Работа с git

Лабораторная работа № 1

Шулуужук Айраана НПИбд-02-22

Содержание

# 1 Цель работы

Научиться пользоваться с git и разобраться с его особенностями.

# 2 Выполнение лабораторной работы

Выполним следующие команды, чтобы git узнал наше имя и электронную почту (рис. 1)

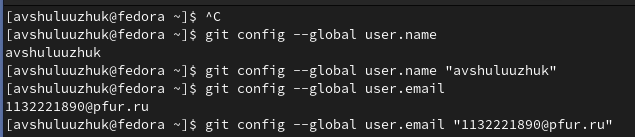


Рис. 1: имя пользователя и его электоронная почта

Установим параметры установки окончаний строк и отображения unicode (рис. 2)

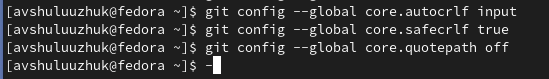


Рис. 2: параметры установки окончаний строк и отображения unicode

Начнем работу в пустом рабочем каталоге с создания пустого каталога с именем hello, затем войдем в него и создадим там файл с именем hello.html (рис. 3)

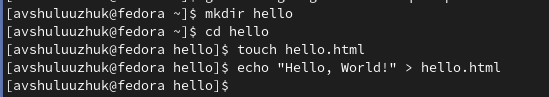


Рис. 3: создание рабочего каталога и необходимых файлов

Создаим репозиторий из этого каталога и добаввим файл в него. Далее проверим состояние репозитория (рис. 4).

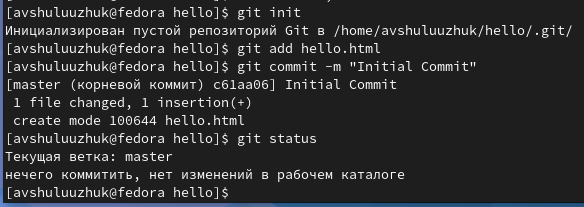


Рис. 4: просмотр состояние репозитория

Добавим HTML-теги к нашему приветствию. Изменим содержимое файла hello.html (рис. 5). Проверим состояние рабочего каталога (рис. 6)

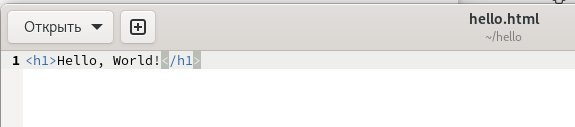


Рис. 5: файл hello.html

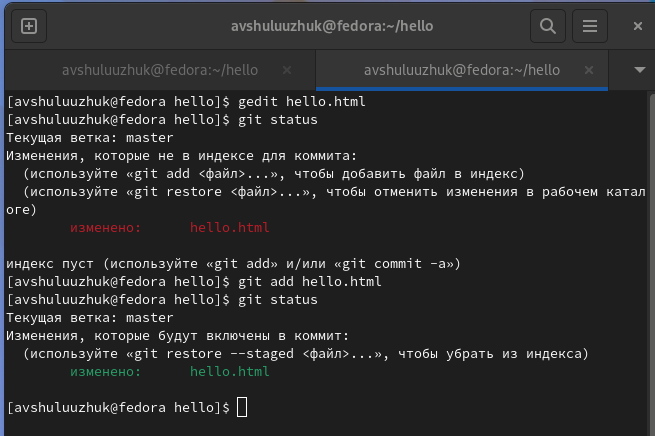


Рис. 6: проверка состояние каталога

Теперь выполним команду git, чтобы проиндексировать изменения. Проверим его состояние. Далее сделаем коммит и проверим состояние (рис. 7).

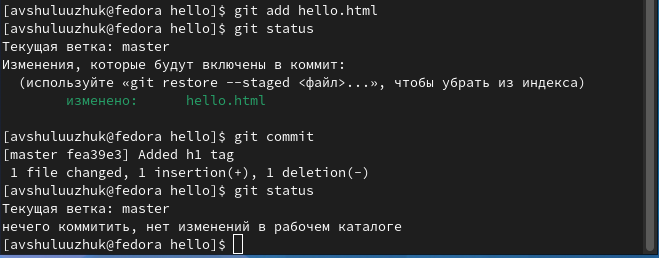


Рис. 7: индексация изменений

Изменим страницу «Hello, World», чтобы она содержала стандартные теги html и body. И добавим это изменение в индексе git (рис. 8).

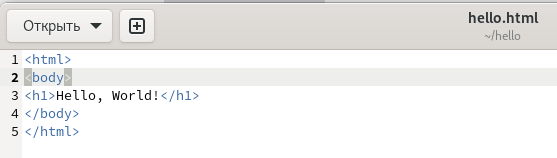


Рис. 8: файл hello.html

Теперь добавим заголовки HTML (секцию head) к странице «Hello, World» (рис. 9).

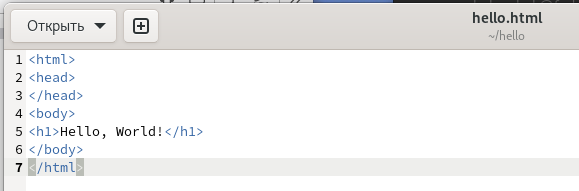


Рис. 9: файл hello.html

Проверим текущий статус (рис. 10).

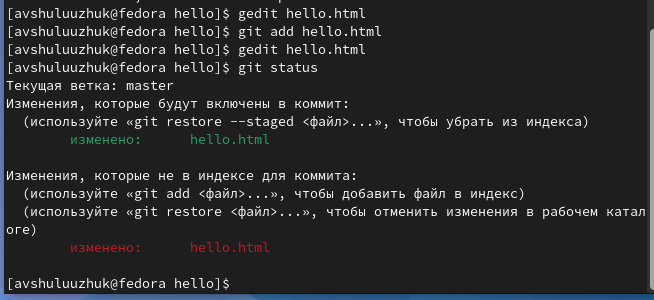


Рис. 10: проверка текущего статуса

Произведем коммит проиндексированного изменения (значение по умолчанию), а затем еще раз проверим состояние. Состояние команды говорит о том, что hello.html имеет незафиксированные изменения, но уже не в буферной зоне. 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 (рис. 11).

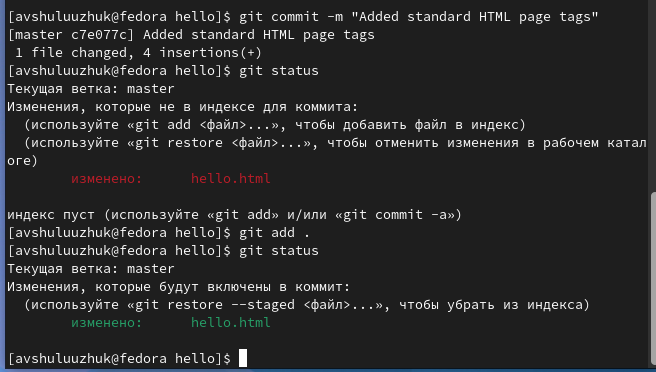


Рис. 11: произведение коммита проиндексированного изменения

Получим список произведенных изменений (рис. 12).

