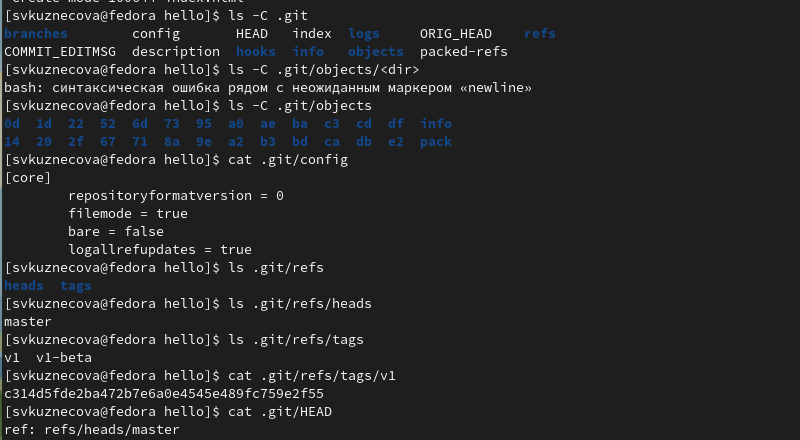


Добавление файла index.html



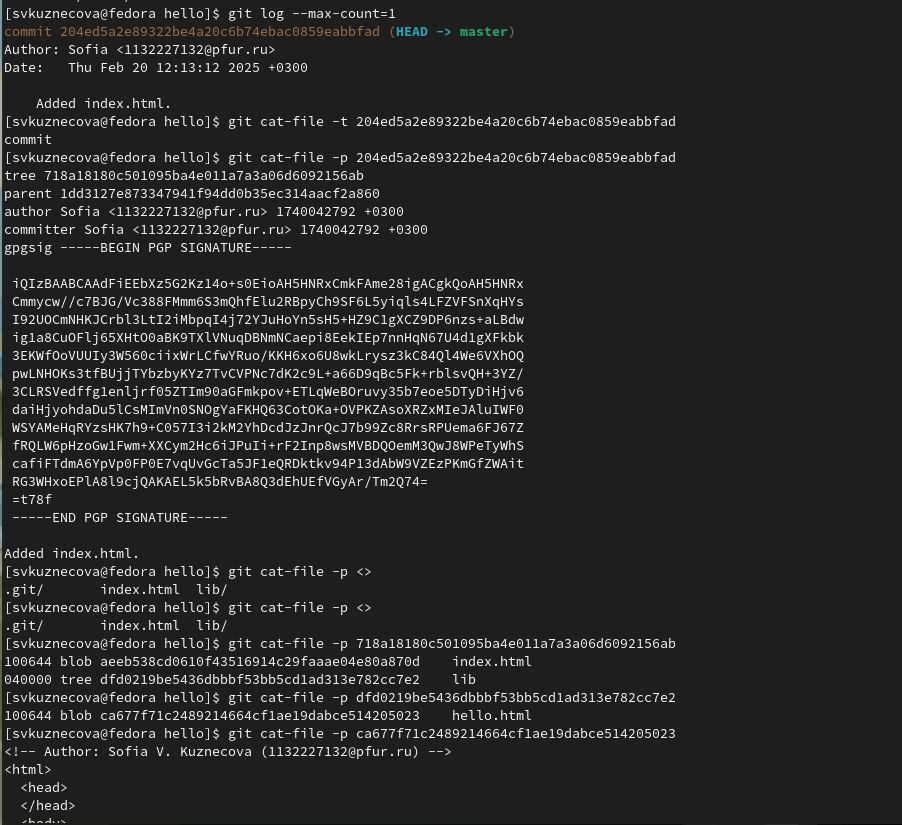
Добавление файла и создание коммита

Посмотрим каталог в котором хранится вся информация git. Увидим набор каталогов, имена которых состоят из 2 символов. Посмотрим в один из каталогов с именем из 2 букв. Также посмотрим файл конфигурации, создающийся для каждого конкретного проекта. Увидим файл HEAD, который содержит ссылку на текущую ветку, в данный момент это должна быть ветка master.



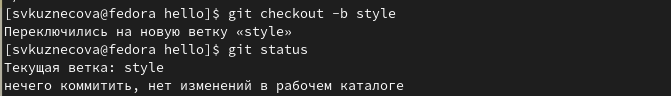
Просмотр каталога в котором хранится вся информация git

Посмотрим последний коммит в репозиторий. Выведем последний коммит. Выведем дерево каталогов, ссылка на который идет в коммите. Выведем каталог lib. Выведем файл hello.html.



Просмотр последнего коммита

Назовем новую ветку «style». Добавим файл стилей style.css. Создадим файл. Сделаем коммит.



