Лабораторная работа №1

Работа с git

Галацан Николай, НПИбд-01-22

Содержание

# 1 Цель работы

Научиться использовать систему управления версиями git, освоить основные команды, применить на практике полученные знания.

# 2 Теоретическое введение

Git — распределённая система управления версиями. Проект был создан Линусом Торвальдсом для управления разработкой ядра Linux, первая версия выпущена 7 апреля 2005 года; координатор — Дзюн Хамано. Среди проектов, использующих Git, — ядро Linux, Swift, Android, Drupal, Cairo, GNU Core Utilities, Mesa, Wine, Chromium, Compiz Fusion, FlightGear, jQuery, PHP, NASM, MediaWiki, DokuWiki, Qt, ряд дистрибутивов Linux.

Программа является свободной и выпущена под лицензией GNU GPL версии 2. По умолчанию используется TCP-порт 9418 [1].

# 3 Выполнение лабораторной работы

## 3.1 Подготовка

Сначала проверяю имя пользователя и электронную почту. Приступаю к настройке core.autocrlf, core.safecrlf, core.quotepath (рис. 1).

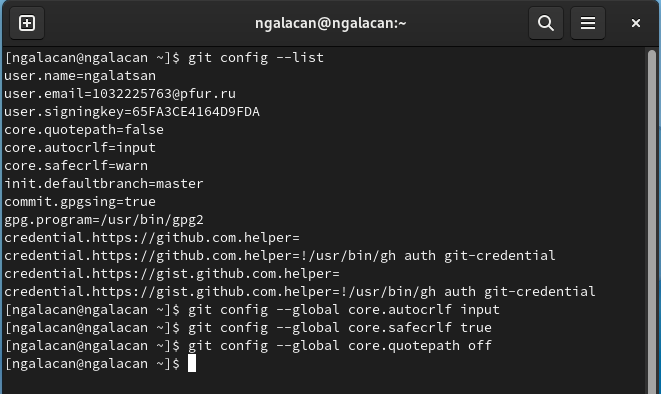


Рис. 1: Настройка основных параметров git

## 3.2 Создание проекта

Создаю рабочий каталог с файлом hello.html. Добавляю запись. Создаю репозиторий и добавляю файл, проверяю текущее состояние. В репозитории хранится текущее состояние рабочего каталога, и нет никаких изменений, ожидающих записи (рис. 2).

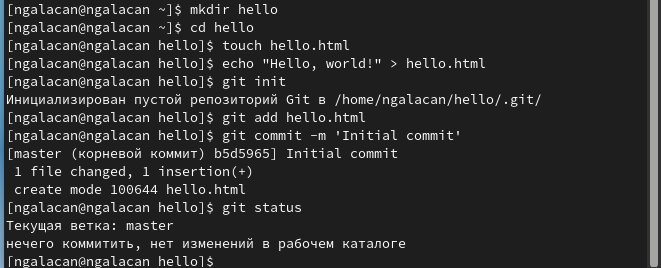


Рис. 2: Создание рабочего пространства и репозитория

## 3.3 Внесение и индексация изменений

Изменяю содержимое файла на <h1>Hello, World!</h1>. Проверяю состояние (git знает, что файл hello.html был изменен, но при этом эти изменения еще не зафиксированы в репозитории), индексирую изменения и вновь проверяю (изменения файла hello.html были проиндексированы. Это означает, что git теперь знает об изменении, но изменение пока не записано в репозиторий) (рис. 3).

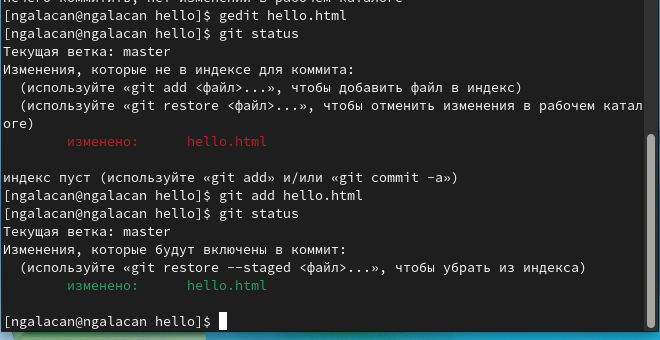


Рис. 3: Внесение изменений и индексация, проверка статуса

Делаю коммит изменений (рис. 4).

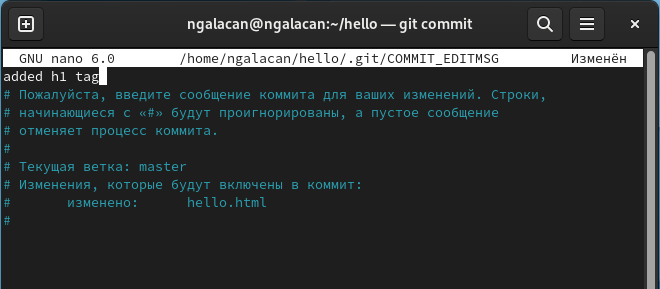


Рис. 4: Коммит первоначальных изменений

Еще раз проверив статус, убеждаюсь, что рабочий каталог чистый. Изменяю страницу «Hello, World!», чтобы она содержала стандартные теги (рис. 5).

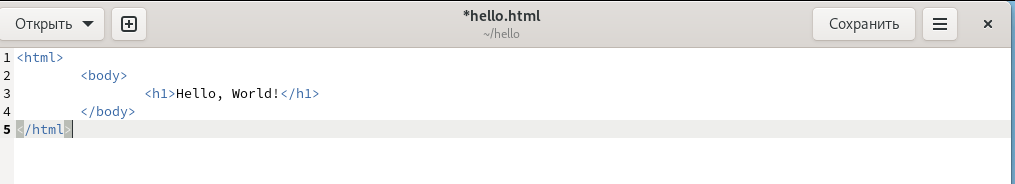


Рис. 5: Добавление стандартных тегов

Теперь добавляю это изменение в индекс git. В hello.html добавляю секцию <head>:

<html>  
 <head>  
 </head>  
 <body>  
 <h1>Hello, World!</h1>  
 </body>  
</html>

Проверяю статус: hello.html указан дважды в состоянии. Делаю коммит индексированного изменения, просматриваю статус: осталось только непроиндексированное изменение. Индексирую и проверяю. Делаю еще один коммит (рис. 6).

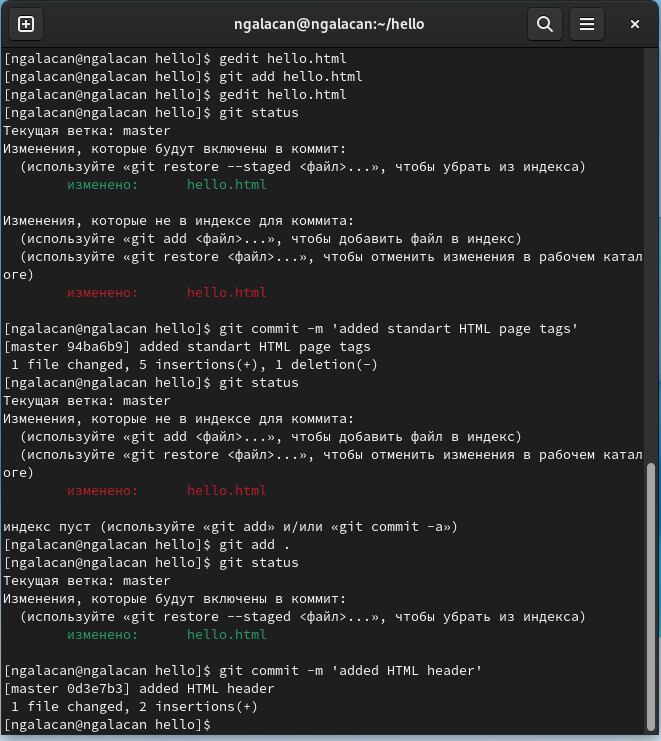


Рис. 6: Внесение изменений, индексированние, коммит

## 3.4 История

Получаю историю изменений. Просматриваю в разных форматах (рис. 7).