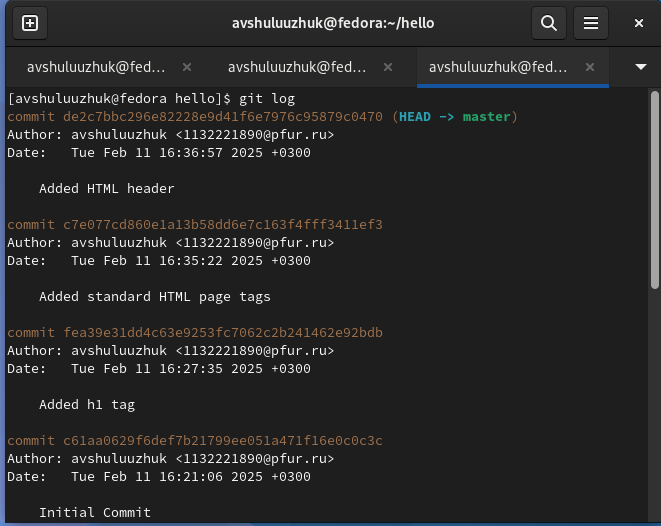


Рис. 12: список произведенных изменений

Варианты отображения лога (рис. 13).

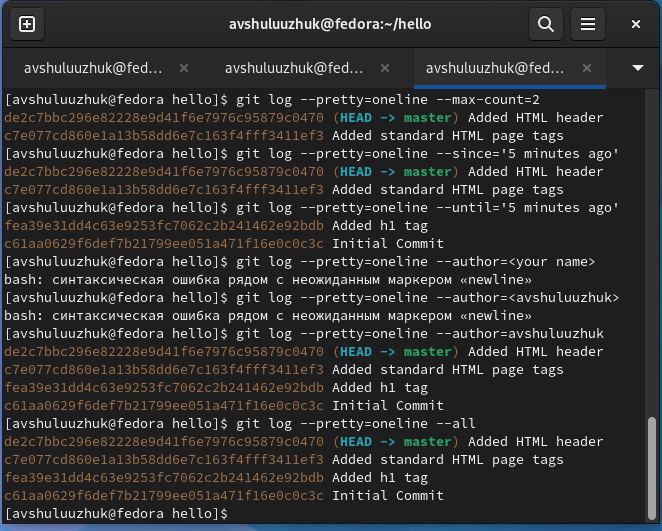


Рис. 13: варианты отображение лога

Изучим данные лога и найдем хэш для первого коммита. Используем этот хэш-код (достаточно первых 7 знаков). Затем проверим содержимое файла hello.html (рис. 14).

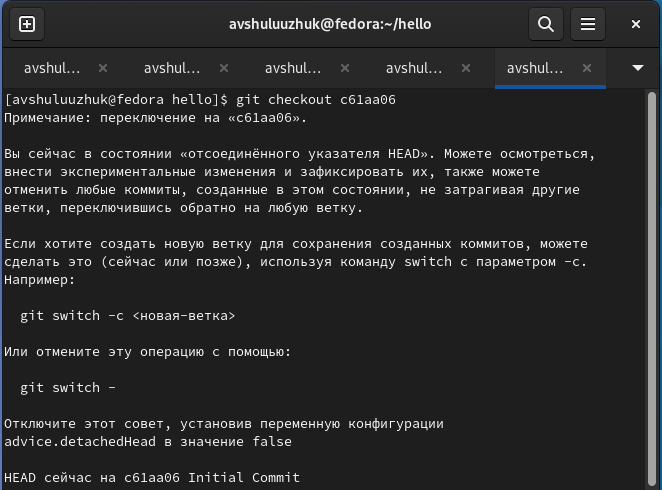


Рис. 14: хэш для первого коммита

Вернемся к последней версии в ветке master (рис. 15).

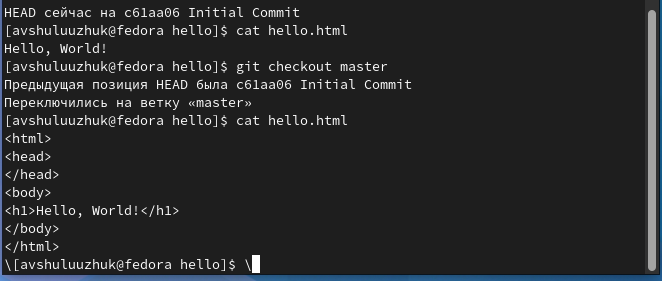


Рис. 15: последняя версия в ветке master

Назовем текущую версию страницы hello первой (v1). Создадим тег первой версии. Теперь текущая версия страницы называется v1. Создадим тег для версии, которая идет перед текущей версией и назовем его v1-beta (рис. 16).

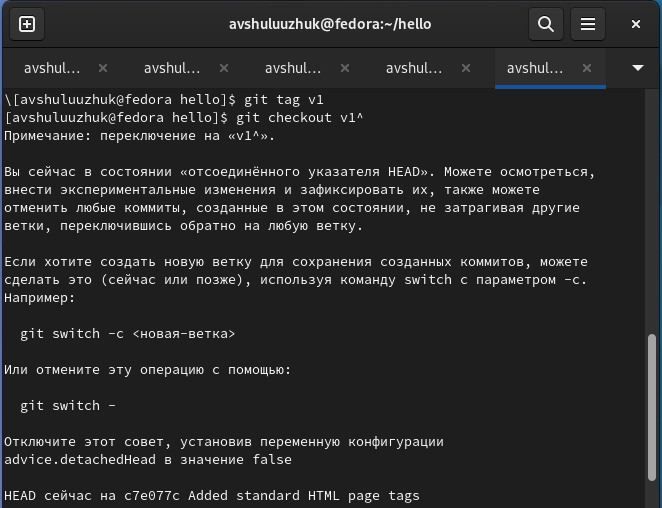


Рис. 16: создание тега первой версии

Просмотрим файл hello.html (рис. 17).

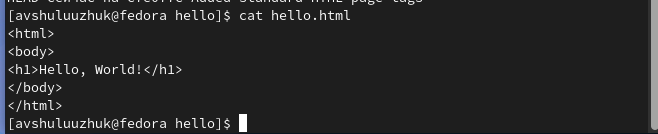


Рис. 17: файл hello.html

Это версия c тегами

и

, но еще пока без

. Сделаем ее версией v1-beta. Попробуем попереключаться между двумя отмеченными версиями (рис. 18).

