

Отчёт по лабораторной работе №1

Дисциплина: Математическое моделирование

Зиязетдинов Алмаз Радикович

Содержание

| | |
|---|----------|
| 1 Цель работы | 6 |
| 2 Теоретическое введение | 7 |
| 3 Выполнение лабораторной работы | 8 |
| 3.1 Подготовка | 8 |
| 3.2 Создание проекта | 8 |
| 3.3 Внесение изменений X | 9 |
| 3.4 Индексация изменений X | 10 |
| 3.5 История | 13 |
| 3.6 Создание тегов версии X | 14 |
| 3.7 Отмена локальных изменений X (до индексации) | 16 |
| 3.8 Отмена проиндексированных изменений X (перед коммитом) | 17 |
| 3.9 Отмена коммитов | 18 |
| 3.10 Удаление коммитов из ветки | 19 |
| 3.11 Удаление тега oops | 20 |
| 3.12 Внесение изменений X в коммиты | 20 |
| 3.13 Перемещение файлов | 20 |
| 3.14 Git внутри: Каталог .git | 21 |
| 3.15 Работа непосредственно с объектами git | 22 |
| 3.16 Создание ветки | 23 |
| 3.17 Навигация по веткам | 24 |
| 3.18 Изменения в ветке main | 25 |
| 3.19 Слияние | 26 |
| 3.20 Создание конфликта | 27 |
| 3.21 Разрешение конфликтов | 27 |
| 3.22 Сброс ветки style | 27 |
| 3.23 Сброс ветки main | 28 |
| 3.24 Слияние в ветку main | 28 |
| 3.25 Клонирование репозиториев | 29 |
| 3.26 Что такое origin? | 29 |
| 3.27 Изменение оригинального репозитория | 30 |
| 3.28 Слияние извлеченных изменений X | 30 |
| 3.29 Добавление ветки наблюдения | 30 |

| | |
|--|-----------|
| 3.30 Создаи те чистый и репозитории и | 31 |
| 3.31 Добавление удаленного репозитория | 31 |
| 3.32 Извлечение общих изменений и | 32 |
| 4 Выводы | 33 |
| Список литературы | 34 |

Список иллюстраций

| | | |
|------|--|----|
| 3.1 | Настройка git | 8 |
| 3.2 | Создание файла и репозитория. Добавление файла в репозиторий | 9 |
| 3.3 | Файл hello.html | 10 |
| 3.4 | Индексация и коммит изменений | 11 |
| 3.5 | Редактор | 11 |
| 3.6 | Файл hello.html | 12 |
| 3.7 | Индексация и коммит изменений | 13 |
| 3.8 | Файл hello.html | 13 |
| 3.9 | История | 14 |
| 3.10 | Просмотр разных версий репозитория | 14 |
| 3.11 | Создание тегов версий, переключение между ними | 15 |
| 3.12 | Доступные теги | 16 |
| 3.13 | Отмена локальных изменений (до индексации) | 17 |
| 3.14 | Изменения в файле hello.html | 17 |
| 3.15 | Изменения в файле hello.html | 18 |
| 3.16 | Отмена локальных изменений (перед коммитом) | 18 |
| 3.17 | Git внутри: Каталог .git | 22 |
| 3.18 | Работа с объектами git | 22 |
| 3.19 | Создание ветки style и файла style.css | 23 |
| 3.20 | Файл hello.html | 23 |
| 3.21 | Файлы hello.html и index.html | 24 |
| 3.22 | Логи | 24 |
| 3.23 | Навигация по веткам | 25 |
| 3.24 | Изменения в ветке main | 26 |
| 3.25 | Слияние main c style | 26 |
| 3.26 | Ветка main | 28 |
| 3.27 | Информация о репозитории и ветках | 29 |
| 3.28 | Добавление ветки наблюдения | 31 |
| 3.29 | Извлечение общих изменений☒ | 32 |

Список таблиц

1 Цель работы

Приобрести практические навыки работы с системой управления версиями Git.

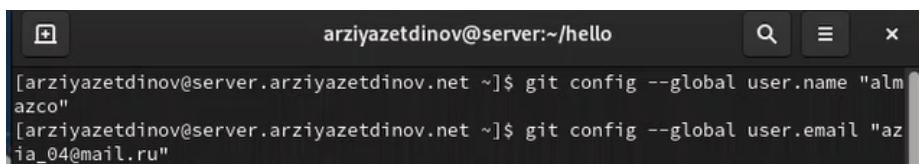
2 Теоретическое введение

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 [[wiki?](#)].