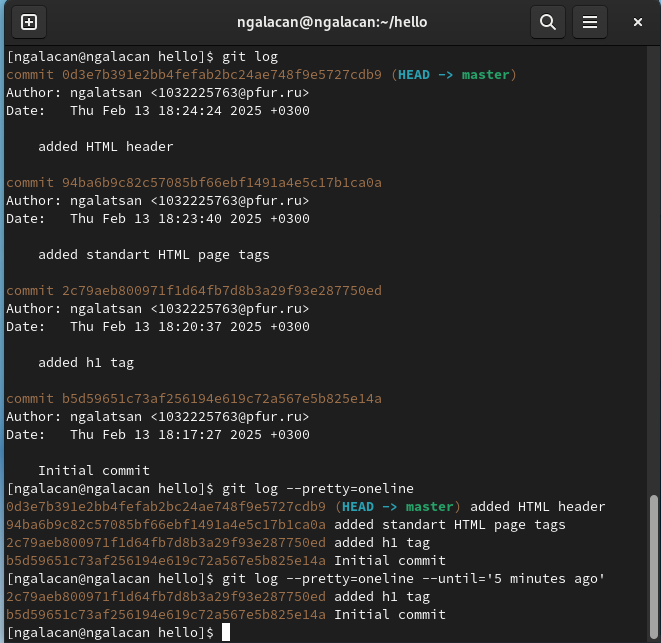


Рис. 7: Просмотр истории

## 3.5 Получение старых версий

Изучаю данные лога и нахожу хэш для первого коммита. Использую этот хэш-код для получения старой версии. Возвращаюсь к последней версии (рис. 8).

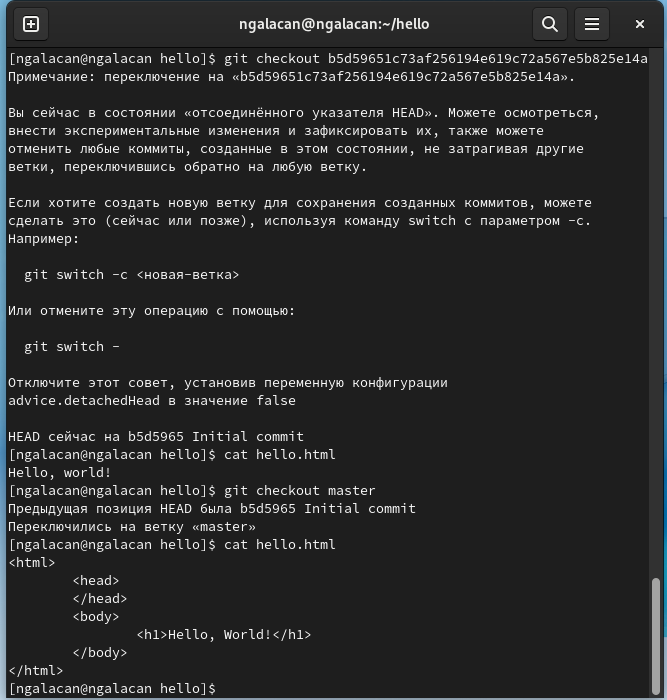


Рис. 8: Получение старых версий

## 3.6 Создание тегов версий

Создаю тег текущей версии (рис. 9), предыдущей версии и переключаюсь между ними, просматриваю теги (рис. 10)

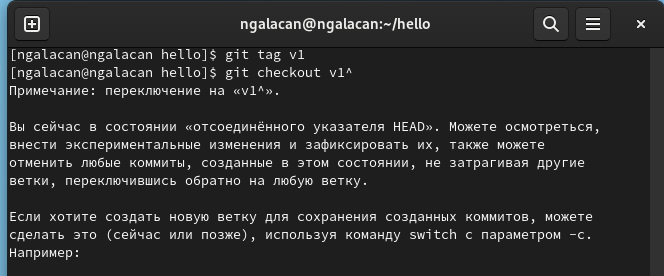


Рис. 9: Создание тега

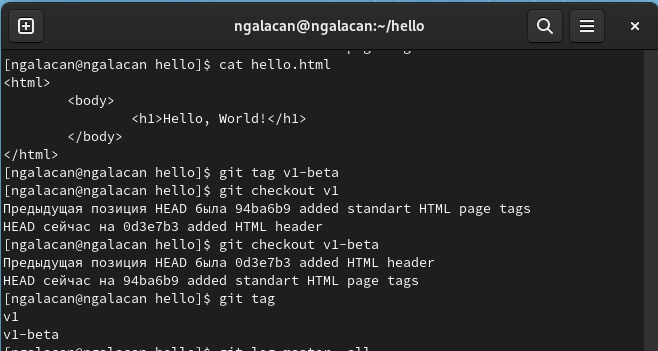


Рис. 10: Переключение по тегам и просмотр

Просматриваю теги в логе. Вижу теги (v1 и v1-beta) в логе вместе с именем ветки (master). Кроме того HEAD показывает коммит, на который я переключился (на данный момент это v1-beta) (рис. 11).

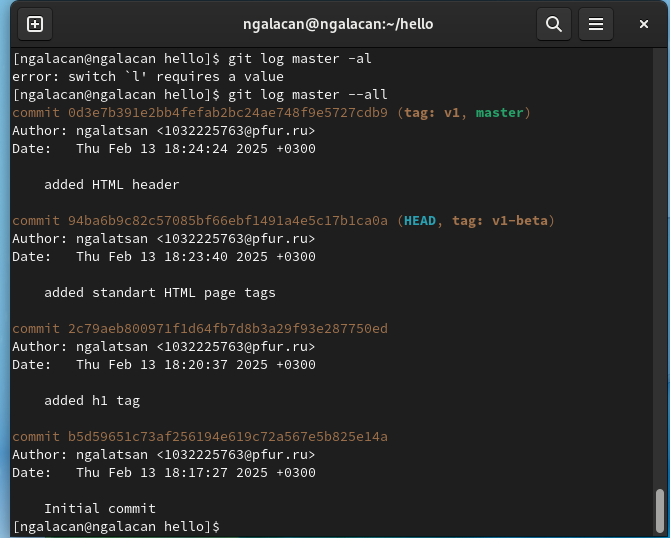


Рис. 11: Просмотр тегов в логе

## 3.7 Отмена локальных изменений (до индексации)

Переключившись на ветку master изменяю файл:

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

Проверяю состояние: файл hello.html был изменен, но еще не проиндексирован. Для переключения версии файла использую команды, проверяю статус, просматриваю файл (рис. 12).