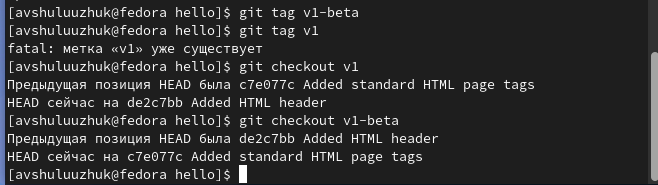


Рис. 18: переключение между двумя отмеченными версиями

Убедимся, что мы находимся на последнем коммите ветки master, прежде чем продолжить работу. Внесем изменение в файл hello.html в виде нежелательного комментария. Проверим состояние рабочего каталога. Мы видим, что файл hello.html был изменен, но еще не проиндексирован (рис. 19) (рис. 20).

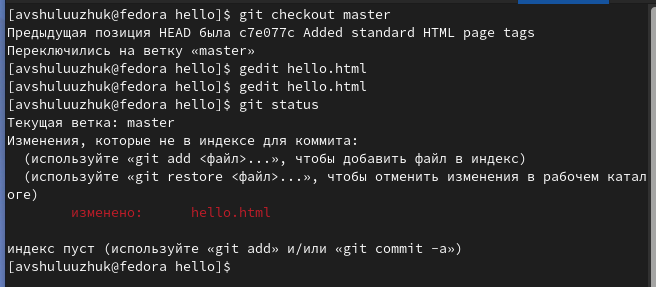


Рис. 19: отмена локальных изменений до индексации

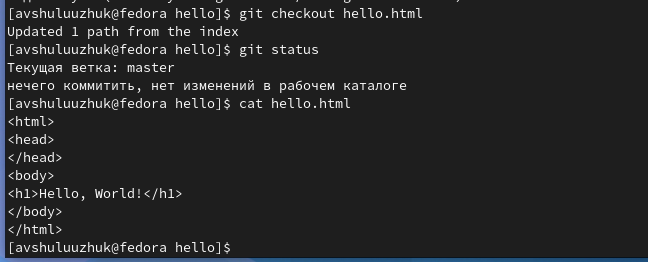


Рис. 20: отмена изменений в рабочем каталоге

Внесем изменение в файл hello.html в виде нежелательного комментария и проиндексируем это изменение. Проверим состояние нежелательного изменения (рис. 21).

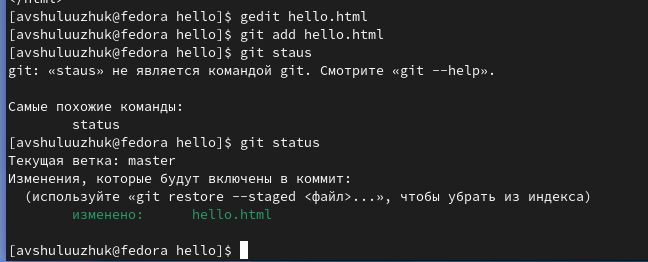


Рис. 21: изменение файла и проиндексирование изменений

Вывод состояния показывает нам именно то, что мы должны сделать для отмены индексации изменения. Переключимся на версию коммита и проверим состояние каталога. Видим, что рабочий каталога снова чист (рис. 22)

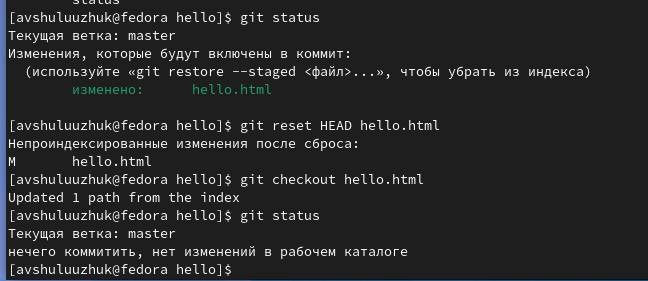


Рис. 22: сброс буферной зоны

Изменим файл hello.html и сделаем коммит (рис. 23)

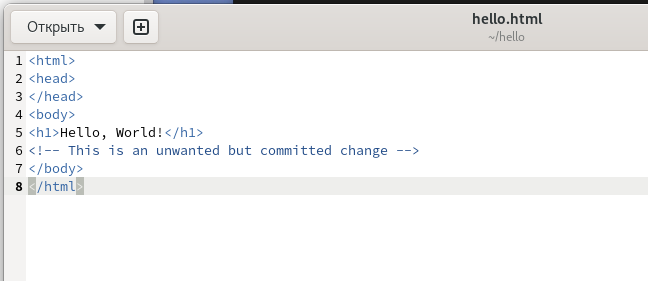


Рис. 23: файл hello.html

Чтобы отменить коммит, нам необходимо сделать коммит, который удаляет изменения, сохраненные нежелательным коммитом. Проверим лог (рис. 24)

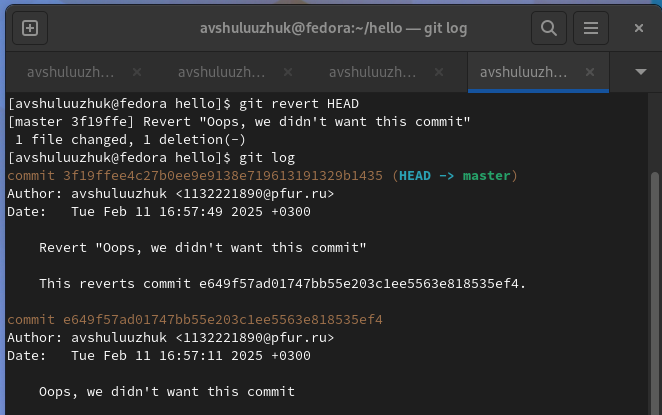


Рис. 24: выполнение коммита с новыми изменениями, отменяющими предыдущие

Сделаем быструю проверку нашей истории коммитов. Мы видим, что два последних коммита в этой ветке — «Oops» и «Revert Oops». Удалим их с помощью сброса. Для начала отметим эту ветку. Но прежде чем удалить коммиты, отметим последний коммит тегом, чтобы потом можно было его найти. Сбросим ветку до этой точки. Поскольку ветка имеет тег, мы можем использовать имя тега в команде сброса (рис. 25).