3 Выполнение лабораторной работы

3.1 Подготовка

Ранее я уже использовала git, поэтому мне не пришлось выполнять установку. Для начала настроим core.autocrlf с параметрами true и input, чтобы сделать все переводы строк текстовых файлов в главном репозитории одинаковыми, а затем настроим отображение unicode (рис. fig. 3.1).



```
arziyazetdinov@server:~/hello
[arziyazetdinov@server.arziyazetdinov.net ~]$ git config --global user.name "alma"
[arziyazetdinov@server.arziyazetdinov.net ~]$ git config --global user.email "alma_04@mail.ru"
```

Рис. 3.1: Настройка git

3.2 Создание проекта

Лабораторную работу выполняю в каталоге со своим логином aamishina.

Начнем работу в пустом рабочем каталоге с создания пустого каталога с именем hello, затем воедем в него и создадим там файл с именем hello.html. Запишем в него фразу “Hello world”. Создадим git репозиторий из этого каталога, выполнив команду git init. Добавим файл в репозиторий: git add hello.html и git commit -m “Initial Commit”. Используем команду git status, чтобы проверить текущее состояние репозитория. Команда проверки состояния сообщает, что коммитить нечего. Это означает,

что в репозитории хранится текущее состояние рабочего каталога, и нет никаких изменений~~X~~, ожидающих записи (рис. fig. 3.2).

```
[arziyazetdinov@server.arziyazetdinov.net ~]$ mkdir hello
[arziyazetdinov@server.arziyazetdinov.net ~]$ cd hello
[arziyazetdinov@server.arziyazetdinov.net hello]$ touch hello.html
[arziyazetdinov@server.arziyazetdinov.net hello]$ echo "Hello, World!" > hello.html
[arziyazetdinov@server.arziyazetdinov.net hello]$ git init
hint: Using 'master' as the name for the initial branch. This default branch name
      is subject to change. To configure the initial branch name to use in all
      of your new repositories, which will suppress this warning, call:
      hint:
      hint: git config --global init.defaultBranch <name>
      hint:
      hint: Names commonly chosen instead of 'master' are 'main', 'trunk' and
      hint: 'development'. The just-created branch can be renamed via this command:
      hint:
      hint: git branch -m <name>
Initialized empty Git repository in /home/arziyazetdinov/hello/.git/
[arziyazetdinov@server.arziyazetdinov.net hello]$ █
```

Рис. 3.2: Создание файла и репозитория. Добавление файла в репозиторий

3.3 Внесение изменений~~X~~

Добавим кое-какие HTML-теги к нашему приветствию. Изменим содержимое файла hello.html на:

Hello, World!

(рис. fig. 3.3).

```
arziyazetdinov@server:~/hello — nano hello.html
GNU nano 5.6.1          hello.html          Modified
<h1>Hello, World!</h1>

Save modified buffer?
Y Yes
N No      ^C Cancel
```

Рис. 3.3: Файл hello.html

Проверим состояние рабочего каталога: `git status`. `git` знает, что файл `hello.html` был изменен, но при этом эти изменения еще не зафиксированы в репозитории. Также обратим внимание на то, что сообщение о состоянии дает подсказку о том, что нужно делать дальше.

3.4 Индексация изменений

Теперь выполним команду `git`, чтобы проиндексировать изменения и проверим состояние: `git add hello.html` и `git status` (рис. fig. 3.4). Изменения файла `hello.html` были проиндексированы. Это означает, что `git` теперь знает об изменении, но изменение пока не записано в репозитории. Следующий коммит будет включать в себя проиндексированные изменения. Сделаем коммит: `git commit`. Откроется редактор: в первой строке введем комментарий: «Added h1 tag» (рис. fig. 3.5). Теперь еще раз проверим состояние: `git status`. Рабочий каталог чистый, можно продолжить работу.

```
[arziyazetdinov@server.arziyazetdinov.net hello]$ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
    modified:   hello.html

no changes added to commit (use "git add" and/or "git commit -a")
[arziyazetdinov@server.arziyazetdinov.net hello]$ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
    modified:   hello.html

no changes added to commit (use "git add" and/or "git commit -a")
[arziyazetdinov@server.arziyazetdinov.net hello]$ git add hello.html
[arziyazetdinov@server.arziyazetdinov.net hello]$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    modified:   hello.html

[arziyazetdinov@server.arziyazetdinov.net hello]$
```

Рис. 3.4: Индексация и коммит изменений

```
# Please enter the commit message for your changes. Lines starting
# with '#' will be ignored, and an empty message aborts the commit.
#
# On branch master
# Changes to be committed:
#   modified:   hello.html
```

Рис. 3.5: Редактор

Добавим стандартные теги страницы (рис. fig. 3.6).