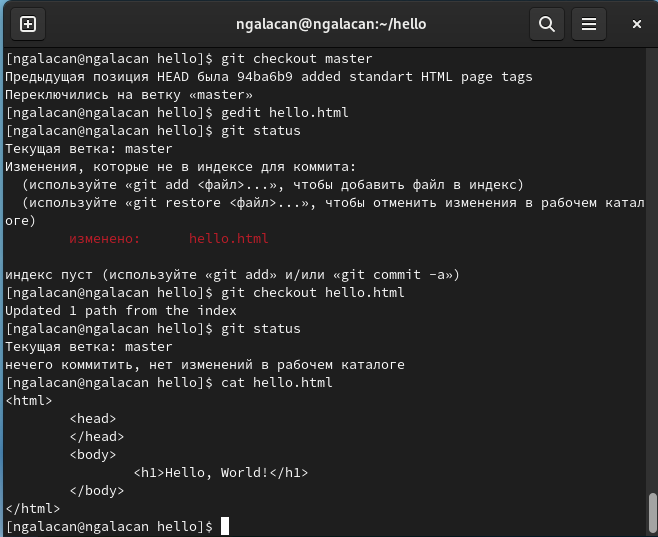


Рис. 12: Отмена изменений до индексации

## 3.8 Отмена проиндексированных изменений (перед коммитом)

Вновь вношу изменений в файл:

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

Индексирую изменения. Выполняю сброс буферной зоны. Команда git reset сбрасывает буферную зону к HEAD. Это очищает буферную зону от изменений, которые мы только что проиндексировали. Использую команду git checkout, чтобы удалить нежелательные изменения в рабочем каталоге (рис. 13).

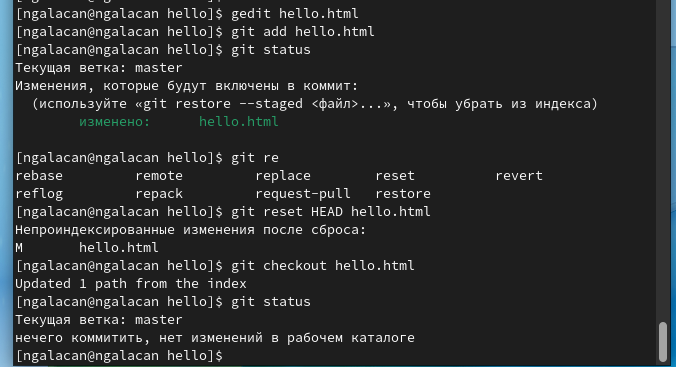


Рис. 13: Отмена изменений после индексации

## 3.9 Отмена коммитов

Изменяю файл:

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

Индексирую и коммичу изменение. Чтобы отменить коммит, необходимо сделать коммит, который удаляет изменения, сохраненные нежелательным коммитом (рис. 14).

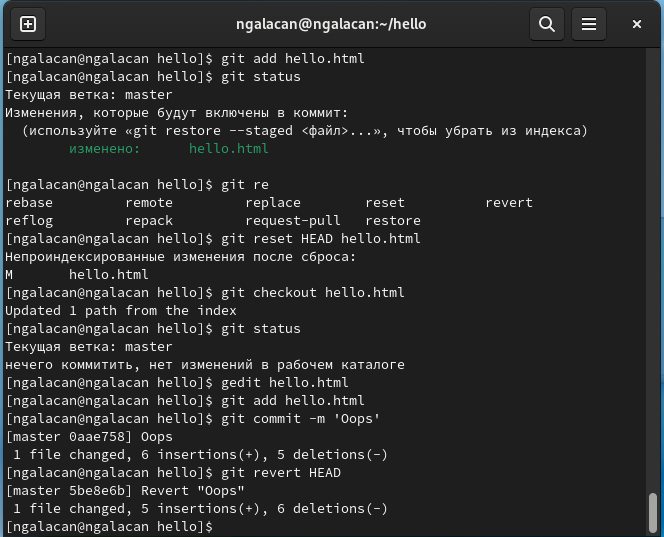


Рис. 14: Отмена изменений после коммита

Проверив лог, вижу оба коммита (рис. 15).

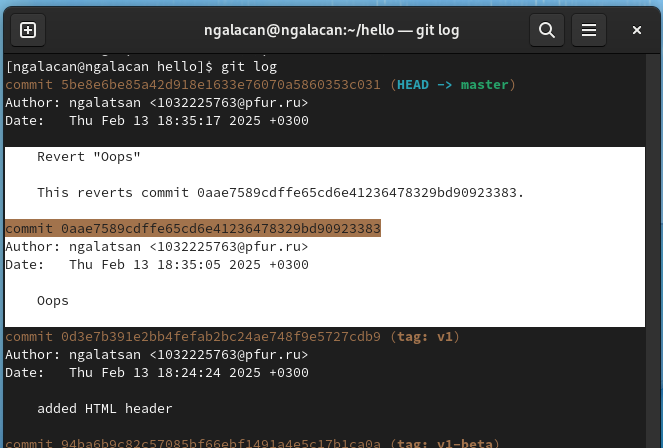


Рис. 15: Нежелательные коммиты

## 3.10 Удаление коммиттов из ветки

Удаляю последние два коммита с помощью сброса, сначала отметив последний коммит тегом oops, чтобы его можно было потом найти. Использую команду git reset, чтобы вернуться к версии до этих коммитов. Теперь в логе их нет, но если посмотретьлоги с опцией --all можно всё ещё их увидеть, но метка HEAD находится на нужной версии (рис. 16).

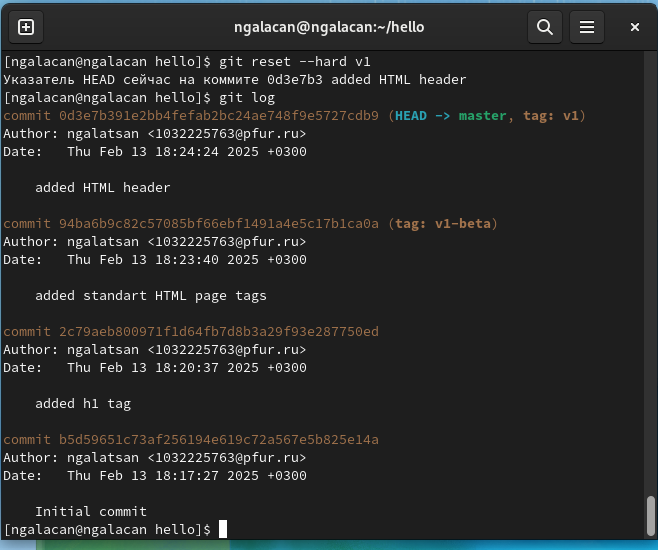


Рис. 16: Сброс коммитов

## 3.11 Удаление тега oops

Удаляю тег и коммиты, на которые он ссылался, сборщиком мусора (рис. 17).