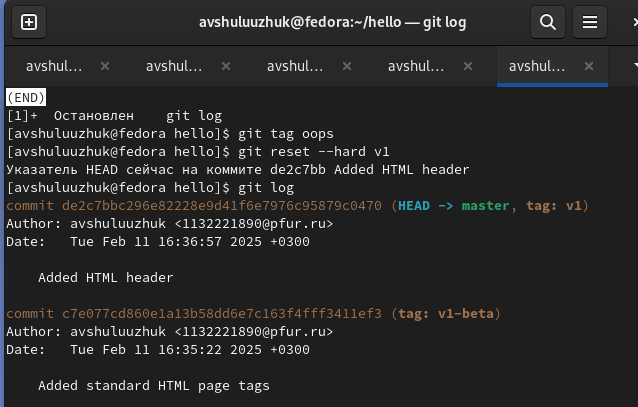


Рис. 25: брос коммитов к предшествующим коммиту Oops

Тег oops свою функцию выполнил. Удалим его и коммиты, на которые он ссылался, сборщиком мусора. Тег «oops» больше не будет отображаться в репозитории (рис. 26).

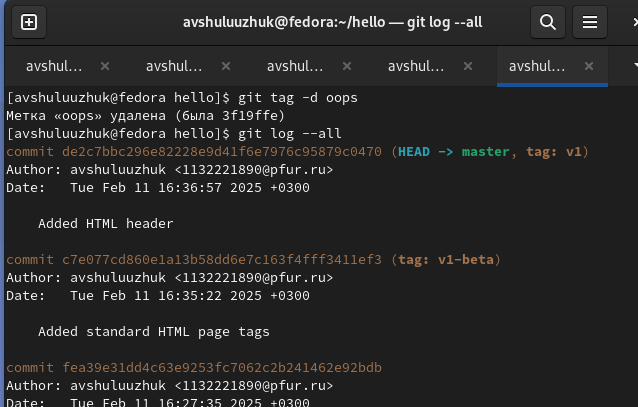


Рис. 26: удаление тега oops

Изменим страницу, а затем сделаем коммит. Добавим в страницу комментарий автора (вставим свою фамилию) (рис. 27).

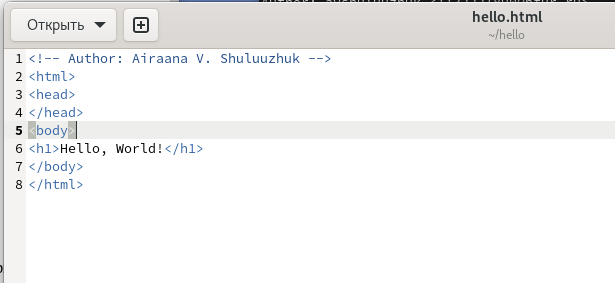


Рис. 27: файл hello.html

После совершения коммита мы понимаем, что любой хороший комментарий должен включать электронную почту автора. Обновите страницу hello, включив в нее email. (рис. 28).

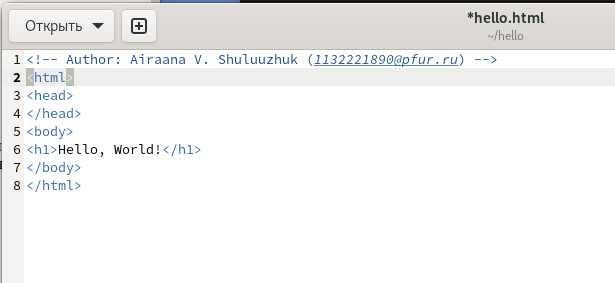


Рис. 28: файл hello.html

Изменим предыдущий коммит, включив в него адрес электронной почты (рис. 29).

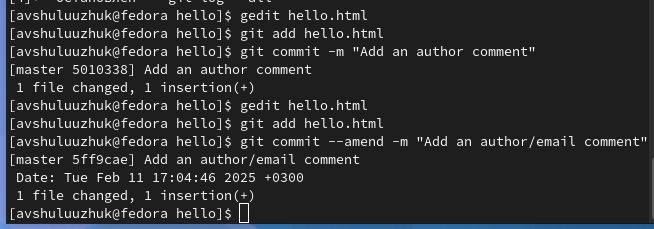


Рис. 29: изменение предыдущего коммита

Переместим файл hello.html в каталог lib. Создадим структуру нашего репозитория. Перенесем страницу в каталог lib (рис. 30).

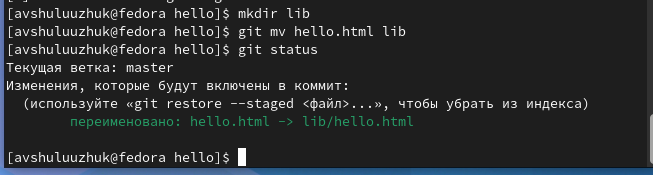


Рис. 30: создание каталога

Сделаем коммит этого перемещения (рис. 31).

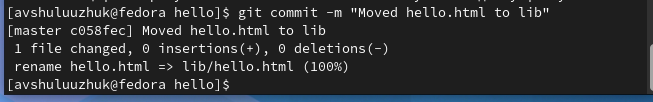


Рис. 31: коммит в новый каталог

Добавим файл index.html в наш репозиторий и сделаем коммит (рис. 32) (рис. 33).



Рис. 32: файл index.html

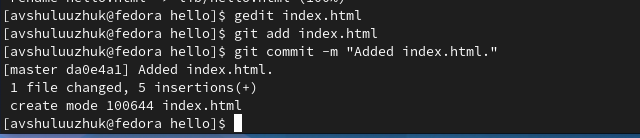


Рис. 33: выполнение коммита

Теперь при открытии index.html, мы видим кусок страницы hello в маленьком окошке (рис. 34).

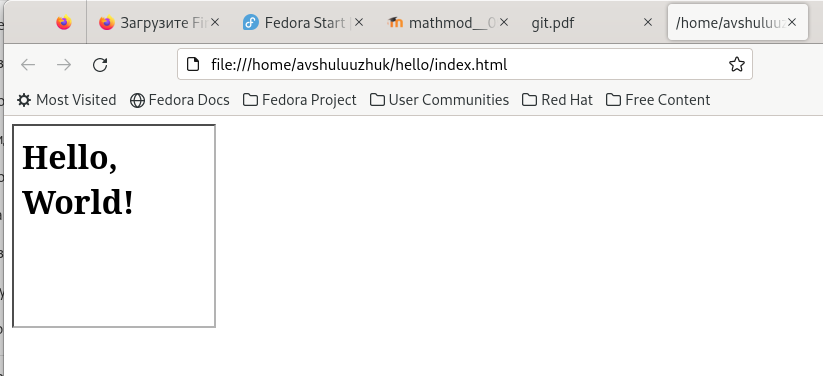


Рис. 34: index.html

Каталог .git. Это каталог, в котором хранится вся информация git. Просмотрим набор каталогов, имена которых состоят из 2 символов. Имена каталогов являются первыми двумя буквами хэша sha1 объекта, хранящегося в git (рис. 35).