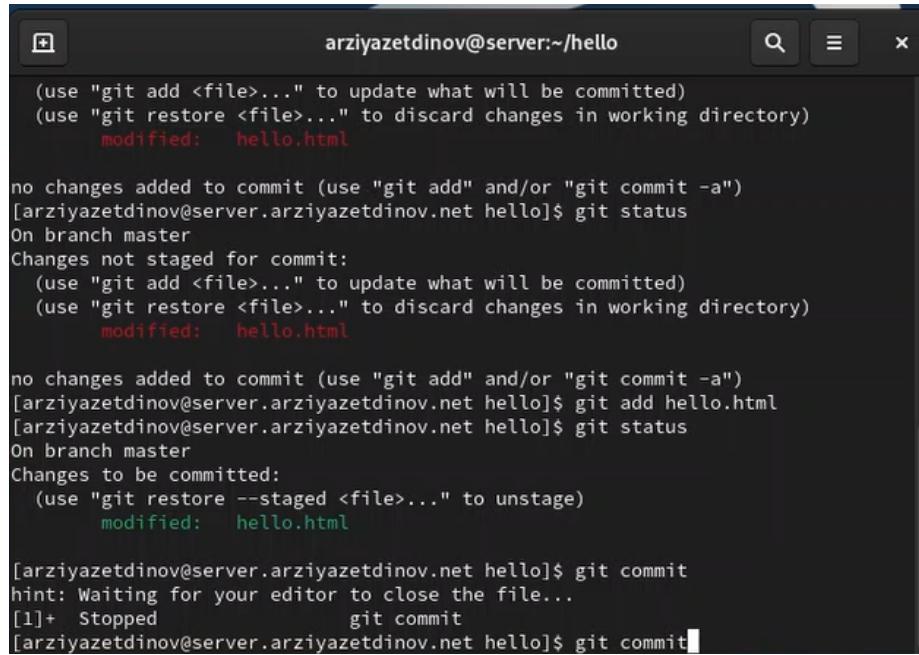


```
GNU nano 5.6.1
<h1>Hello, World!</h1>
```

Рис. 3.6: Файл hello.html

Теперь добавим это изменение в индекс git: git add hello.html (рис. fig. 3.7). Добавим заголовки HTML (секцию

) к странице «Hello, World» (рис. fig. 3.8). Проверим текущий статус: git status. Обратим внимание на то, что hello.html указан дважды в состоянии. Первое изменение (добавление стандартных тегов) проиндексировано и готово к коммиту. Второе изменение (добавление заголовков HTML) является непроиндексированным. Если бы вы делали коммит сейчас, заголовки не были бы сохранены в репозитории. Произведем коммит проиндексированного изменения, а затем еще раз проверим состояние: git commit -m “Added standard HTML page tags” и git status. 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: git add . и git status. Второе изменение было проиндексировано и готово к коммиту. Сделаем коммит второго изменения: git commit -m “Added HTML header”.