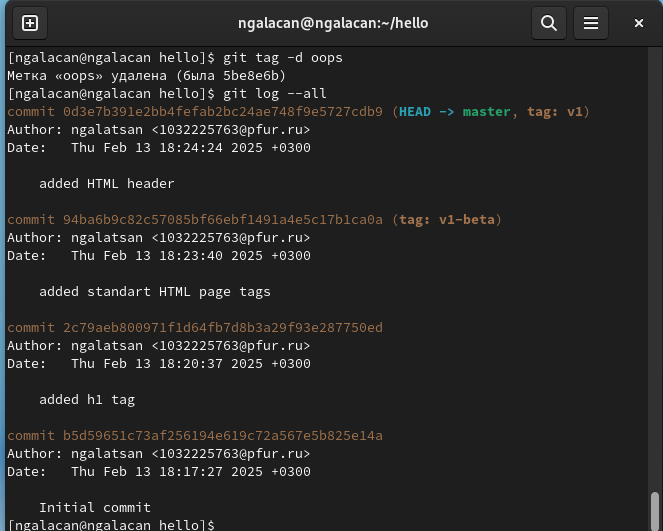


Рис. 17: Удаление тега и коммитов

## 3.12 Внесение изменений в коммиты

Изменяю файл:

<!-- Author: N. Galatsan -->  
<html>  
 <head>  
 </head>  
 <body>  
 <h1>Hello, World!</h1>  
 </body>  
</html>

Индексирую изменения и делаю коммит. Добавляю в файл информацию об электронной почте:

<!-- Author: N. Galatsan (1032225763@pfur.ru) -->  
<html>  
 <head>  
 </head>  
 <body>  
 <h1>Hello, World!</h1>  
 </body>  
</html>

Изменяю предыдущий коммит. Можно 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 (рис. 18).

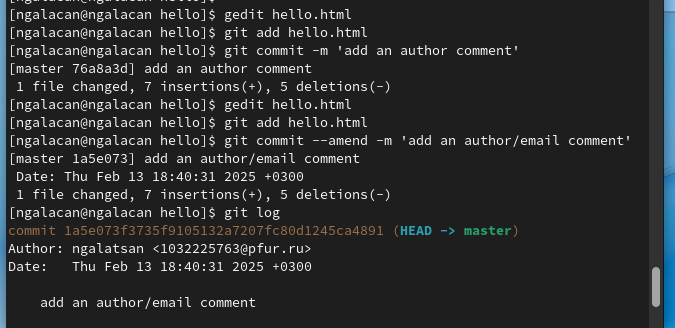


Рис. 18: Изменение коммита

## 3.13 Перемещение файлов

Создаю каталог lib и перемещаю туда файл hello.html с помощью git mv. Делаю коммит, добавляю файл index.html (рис. 20) в репозиторий, делаю коммит (рис. 19).

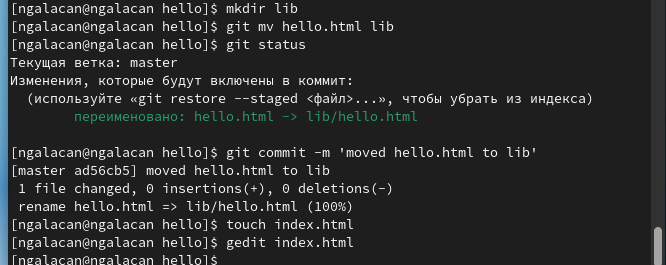


Рис. 19: Перемещение файла



Рис. 20: index.html

Теперь при открытии index.html, можно увидеть кусок страницы hello в маленьком окошке (рис. 21).

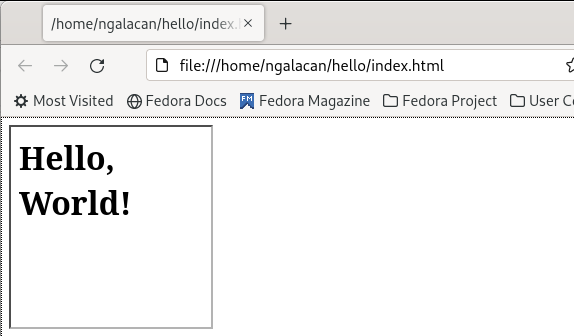


Рис. 21: index.html в браузере

## 3.14 Git внутри: Каталог .git

Просматриваю данные из каталога .git, объектные файлы, ветки, теги и хэш (рис. 22).

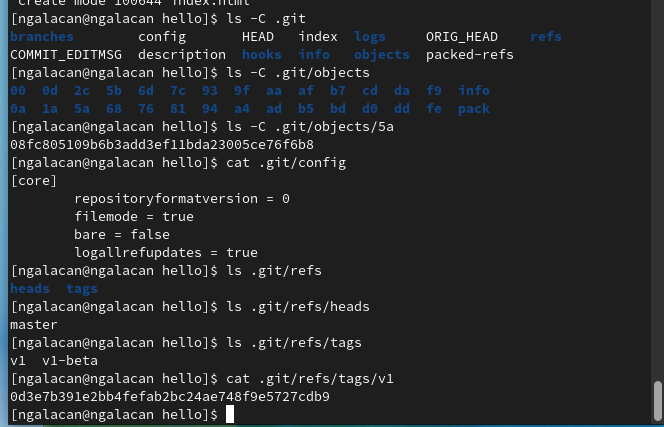


Рис. 22: Просмотр .git

Просматриваю файл HEAD, который содержит ссылку на текущую ветку. Ищу последний коммит. Вывожу последний коммит с помощью SHA1 хэша, просматриваю дерево, каталог lib, файл hello.html (рис. 23).

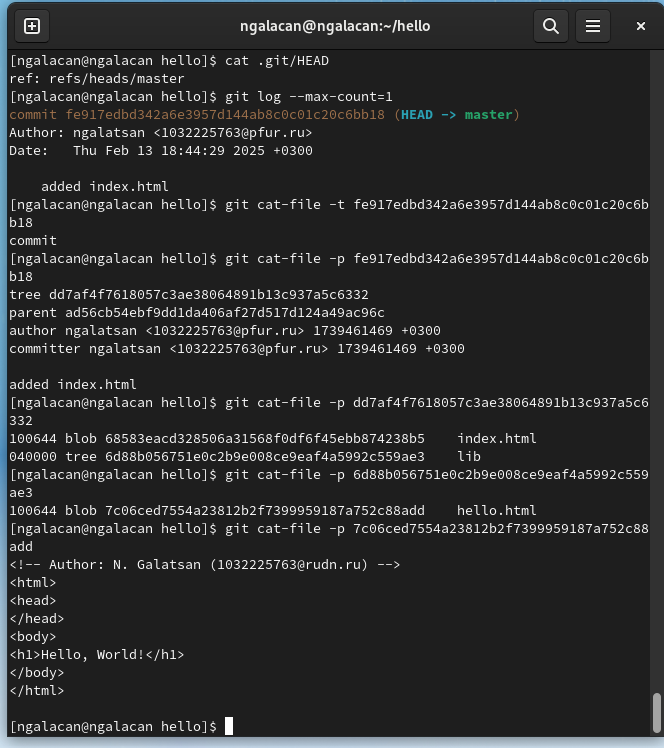


Рис. 23: Просмотр по хэшу

Вручную дохожу до оригинального файла hello.html с первого коммита по родительским хэшам и хэшу дерева (рис. 24).