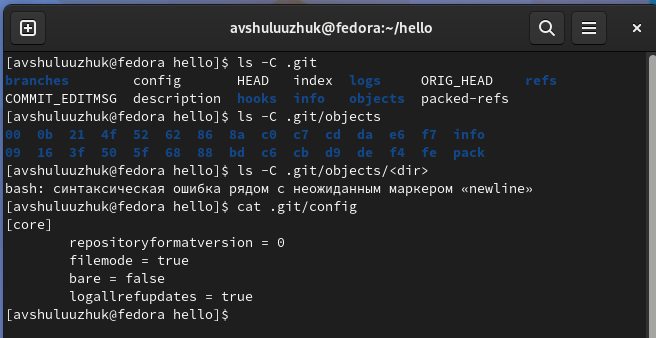


Рис. 35: каталог .git

Просмотрим файл конфигурации. Это файл конфигурации, создающийся для каждого конкретного проекта. Записи в этом файле будут перезаписывать записи в файле .gitconfig главного каталога, по крайней мере в рамках этого проекта. (рис. 36).

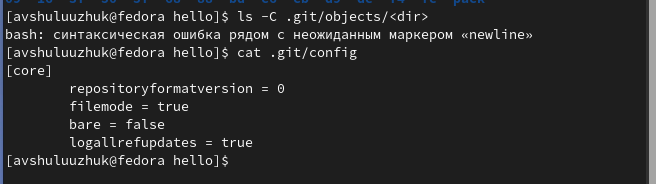


Рис. 36: файл конфигурации

Просмотрим ветки и теги. Каждый файл соответствует тегу, ранее созданному с помощью команды git tag. Его содержание — это всего лишь хэш коммита, привязанный к тегу (рис. 37).

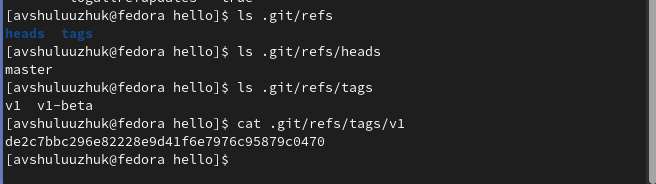


Рис. 37: ветки и теги

Работа непосредственно с объектами git. Просмотрим последний коммит. Эта команда должна показать последний коммит в репозиторий и выведем последний коммит с помощью SHA1 хэша (рис. 38).

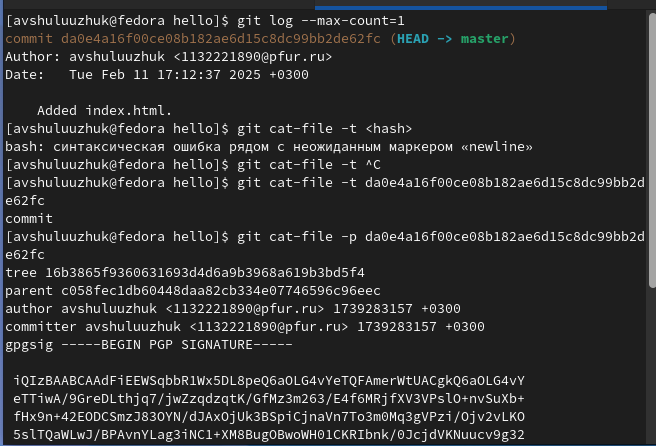


Рис. 38: поиск последнего коммита

Выведем дерево каталогов, ссылка на который идет в коммите. Это должно быть описание файлов (верхнего уровня) в нашем проекте (для конкретного коммита). Используем SHA1 хэш из строки «дерева», из списка выше. Выведем каталога lib и файл hello.html (рис. 39)

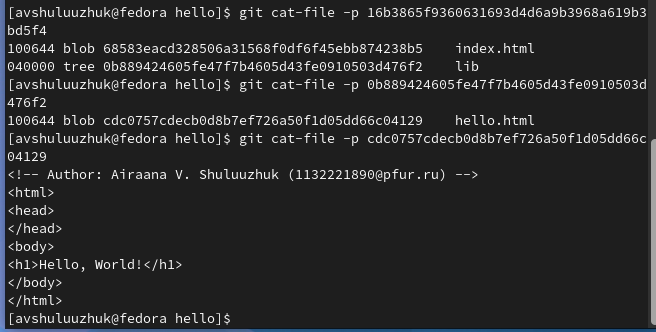


Рис. 39: вывод дерева каталогов

Создадим ветку. Назовем нашу новую ветку «style». Добавим изменения в файл стилей и выполним коммит (рис. 40).

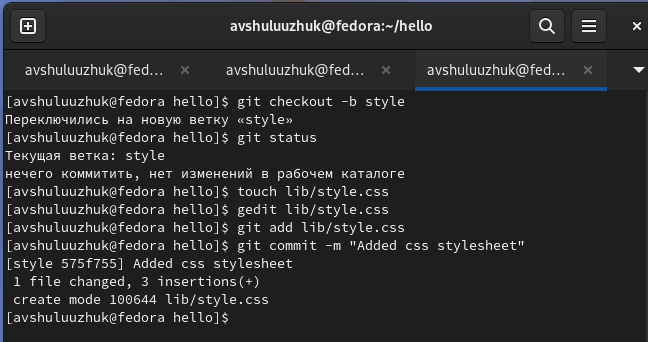


Рис. 40: создание новой ветки

Обновим файл hello.html, чтобы использовать стили style.css и выполним коммит (рис. 41).

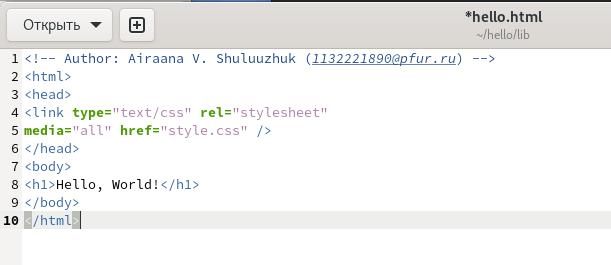


Рис. 41: файл hello.html

Обновим файл index.html, чтобы он тоже использовал style.css и выполним коммит (рис. 42)



Рис. 42: index.html

Теперь в нашем проекте есть две ветки (рис. 43)

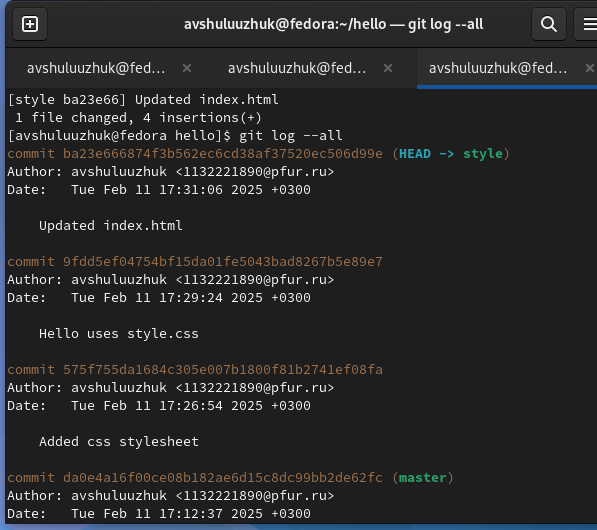


Рис. 43: навигация по веткам

Используем команду git checkout для переключения между ветками. Мы находимся на ветке master. Это заметно по тому, что файл hello.html не использует стили style.css (рис. 44)