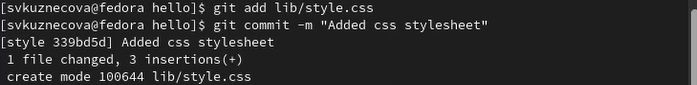
Ветка «style»

Файл стилей style.css

Файл стилей style.css



Редактируем файл

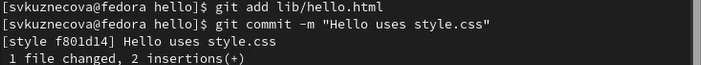


Коммит

Обновим файл hello.html, чтобы использовать стили style.css. Сделаем крммит. Обновим файл index.html, чтобы он тоже использовал style.css. Сделаем коммит.



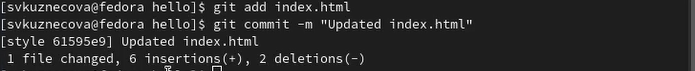
Обновление файла hello.html



Коммит

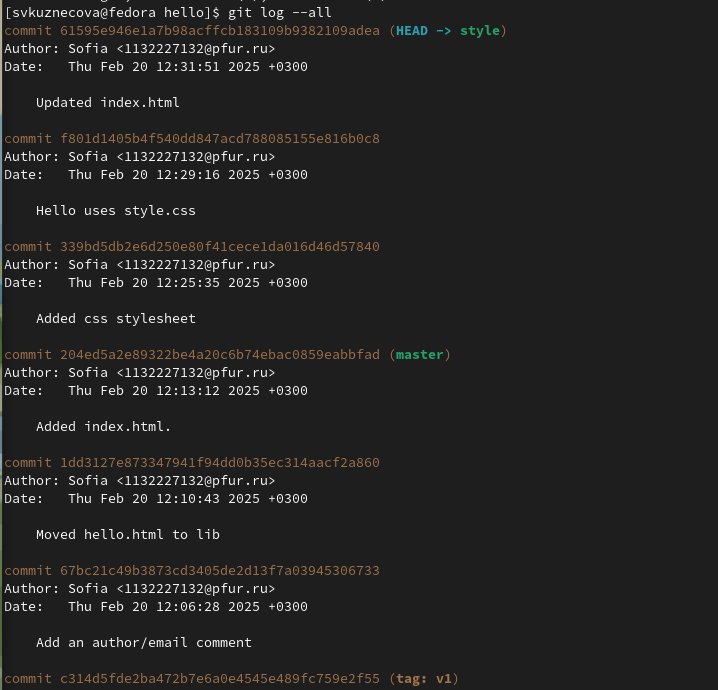


Обновление файла index.html

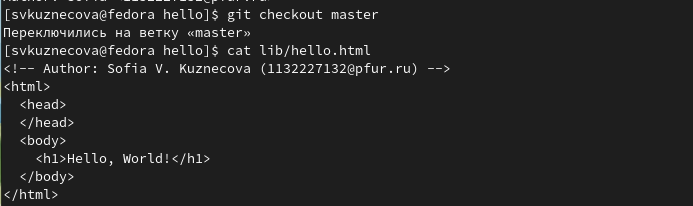


Коммит

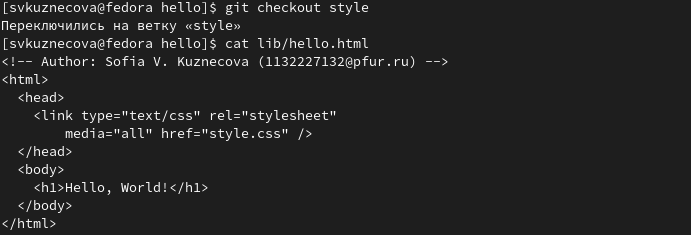
Просмотрим проект, который теперь имеет две ветки. Используем команду git checkout для переключения между ветками. Вернемся к ветке style.