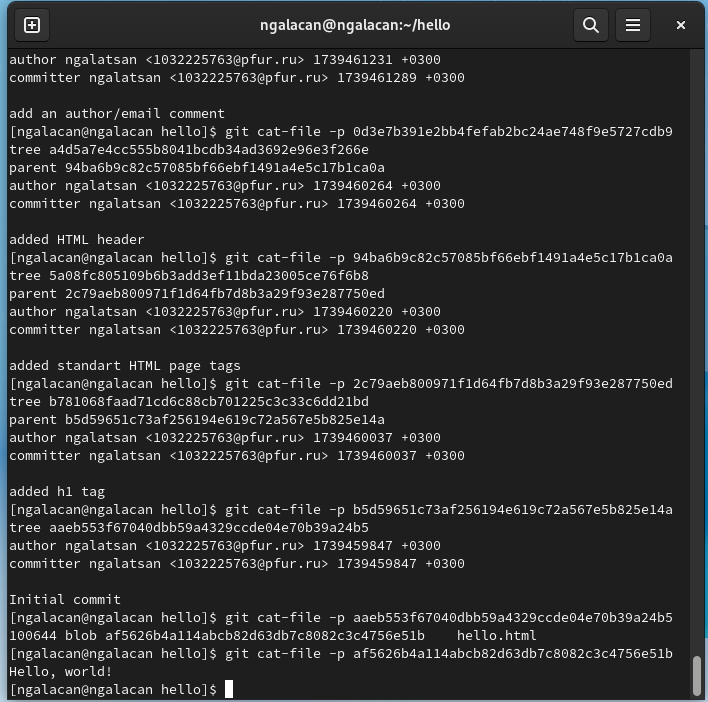


Рис. 24: Поиск первоначального файла hello.html по хэшу

## 3.15 Создание ветки

Создаю ветку style и добавляю в lib файл стилей style.css:

h1 {  
 color: red;  
}

Индексирую и делаю коммит. Изменяю основную страницу, index.html, добавив поддержку стилей (рис. 25).

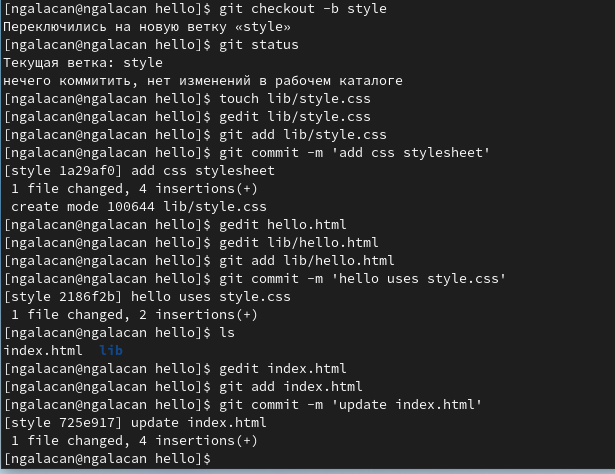


Рис. 25: Создание новой ветки и изменение файлов

В проекте теперье есть две ветки. Просматриваю лог, переключаюсь между ветками (рис. 26).

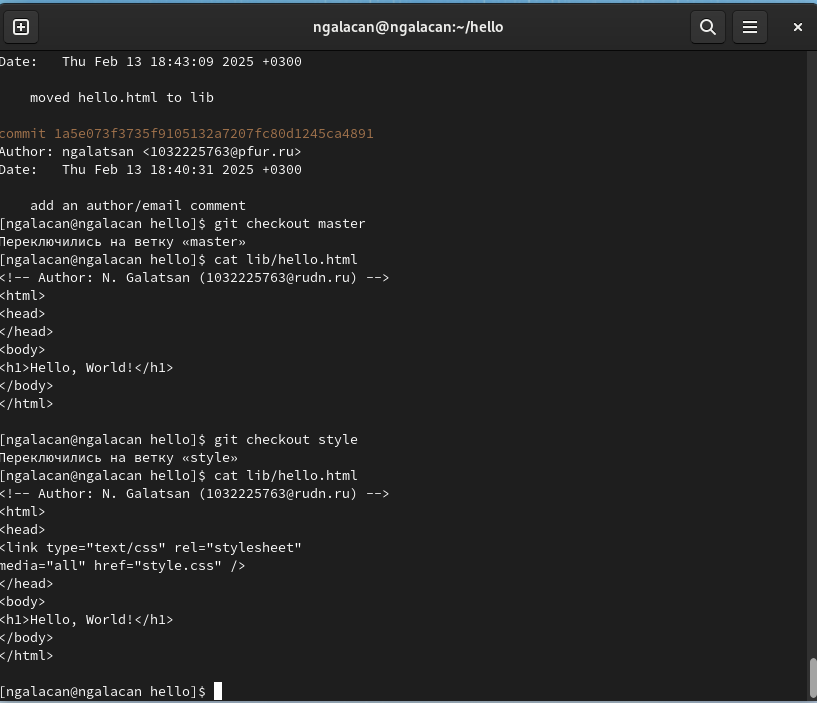


Рис. 26: Просмотр файлов на двух ветках

## 3.16 Изменения в ветке master

Создаю README.md в ветке master, делаю коммит изменений (рис. 27).

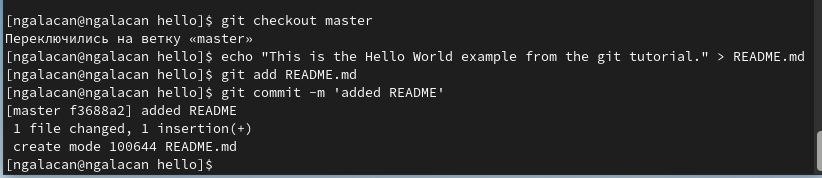


Рис. 27: Создание README.md в master

Просматриваю ветки и их отличия (рис. 28).

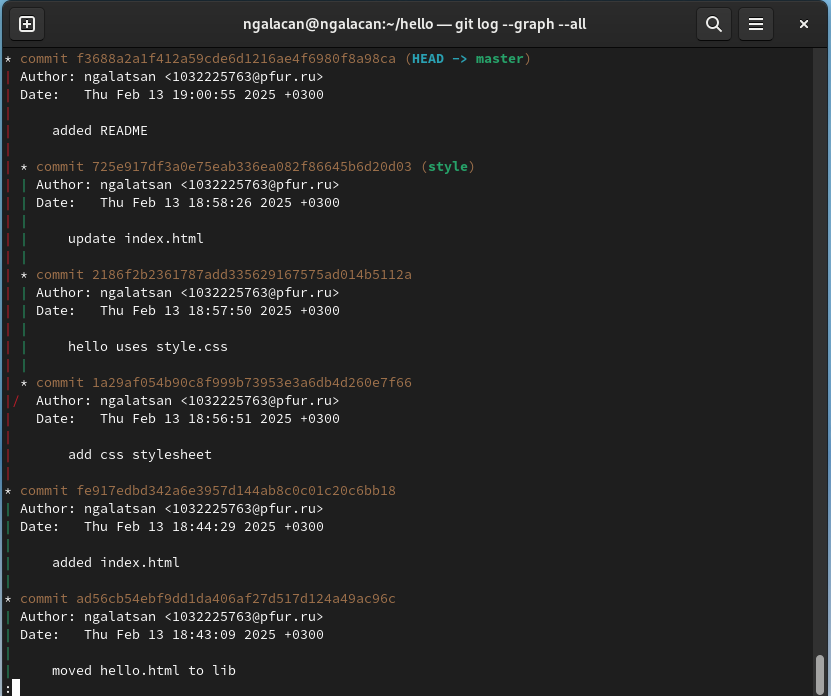


Рис. 28: Просмотр веток

## 3.17 Слияние

Слияние переносит изменения из двух веток в одну. Возвращаюсь к ветке style и сливаю master с style и просматриваю ветки (рис. 29).