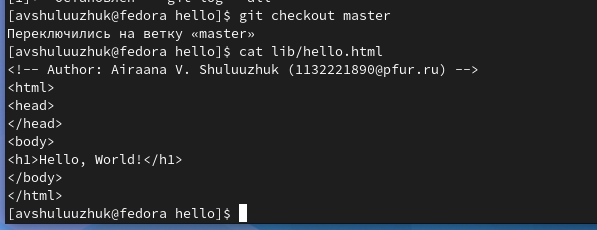


Рис. 44: Переключение на ветку master

Вернемся к ветке style. Содержимое lib/hello.html подтверждает, что мы вернулись на ветку style (рис. 45).

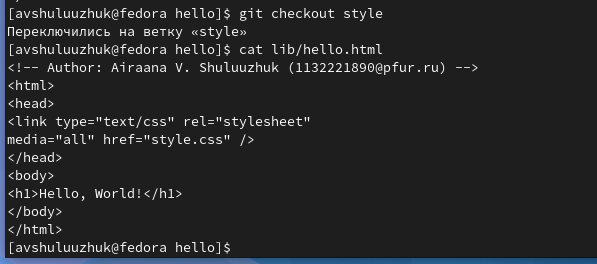


Рис. 45: переход к ветке style

Теперь у нас в репозитории есть две отличающиеся ветки. Используем следующую лог-команду для просмотра веток и их отличий (рис. 46).

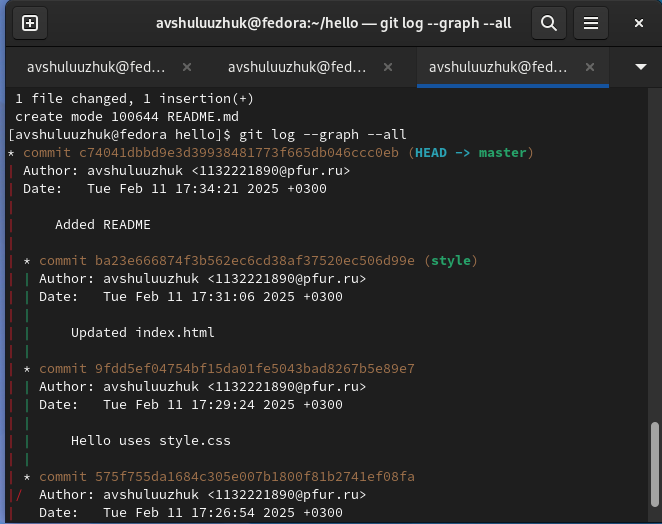


Рис. 46: просмотр текущих веток

Теперь вернемся к ветке style и попытаемся объединить ее с новой веткой master. Если мы откроем lib/hello.html, мы увидим (рис. 47).

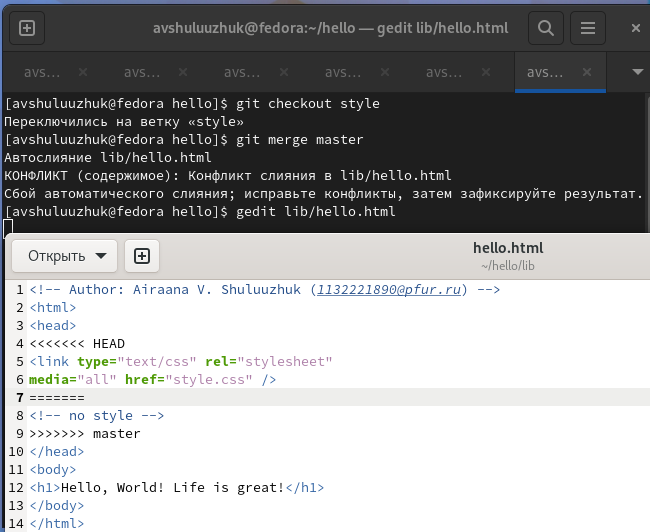


Рис. 47: слияние master с веткой style

Необходимо вручную разрешить конфликт. Внесем изенения в lib/hello.html для достижения следующего результата и сделаем коммит решения коммита (рис. 48).

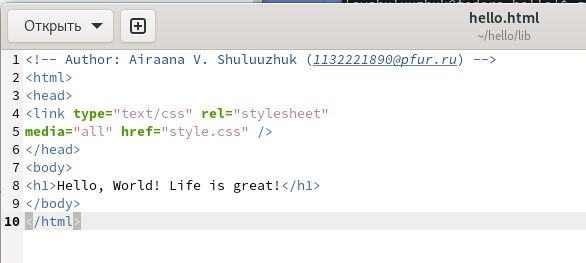


Рис. 48: разрешение конфликта

Для сброса ветки style нам необходимо найти последний коммит перед слиянием (рис. 49).

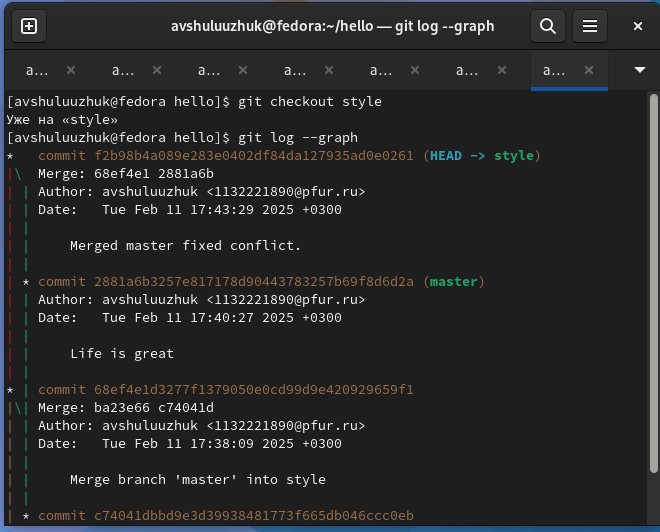


Рис. 49: сброс ветки style

Мы видим, что коммит «Updated index.html» был последним на ветке style перед слиянием. Cбросим ветку style к этому коммиту. Проверим ветку. Поищем лог ветки style. У нас в истории больше нет коммитов слияний (рис. 50).