Просмотр проекта



Команда git checkout



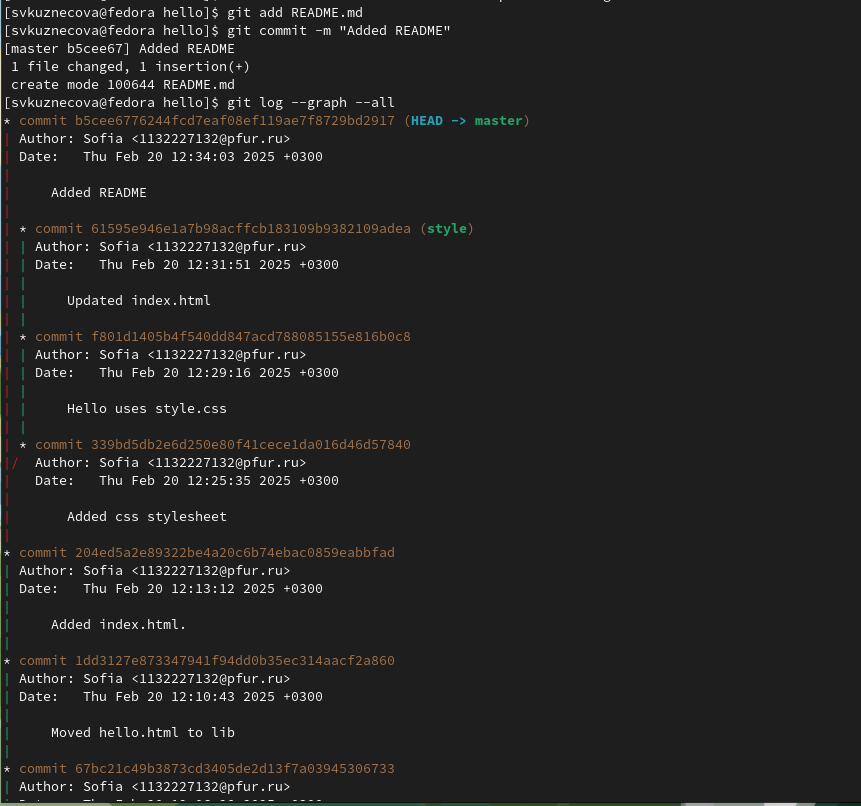
Ветка style

Создадим файла README в ветке master. Сделаем коммит изменений README.md в ветку master.

Файл README

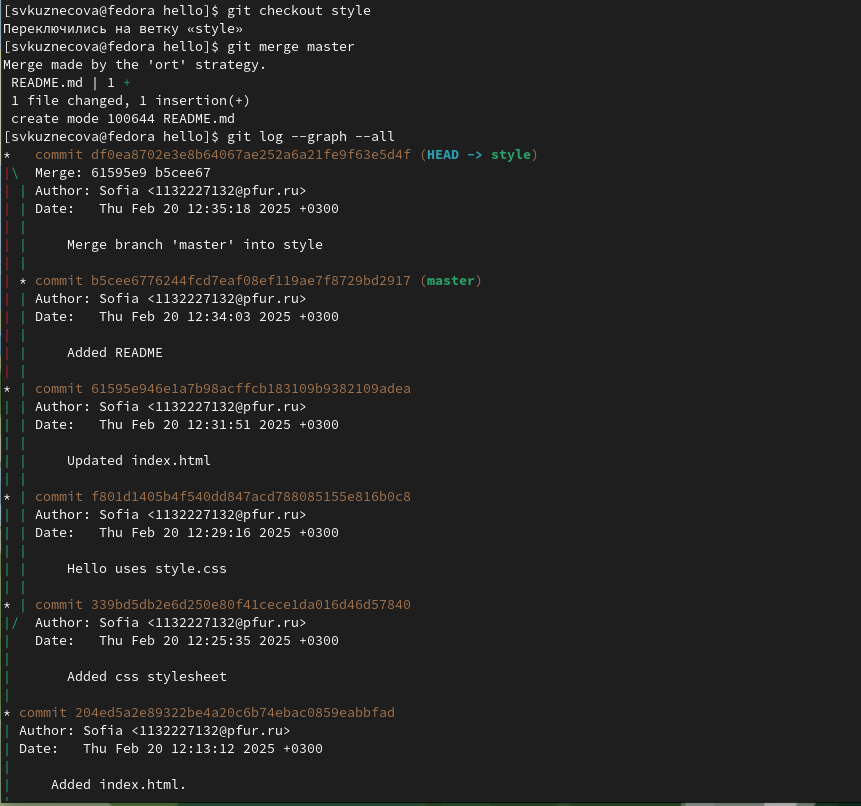
Файл README

Сделаем коммит. Просмотрим текущие ветки.



Коммит изменений README.md

Вернемся к ветке style и сольем master с style.

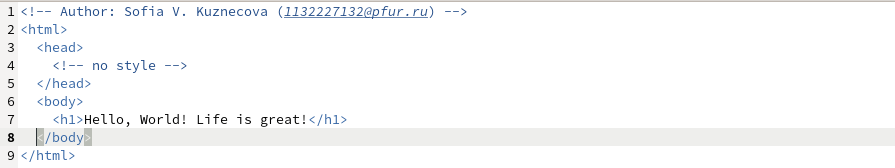


Солияние master с style

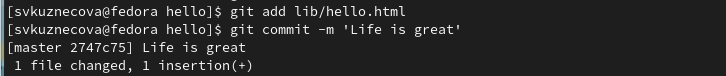
Вернёмся в ветку master и внесём следующие изменения. Сделаем коммит. Просмотрим ветки.