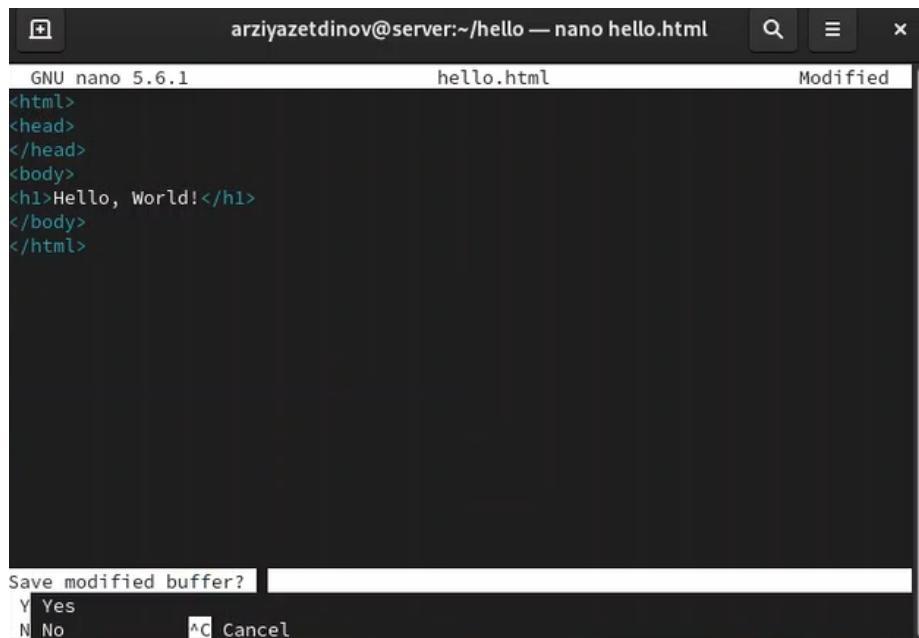
```
arziyazetdinov@server:~/hello
(use "git add <file>..." to update what will be committed)
(use "git restore <file>..." to discard changes in working directory)
 modified: hello.html

no changes added to commit (use "git add" and/or "git commit -a")
[arziyazetdinov@server.arziyazetdinov.net hello]$ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
    modified: hello.html

no changes added to commit (use "git add" and/or "git commit -a")
[arziyazetdinov@server.arziyazetdinov.net hello]$ git add hello.html
[arziyazetdinov@server.arziyazetdinov.net hello]$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    modified: hello.html

[arziyazetdinov@server.arziyazetdinov.net hello]$ git commit
hint: Waiting for your editor to close the file...
[1]+  Stopped                  git commit
[arziyazetdinov@server.arziyazetdinov.net hello]$ git commit
```

Рис. 3.7: Индексация и коммит изменений



```
arziyazetdinov@server:~/hello — nano hello.html
GNU nano 5.6.1
<html>
<head>
</head>
<body>
<h1>Hello, World!</h1>
</body>
</html>

Save modified buffer? [Y/n]
 Y Yes
 N No      ^C Cancel
```

Рис. 3.8: Файл hello.html

3.5 История

Получим список произведенных изменений: `git log`. Посмотрим односрочныи формат истории: `git log --pretty=oneline` (рис. fig. 3.9).

```
nothing to commit, working tree clean
[arziyazetdinov@server.arziyazetdinov.net hello]$ git commit -m "Added HTML header"
On branch master
nothing to commit, working tree clean
[arziyazetdinov@server.arziyazetdinov.net hello]$ git log
commit 2580f5e8e53166dbc2fdb891ae84de9a21b8b2a3 (HEAD -> master)
Author: almazco <azia_04@mail.ru>
Date:   Fri Jun 6 15:59:58 2025 +0000

    Added standard HTML page tags

commit 2c77740690435ff6c4c9fbdea74728e9aad16889
Author: almazco <azia_04@mail.ru>
Date:   Fri Jun 6 15:53:49 2025 +0000

    Initial Commit
[arziyazetdinov@server.arziyazetdinov.net hello]$
```

Рис. 3.9: История

Возвращаться назад в историю очень просто. Команда `checkout` скопирует любой снимок из репозитория в рабочий каталог. Получим хэши предыдущих версий: `git log`. Изучим данные лога и найдем хэш для первого коммита. Используем этот хэш-код (достаточно первых 7 знаков) в команде: `git checkout .`. Затем проверим содержимое файла `hello.html`: `cat hello.html`. Вернемся к последней версии в ветке `main`: `git checkout main` и `cat hello.html` (рис. fig. 3.10).

```
[arziyazetdinov@server.arziyazetdinov.net hello]$ git log
commit 2580f5e8e53166dbc2fdb891ae84de9a21b8b2a3 (HEAD -> master)
Author: almazco <azia_04@mail.ru>
Date:   Fri Jun 6 15:59:58 2025 +0000

    Added standard HTML page tags

commit 2c77740690435ff6c4c9fbdea74728e9aad16889
Author: almazco <azia_04@mail.ru>
Date:   Fri Jun 6 15:53:49 2025 +0000

    Initial Commit
[arziyazetdinov@server.arziyazetdinov.net hello]$
```