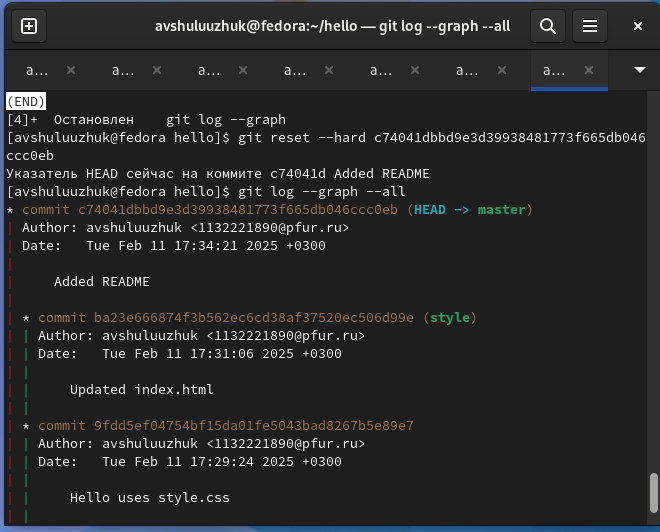


Рис. 50: сброс ветки style и проверка

Просмотрим лог историй. Теперь ветки style и master идентичны (рис. 51).

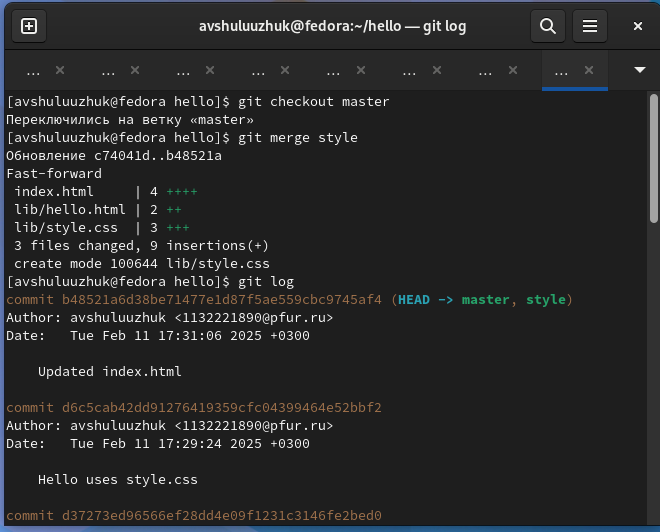


Рис. 51: слияние style в master

Перейдем в рабочий каталог и сделаем клон репозитория hello. Сейчас мы находимся в рабочем каталоге. В этот момент вы должны находиться в «рабочем» каталоге. Здесь должен быть единственный репозиторий под названием «hello». Создадим клон репозитория. В вашем рабочем каталоге теперь должно быть два репозитория: оригинальный репозиторий «hello» и клонированный репозиторий «cloned\_hello» (рис. 52)

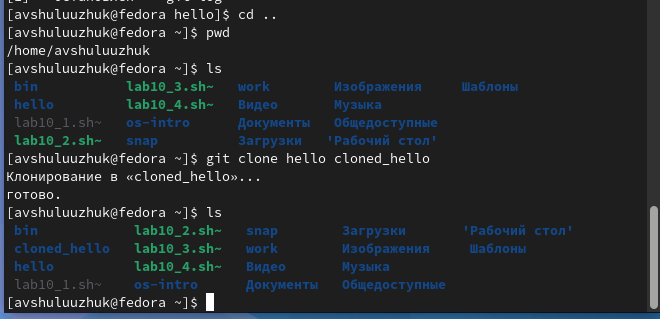


Рис. 52: переход в рабочий каталог и создание клона репозитория

Взглянем на клонированный репозиторий. Увидем список всех файлов на верхнем уровне оригинального репозитория README.md, index.html и lib. Просмотрим историю репозитория (рис. 53).

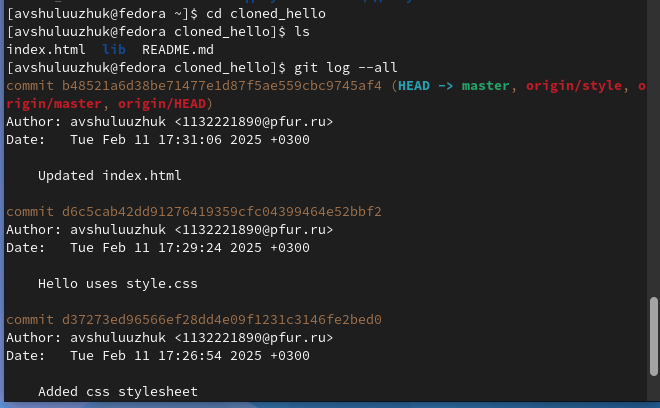


Рис. 53: просмотр клонированного репозитоия

Выполним git remote. Мы видим, что клонированный репозиторий знает об имени по умолчанию удаленного репозитория. Посмотрим, можем ли мы получить более подробную информацию об имени по умолчанию (рис. 54).

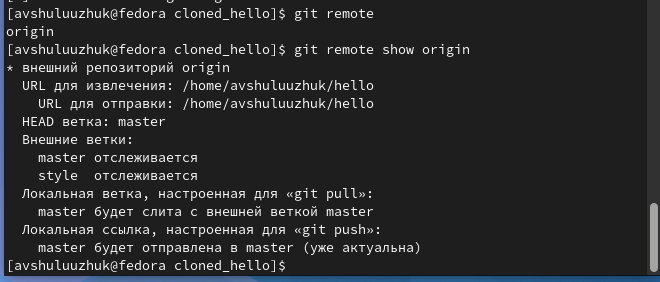


Рис. 54: просмотр состояния репозиториев

Посмотрим на ветки, доступные в нашем клонированном репозитории. Как мы видим, в списке только ветка master. Команда git branch выводит только список локальных веток по умолчанию. Для того, чтобы увидеть все ветки, попробуйте следующую команду: git branch -a. Git выводит все коммиты в оригинальный репозиторий, но ветки в удаленном репозитории не рассматриваются как локальны (рис. 55).

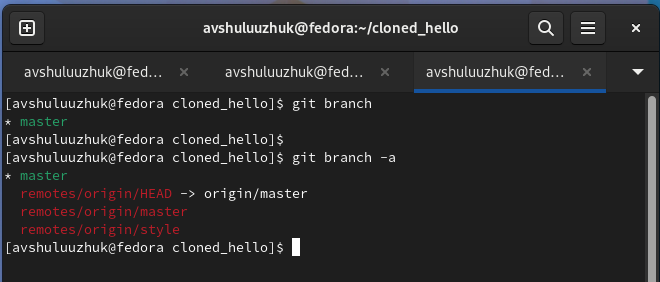


Рис. 55: спиоск удаленных веток

Внесем изменения в оригинальный репозиторий hello. Сейчас мы находимся в репозитории hello. Внесем следующие изменения в файл README.md: This is the Hello World example from the git tutorial. Теперь добавим это изменение и сделаем коммит. Теперь в оригинальном репозитории есть более поздние изменения, которых нет в клонированной версии. Далее мы извлечем и сольем эти изменения в клонированный репозиторий (рис. 56).