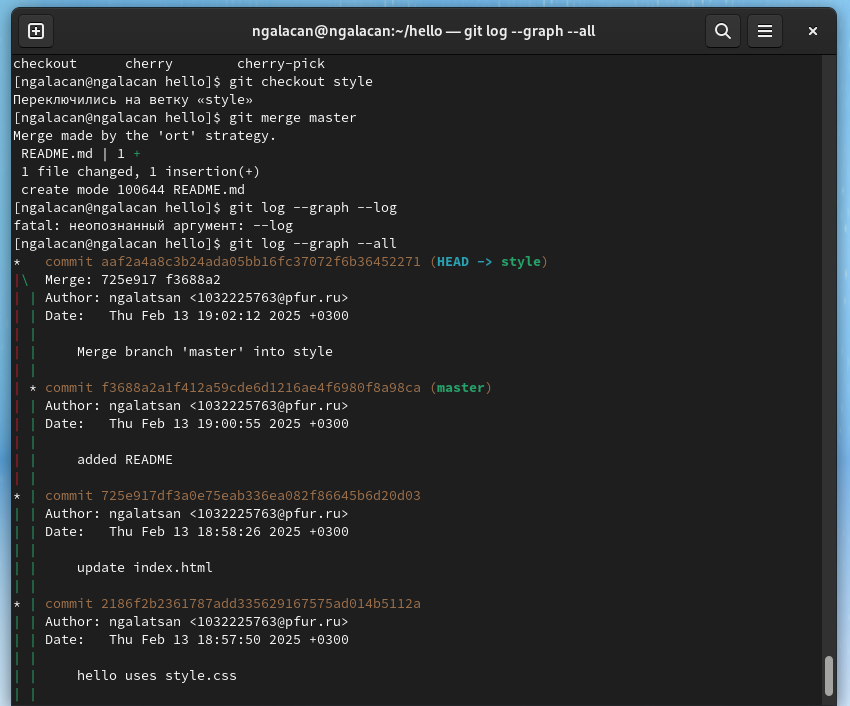


Рис. 29: Слияние master с style

## 3.18 Создание конфликта

В ветке master изменяю hello.html:

Hello, World! Life is great!

Делаю коммит, просматриваю ветки, вижу конфликт (рис. 30).

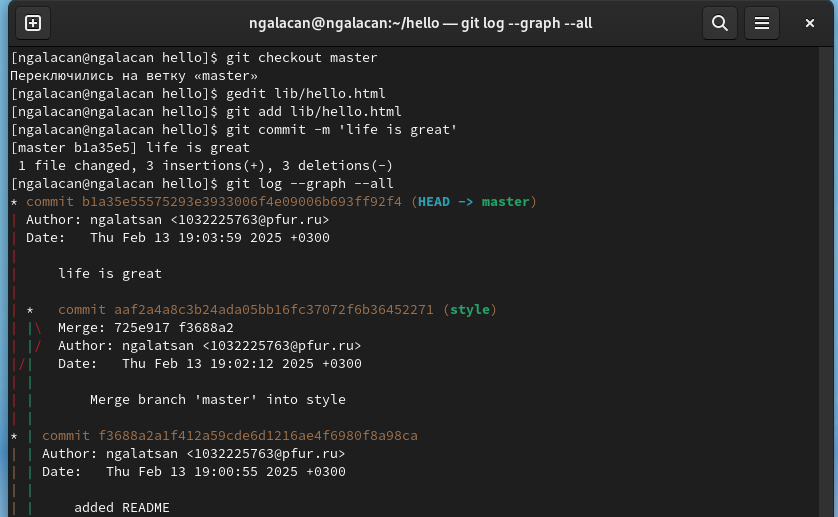


Рис. 30: Создание конфликта

Возвращаюсь к ветке style и пытаюсь объединить ее с новой веткой master. В файле lib/hello.html можно увидеть записи с обеих версий этого файла. Первый раздел — версия текущей ветки (style). Второй раздел — версия ветки master. Вношу изменения в lib/hello.html, оставив только необходимую запись и добавляю этот файл в репозиторий, чтобы вручную разрешить конфликт (рис. 31).

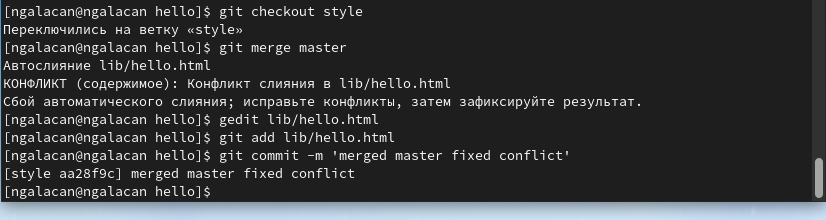


Рис. 31: Разрешение конфликта и слияние

## 3.19 Сброс ветки style

Вернемся на ветке style к точке перед тем, как мы слили ее с веткой master. Можно 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. Необходимо найти последний коммит перед слиянием. Сделаю сброс по хэшу и проверю (рис. 32).

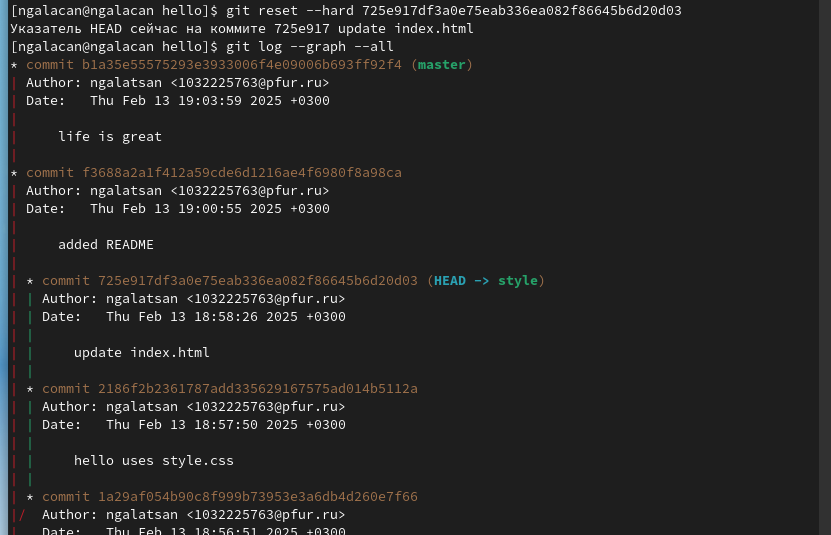


Рис. 32: Сброс ветки style по хэшу

## 3.20 Сброс ветки master

Аналогичным образом сбрасываю и эту ветку (рис. 33).