Рис. 3.10: Просмотр разных версий репозитория

3.6 Создание тегов версий

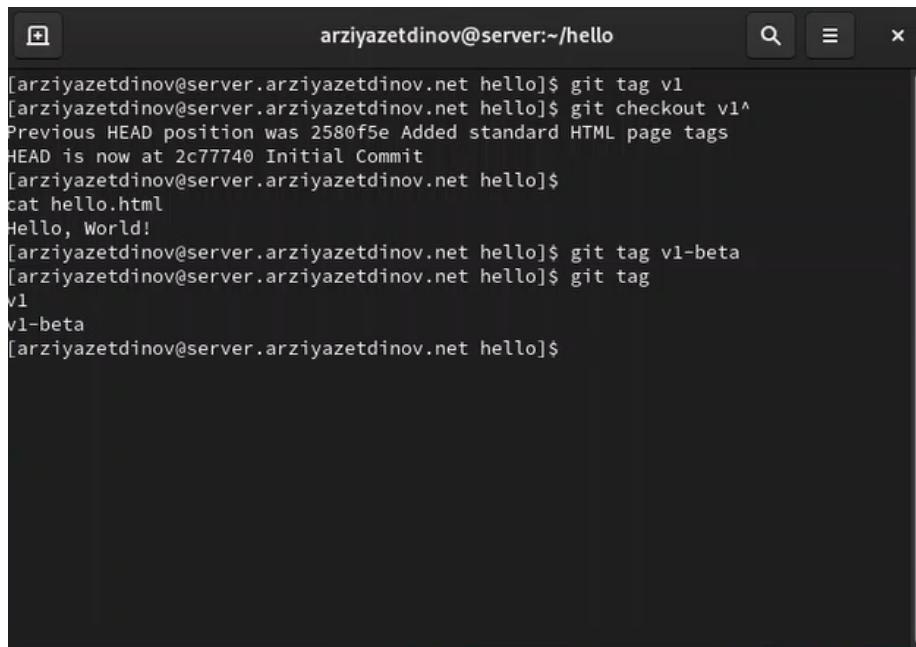
Давайте назовем текущую версию страницы `hello` первой (v1). Создаем тег первой версии: `git tag v1`. Давайте создадим тег для версии, которая идет перед текущей версией и назовем его `v1-beta`. Переключа-

емся на предыдущую версию и просмотрим содержимое файла hello.html:
git checkout v1^ и cat hello.html. Это версия с тегами

и

, но еще пока без

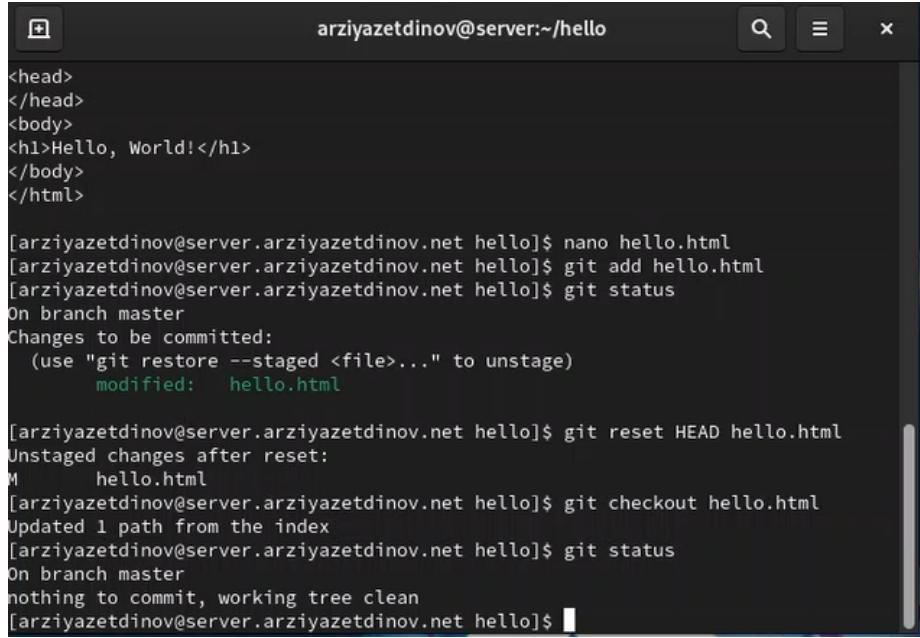
. Даваи~~х~~те сделаем ее версией~~х~~ v1-beta: git tag v1-beta. Теперь попе-
реключаемся между двумя отмеченными версиями: git checkout v1 и git
checkout v1-beta (рис. fig. 3.11).



```
[arziyazetdinov@server.arziyazetdinov.net hello]$ git tag v1
[arziyazetdinov@server.arziyazetdinov.net hello]$ git checkout v1^
Previous HEAD position was 2580f5e Added standard HTML page tags
HEAD is now at 2c77740 Initial Commit
[arziyazetdinov@server.arziyazetdinov.net hello]$
cat hello.html
Hello, World!
[arziyazetdinov@server.arziyazetdinov.net hello]$ git tag v1-beta
[arziyazetdinov@server.arziyazetdinov.net hello]$ git tag
v1
v1-beta
[arziyazetdinov@server.arziyazetdinov.net hello]$
```