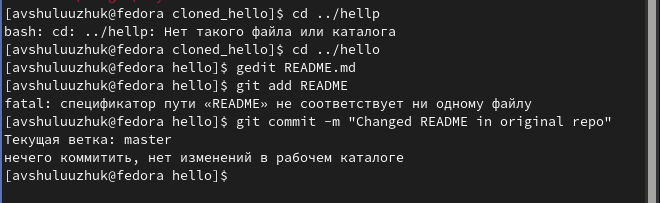


Рис. 56: внесение изменения в оригинальный репозиторий hello

Научимся извлекать изменения из удаленного репозитория. Сейчас мы находимся в репозитории cloned\_hello (рис. 57).

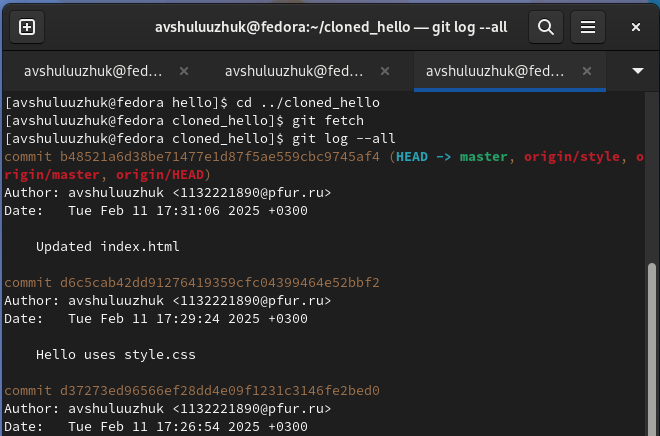


Рис. 57: изввлечение изменений

Сольем извлеченные изменения в локальную ветку master. Еще раз проверим файл README.md. Мы должны увидеть изменения. Команда git fetch не сливает изменения, мы можем вручную слить изменения из удаленного репозитория. Теперь рассмотрим объединение fetch и merge в одну команду (рис. 58).

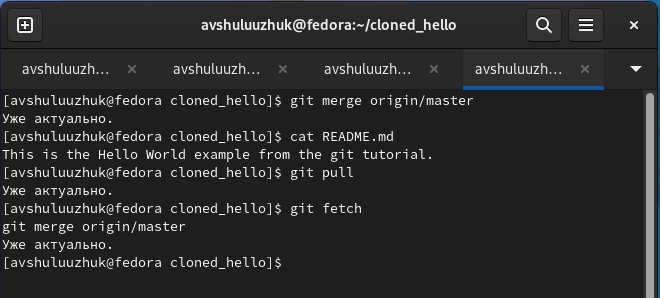


Рис. 58: слияние извлеченных изменений

Добавим локальную ветку, которая отслеживает удаленную ветку. Теперь можем видеть ветку style в списке веток и логе (рис. 59)

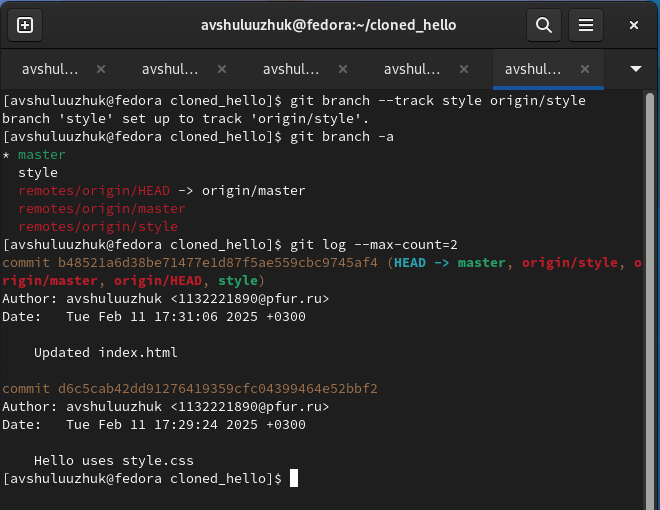


Рис. 59: добавление локальной ветки, которая отслеживает удаленныую ветку

Создадим чистый репозиторий. Сейчас мы находимся в рабочем каталоге (рис. 60).

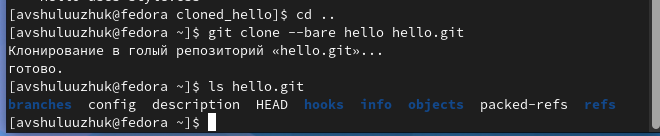


Рис. 60: создание чистого репозитория

Добавим репозиторий hello.git к нашему оригинальному репозиторию. 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. Отредактируем файл README.md и сделаем коммит. Теперь отправим изменения в общий репозиторий. Общим называется репозиторий, получающий отправленные нами изменения (рис. 61).

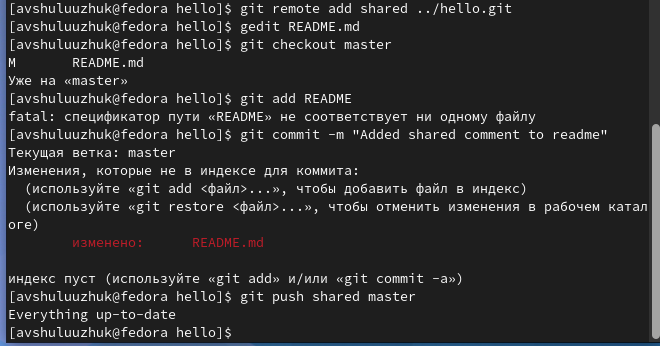


Рис. 61: добавление удаленного репозитория и отправка изменений

Научимся извлекать изменения из общего репозитория. Быстро переключимся в клонированный репозиторий и извлечем изменения, только что отправленные в общий репозиторий. Сейчас мы находимся в репозитории cloned\_hello (рис. 62).

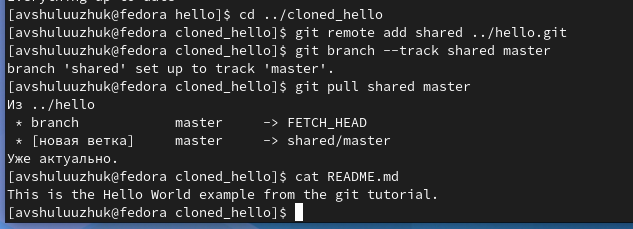


Рис. 62: извлечение общих изменений

# 3 Выводы

В результате выполнения лабораторной работы научились пользоваться с git и разобрались с его особенностями.