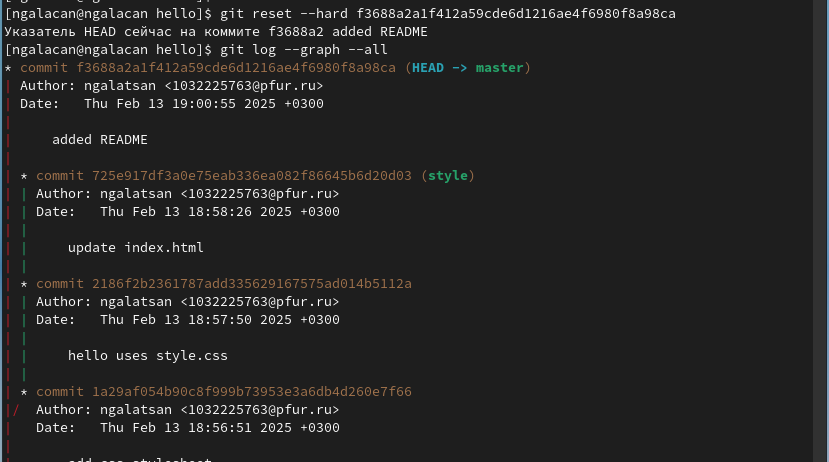


Рис. 33: Сброс ветки master по хэшу

## 3.21 Перебазирование

Используем команду rebase вместо команды merge. Мы вернулись в точку до первого слияния и хотим перенести изменения из ветки master в нашу ветку style (рис. 34).

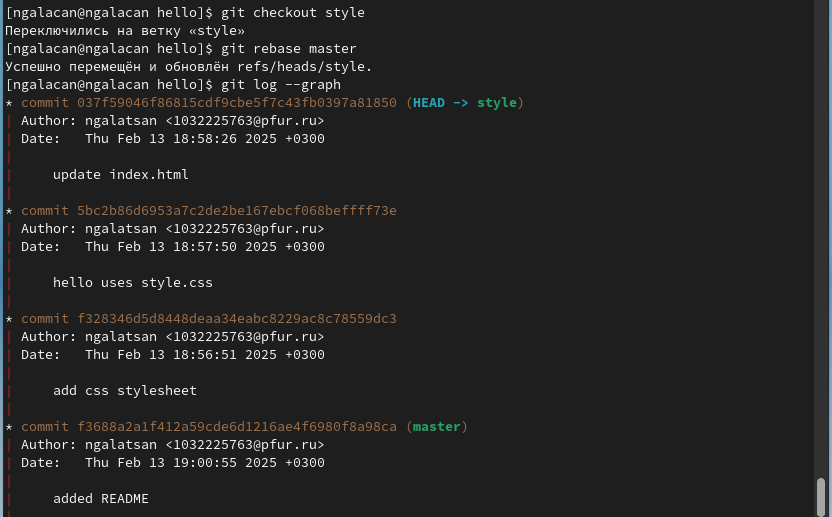


Рис. 34: Перебазирование

## 3.22 Слияние в ветку master

Выполняю слияние style в master. Просматриваю лог. Ветки идентичны (рис. 35).

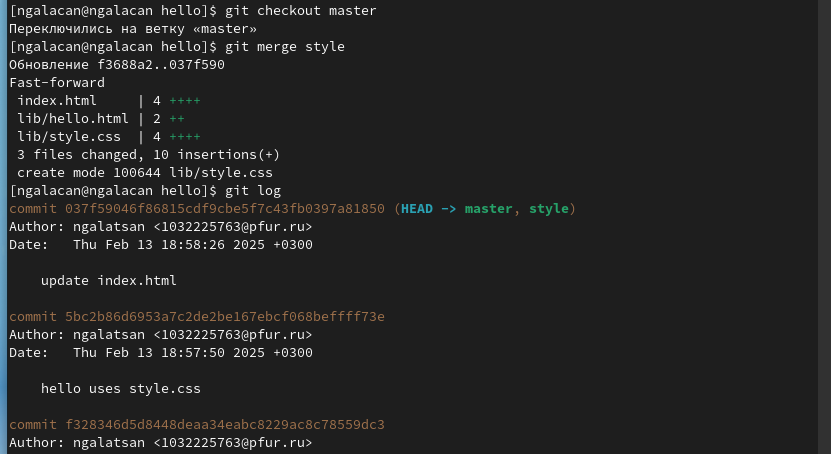


Рис. 35: Слияние style в master

## 3.23 Клонирование репозиториев

Создаю клон первоначального репозитория. Просматриваю его. Просматриваю лог клонированного репозитория. Отображаются все файлы и коммиты оригинального репозитория (рис. 36).

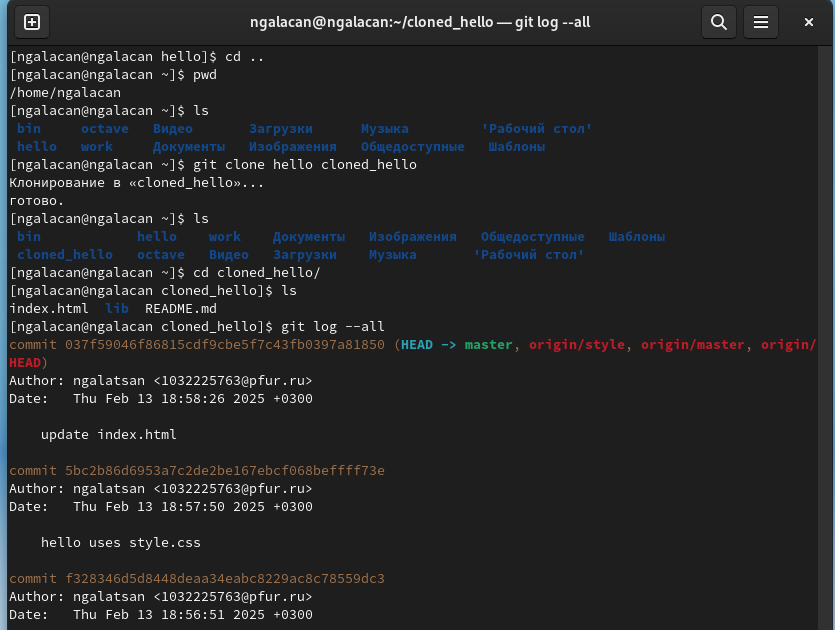


Рис. 36: Создание клона репозитория, просмотр файлов и коммитов

## 3.24 Удаленные ветки

Просматриваю подробную информацию о ветках. Просматриваю доступные ветки, затем и удаленные. Git выводит все коммиты в оригинальный репозиторий, но ветки в удаленном репозитории не рассматриваются как локальные. Если мы хотим собственную ветку style, мы должны сами ее создать (рис. 37).

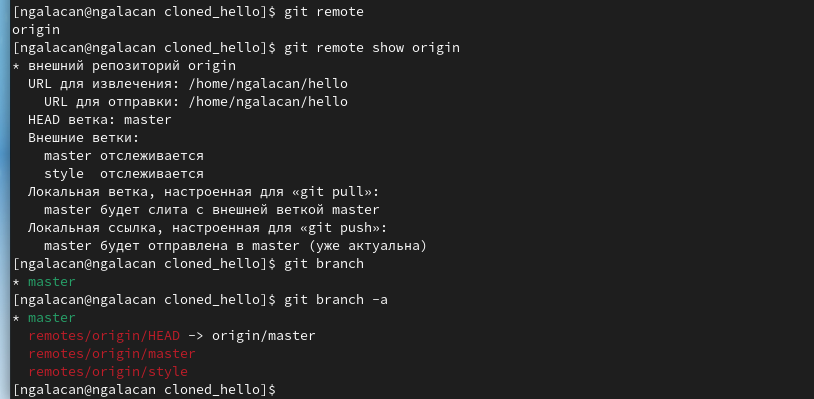


Рис. 37: Просмотр веток в клоне

## 3.25 Изменение оригинального репозитория

Перейдя в оригинальный репозиторий, изменяю README.md:

This is the Hello World example from the git tutorial.