Рис. 3.11: Создание тегов версий, переключение между ними

Посмотрим, какие теги доступны, используя команду git tag: git tag.
Также посмотрим теги в логе: git log main –all. Можем видеть теги (v1 и
v1-beta) в логе вместе с именем ветки (main). Кроме того HEAD показывает
коммит, на который~~х~~ вы переключились (на данный~~х~~ момент это v1-beta)
(рис. fig. 3.12).



```
arziyazetdinov@server:~/hello
<head>
</head>
<body>
<h1>Hello, World!</h1>
</body>
</html>

[arziyazetdinov@server.arziyazetdinov.net hello]$ nano hello.html
[arziyazetdinov@server.arziyazetdinov.net hello]$ git add hello.html
[arziyazetdinov@server.arziyazetdinov.net hello]$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    modified:   hello.html

[arziyazetdinov@server.arziyazetdinov.net hello]$ git reset HEAD hello.html
Unstaged changes after reset:
M       hello.html
[arziyazetdinov@server.arziyazetdinov.net hello]$ git checkout hello.html
Updated 1 path from the index
[arziyazetdinov@server.arziyazetdinov.net hello]$ git status
On branch master
nothing to commit, working tree clean
[arziyazetdinov@server.arziyazetdinov.net hello]$
```

Рис. 3.12: Доступные теги

3.7 Отмена локальных изменений (до индексации)

Перейдем на последний коммит ветки main: git checkout main (рис. fig. 3.13). Внесем изменение в файл hello.html в виде нежелательного комментария (рис. fig. 3.14). Проверим состояние рабочего каталога: git status. Мы видим, что файл hello.html был изменен, но еще не проиндексирован. Используем команду git checkout для переключения версии файла hello.html в репозитории: git checkout hello.html, git status и cat hello.html. Команда git status показывает нам, что не было произведено никаких изменений, не зафиксированных в рабочем каталоге.

```
1 file changed, 1 insertion(+), 1 deletion(-)
[arziyazetdinov@server.arziyazetdinov.net hello]$ git revert HEAD
hint: Waiting for your editor to close the file... error: There was a problem with the editor 'vi'.
Please supply the message using either -m or -F option.
[arziyazetdinov@server.arziyazetdinov.net hello]$ git log
commit fa4eefb4b173e76adda87bf642582ddce617f2 (HEAD -> master)
Author: almazco <azia_04@mail.ru>
Date:   Fri Jun 6 16:08:27 2025 +0000

    Oops, we didn't want this commit

commit 2580f5e8e53166dbc2fdb891ae84de9a21b8b2a3 (tag: v1)
Author: almazco <azia_04@mail.ru>
Date:   Fri Jun 6 15:59:58 2025 +0000

    Added standard HTML page tags

commit 2c77740690435ff6c4c9fbdea74728e9aad16889 (tag: v1-beta)
Author: almazco <azia_04@mail.ru>
Date:   Fri Jun 6 15:53:49 2025 +0000

    Initial Commit
[arziyazetdinov@server.arziyazetdinov.net hello]$
```

Рис. 3.13: Отмена локальных изменений (до индексации)

```
GNU nano 5.6.1          hello.html
<html>
<head>
</head>
<body>
<h1>Hello, World!</h1>
</body>
</html>
```

[Read 8 lines]

^G Help ^O Write Out ^W Where Is ^K Cut ^T Execute ^C Location
^X Exit ^R Read File ^\ Replace ^U Paste ^J Justify ^_ Go To Line

Рис. 3.14: Изменения в файле hello.html

3.8 Отмена проиндексированных изменений (перед коммитом)

Изменим файл (рис. fig. 3.15).