Ветку master

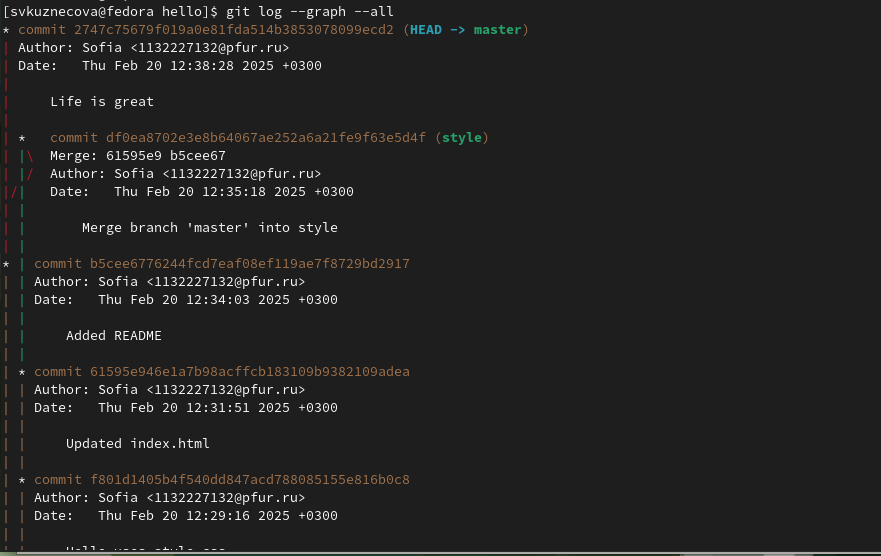
Ветку master



Изменения в ветке master

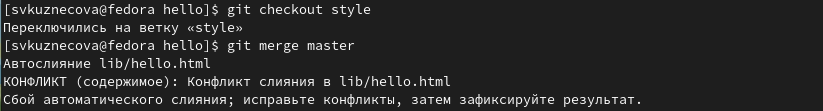


Коммит



Просмотр веток

Теперь вернемся к ветке style и попытаемся объединить ее с новой веткой master. Откроем файл lib/hello.html и увидим. Вручную разрешим конфликт. Внесём изменения в lib/hello.html для достижения результата. Сделаем коммит.