Добавляю изменение и делаю коммит.

## 3.26 Извлечение изменений

Перехожу в клон и извлекаю изменения из оригинального репозитория. На данный момент в репозитории есть все коммиты из оригинального репозитория, но они не интегрированы в локальные ветки клонированного репозитория (рис. 38).

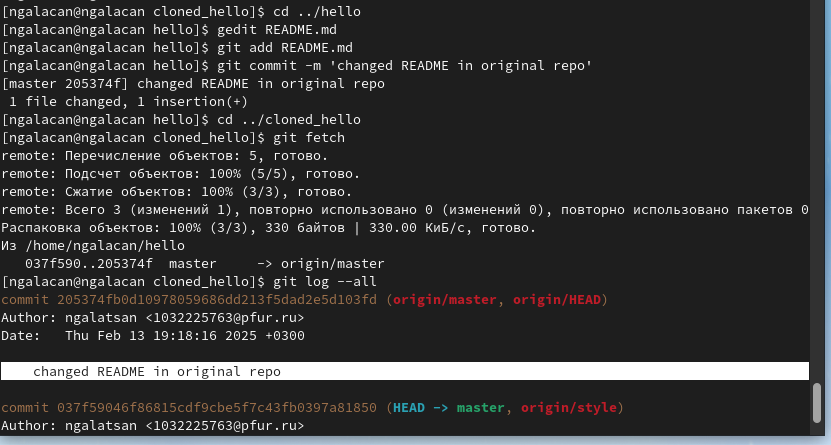


Рис. 38: Извлечение изменений из оригинального репозитория

## 3.27 Слияние извлеченных изменений

Просматриваю README.md. Произвожу слияние и проверяю, что файл на клоне обновился. Вместо извлечения и слияния можно было использовать одну команду: git pull(рис. 39).

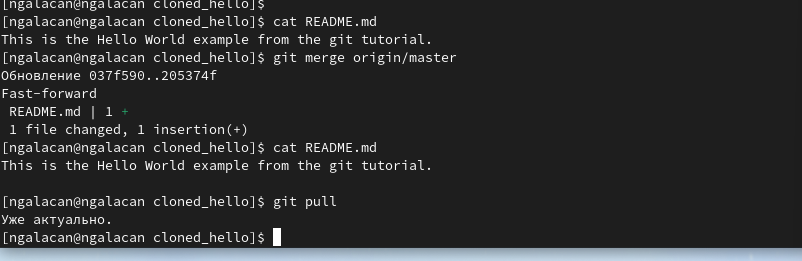


Рис. 39: Слияние извлеченных изменений

## 3.28 Добавление ветки наблюдения

Добавляю локальную ветку, которая отслеживает удаленную ветку (рис. 40).

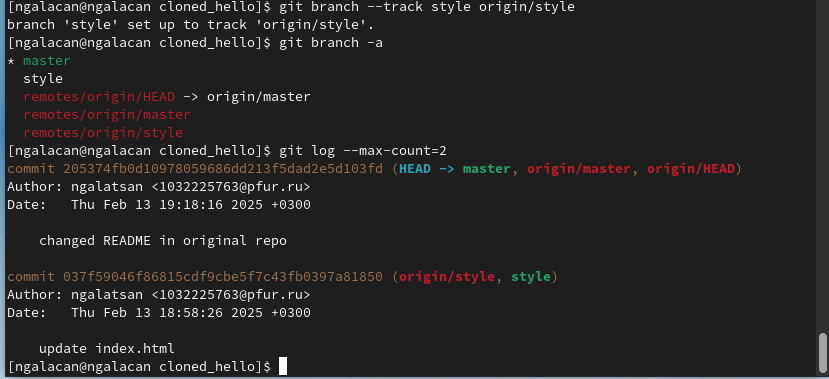


Рис. 40: Добавление ветки наблюдения

## 3.29 Чистые репозитории

Создаю чистый репозиторий. Добавляю его к оригинальному репозиторию. В файл README.md вношу изменения:

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

Добавляю изменения в master и делаю коммит. Отправляю изменения в общий репозиторий (рис. 41).

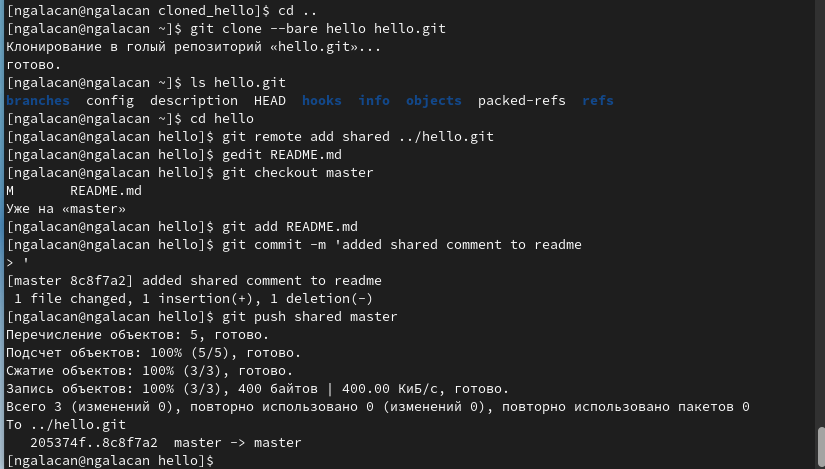


Рис. 41: Создание чистого репозитория, добавление в оригинальный, отправка изменений в общий

## 3.30 Извлечение общих изменений

Переключаюсь на клон и добавляю чистый репозиторий. С помощью git pull shared master извлекаю изменения. Просматриваю README.md (рис. 42).

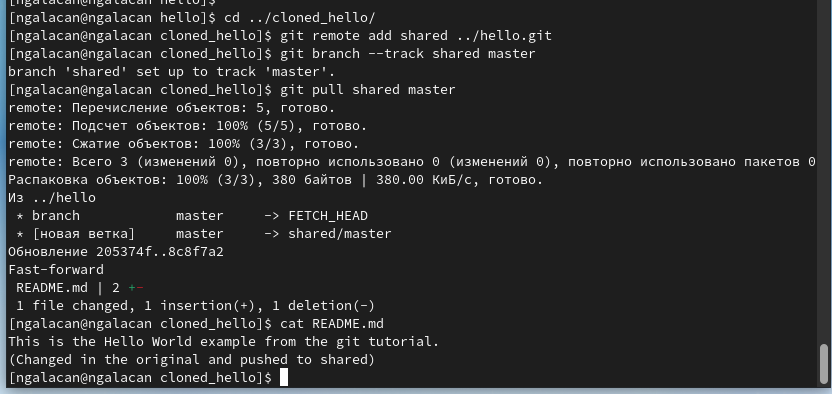


Рис. 42: Извлечение общих изменений, проверка

# 4 Выводы

В результате выполнения работы были получены практические навыки использования системы управления версиями git.

# Список литературы

1. Git [Электронный ресурс]. Wikimedia Foundation, Inc., 2025. URL: <https://ru.wikipedia.org/wiki/Git>.