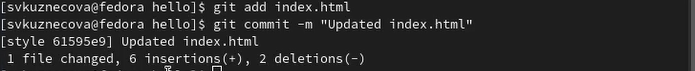
Слияние master с веткой style



Файл lib/hello.html

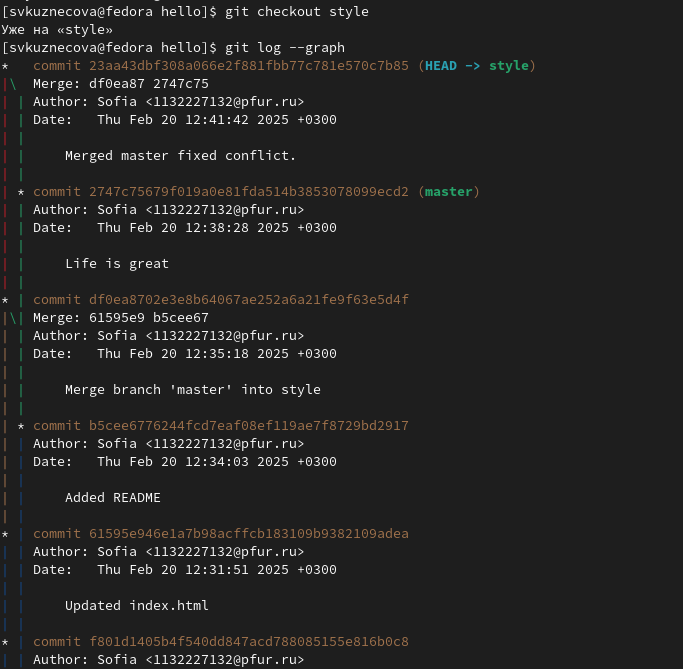


Разрешение конфликта



Коммит

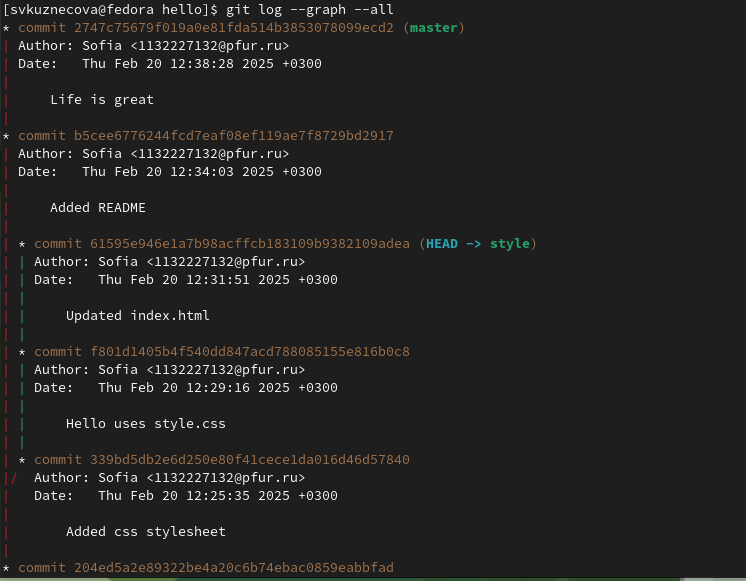
Находим последний коммит перед слиянием. Видим, что коммит «Updated index.html» был последним на ветке style перед слиянием. Поэтому сбросим ветку style к этому коммиту. Ищем лог ветки style. У нас в истории больше нет коммитов слияний.



Коммит перед слиянием

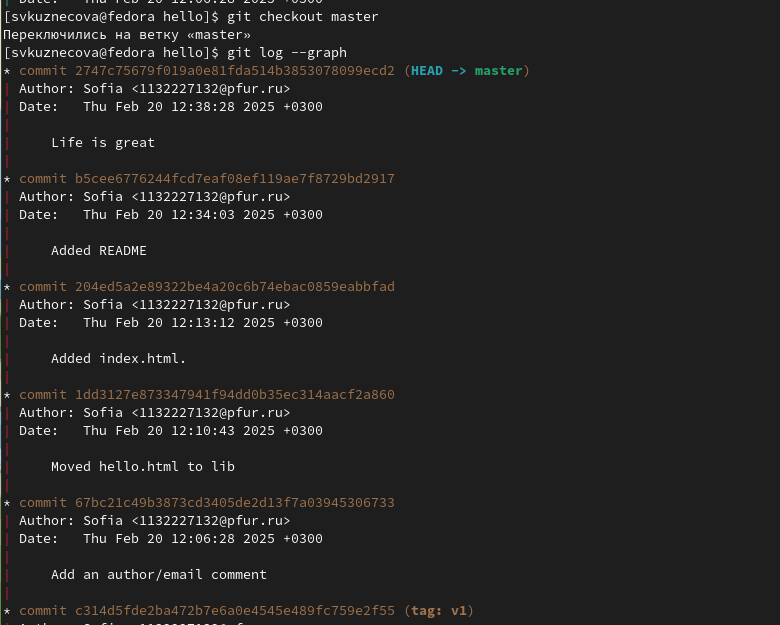
Updated index.html

Updated index.html

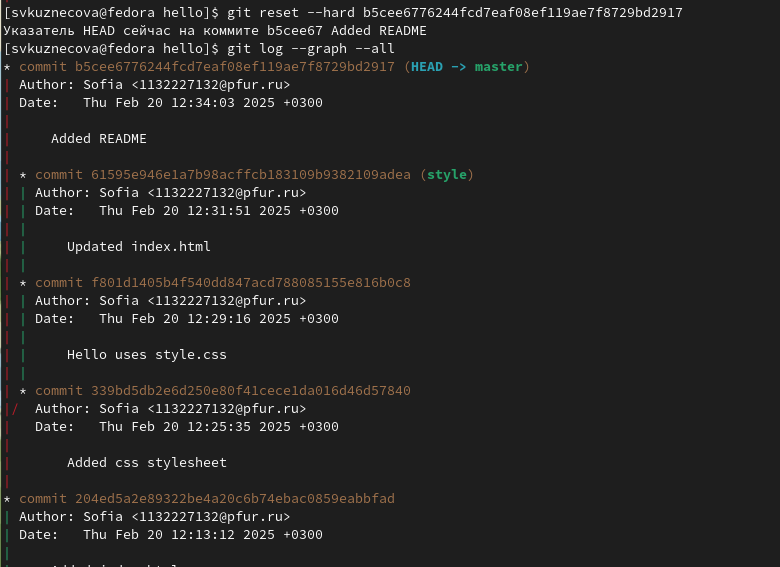


Лог ветки style

Добавив интерактивный режим в ветку master, мы внесли изменения, конфликтующие с изменениями в ветке style. Вернемся в ветке master в точку перед внесением конфликтующих изменений. Коммит «Added README» идет непосредственно перед коммитом конфликтующего интерактивного режима. Сбрасываем ветку master к коммиту «Added README».

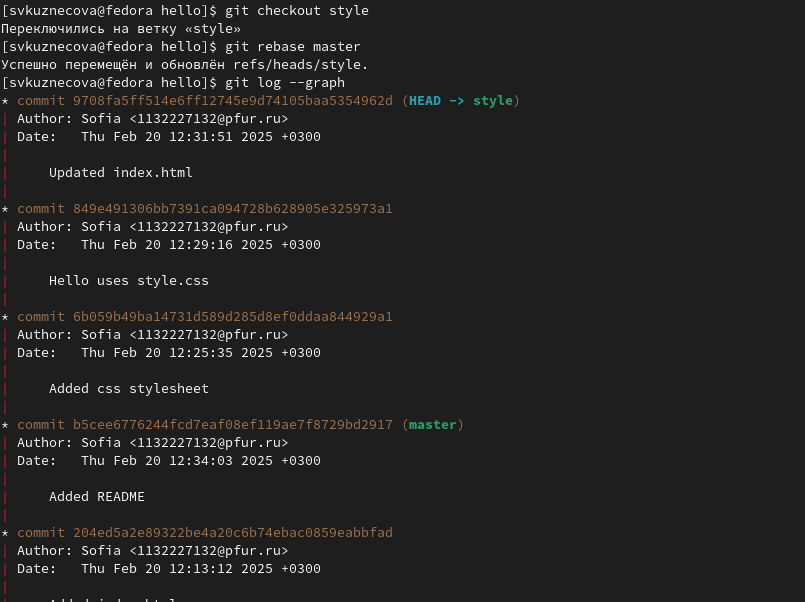


Интерактивный режим в ветке master



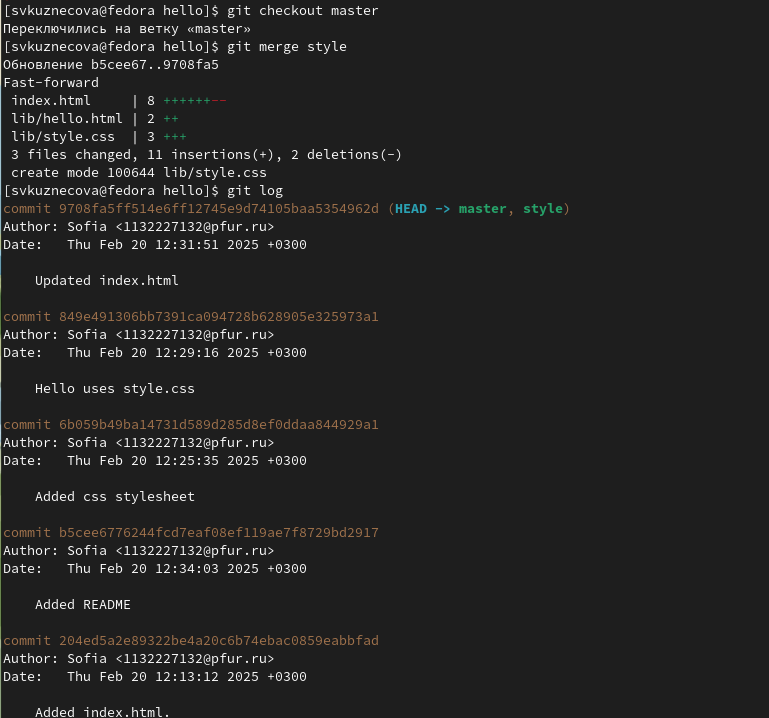
Коммит «Added README»

Используем команду rebase вместо команды merge. Вернувшись в точку до первого слияния хотим перенести изменения из ветки master в нашу ветку style. На этот раз для переноса изменений из ветки master используем команду git rebase вместо слияния.



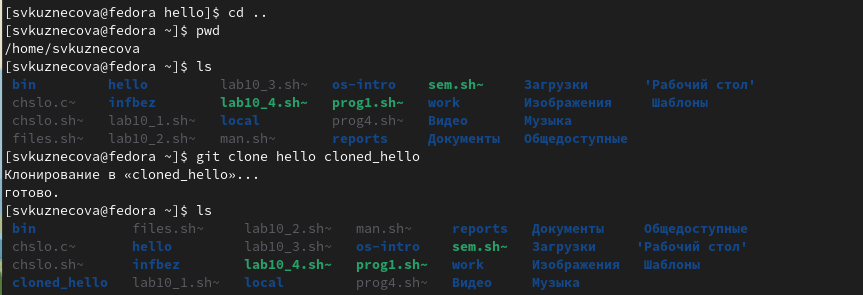
Команда git rebase

Сольём style в master. Просмотрим логи.



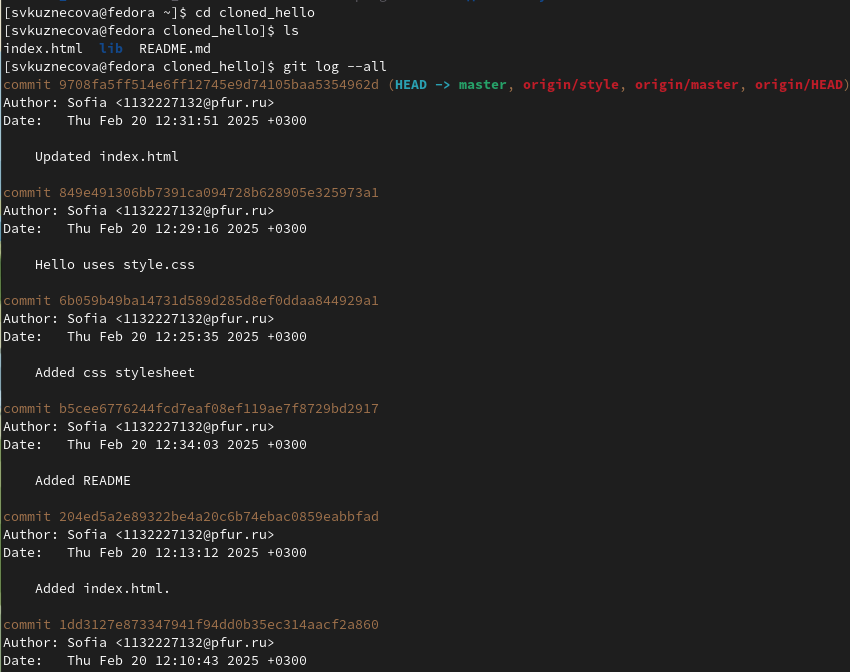
Style в master

Перейдём в рабочий каталог и создадим клон репозитория hello.



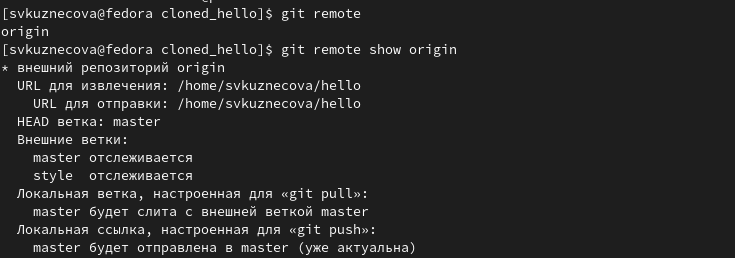
Клон репозитория hello

Взглянем на клонированный репозиторий и увидим список всех файлов на верхнем уровне оригинального репозитория README.md, index.html и lib. Просмотрим историю репозитория.



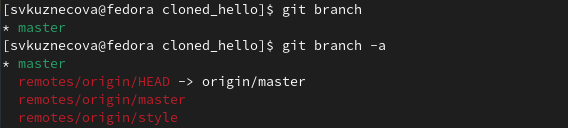
Клон

Мы видим, что клонированный репозиторий знает об имени по умолчанию удаленного репозитория. Посмотрим, можем ли мы получить более подробную информацию об имени по умолчанию(можем).



Оrigin

Посмотрим на ветки, доступные в нашем клонированном репозитории. Мы видим, в списке только ветка master. Команда git branch выводит только список локальных веток по умолчанию. Git выводит все коммиты в оригинальный репозиторий, но ветки в удаленном репозитории не рассматриваются как локальные.

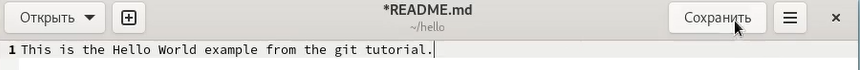


Удалённые ветки

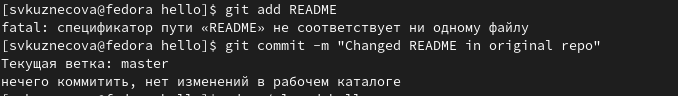
Войдём в репозиторий hello. Внесём изменения в файл README.md. Добавим это изменение и сделайте коммит. Извлечём изменения из удаленного репозитория. Проверим файл README.md и удостоверимся, что он не изменился.

Репозиторий hello

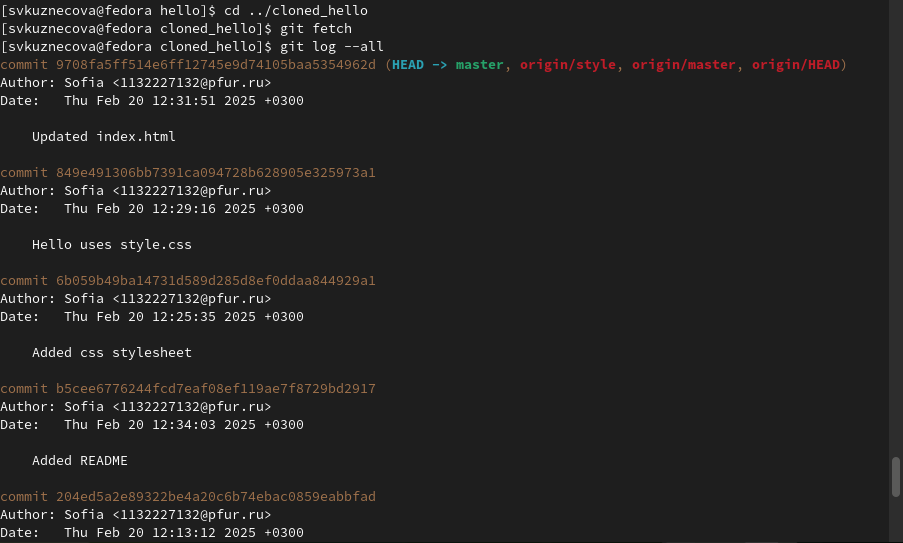
Репозиторий hello



Файл README.md



Коммит

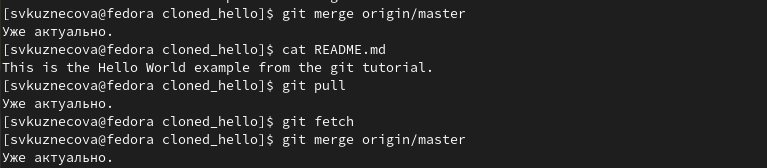


Извлечение изменений

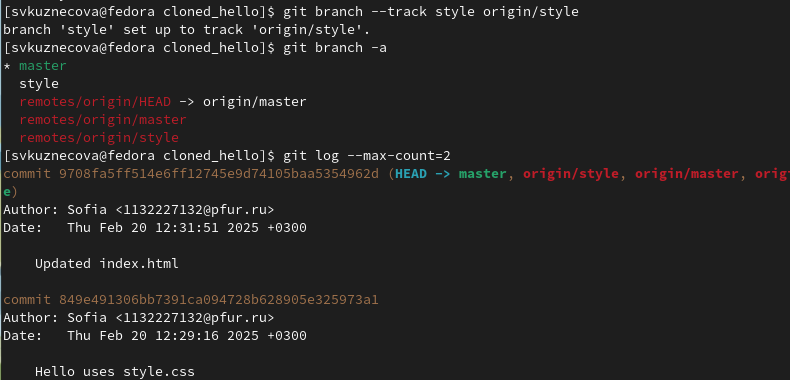
Проверка README.md

Проверка README.md

Ещё раз проверим файл README.md и видим изменения. Теперь давайте рассмотрим объединение fetch и merge в одну команду, эквивалентно двум следующим шагам. Добавим локальную ветку, которая отслеживает удаленную ветку. Теперь мы можем видеть ветку style в списке веток и логе.



Файл README.md



Локальная ветка

Создаём чистый репозиторий.



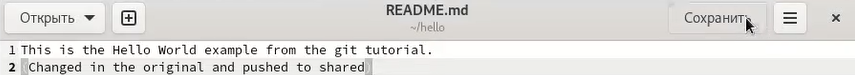
Чистый репозиторий

Добавим репозиторий hello.git к нашему оригинальному репозиторию.

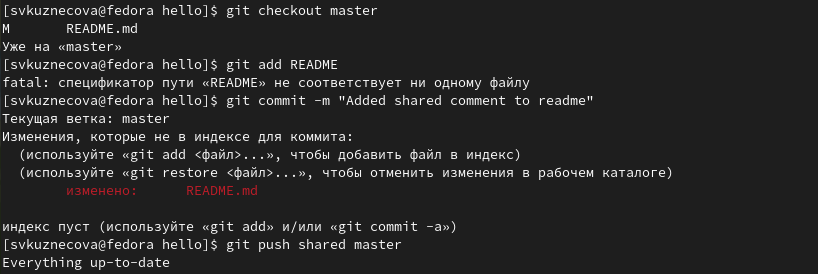
Репозиторий hello.git

Репозиторий hello.git

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. Отредактируйте файл README.md и сделайте коммит. Теперь отправьте изменения в общий репозиторий.

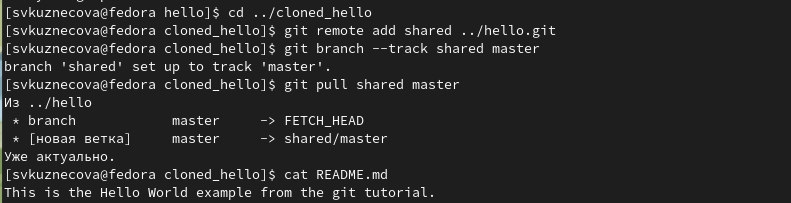


Файл README.md



Коммит

Извлечём изменения из общего репозитория. Быстро переключимся в клонированный репозиторий и извлечём изменения, только что отправленные в общий репозиторий.



Клонированный репозиторий

# Выводы

В ходе выполнения лабораторной работы научилась работать с git.