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

Рис. 3.15: Изменения в файле hello.html

Проиндексируем изменения: git add hello.html. Проверим состояние нежелательного изменения: git status. Состояние показывает, что изменение было проиндексировано и готово к коммиту. Выполним сброс буферной зоны: git reset HEAD hello.html. Команда git reset сбрасывает буферную зону к HEAD. Это очищает буферную зону от изменений, которые мы только что проиндексировали. Команда git reset (по умолчанию) не изменяет рабочий каталог. Поэтому рабочий каталог все еще содержит нежелательные комментарии. Мы можем использовать команду git checkout, чтобы удалить нежелательные изменения в рабочем каталоге. Переключимся на версию коммита: git checkout hello.html и git status. Наш рабочий каталог опять чист (рис. fig. 3.16).

```
[arziyazetdinov@server.arziyazetdinov.net hello]$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    renamed:   hello.html -> lib/hello.html

[arziyazetdinov@server.arziyazetdinov.net hello]$
```

Рис. 3.16: Отмена локальных изменений (перед коммитом)

3.9 Отмена коммитов

Отменим коммит путем создания нового коммита, отменяющего нежелательные изменения. Изменим файл hello.html:

```
<html>
```

```
  <head>
```

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

Выполним: `git add hello.html` и `git commit -m "Oops, we didn't want this commit"` (рис. fig. ??). Сделаем коммит с новыми изменениями, отменяющими предыдущие: `git revert HEAD`. В редакторе отредактируем коммит-сообщение . Сохраним и закроем файл. Проверим лог: `git log`.

3.10 Удаление коммитов из ветки

Прежде чем удалить коммиты, давайте отметим последний коммит тегом, чтобы потом можно было его найти: `git tag oops`. Глядя на историю лога, видим, что коммит с тегом «v1» является коммитом, предшествующим ошибочному коммиту. Давайте сбросим ветку до этой точки. Поскольку ветка имеет тег, мы можем использовать имя тега в команде сброса (если она не имеет тега, мы можем использовать хэш-значение): `git reset –hard v1` и `git log`. Наша ветка `main` теперь указывает на коммит `v1`, а коммитов `Oops` и `Revert Oops` в ветке уже нет .

Давайте посмотрим на все коммиты: `git log –all` (рис. fig. ??). Мы видим, что ошибочные коммиты не исчезли. Они все еще находятся в репозитории.

3.11 Удаление тега oops

Давай~~те~~ удалим тег oops его и коммиты, на которые он ссылался, сборщиком мусора: git tag -d oops и git log –all. Тег «oops» больше не будет отображаться в репозитории .

3.12 Внесение изменений~~и~~ в коммиты

Изменим страницу, а затем сделаем коммит.

Обновим страницу hello, включив в нее email

Изменим предыдущий~~и~~ коммит: git add hello.html и git commit –amend -m “Add an author/email comment”. Просмотрим историю: git log. Мы можем увидеть, что оригинальный~~и~~ коммит «автор» заменен коммитом «автор/email».

3.13 Перемещение файлов

Переместим файл hello.html в каталог lib: mkdir lib, git mv hello.html lib и git status. Сделаем коммит этого перемещения: git commit -m “Moved hello.html to lib” (рис. fig. 3.13).

Перемещение файлов ## Подробнее о структуре

Добавим файл index.html в наш репозиторий~~и~~:

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

Добавим файл и сделаем коммит: `git add index.html` и `git commit -m "Added index.html"` (рис. fig. ??). Теперь при открытии `index.html`, видим кусок страницы `hello` в маленьком окошке

3.14 Git внутри: Каталог .git

Выполним: `ls -C .git`. Это каталог, в котором хранится вся информация git. Выполним: `ls -C .git/objects`. Видим набор каталогов, имена которых состоят из 2 символов. Имена каталогов являются первыми двумя буквами хэша sha1 объекта, хранящегося в git. Выполним: `ls -C .git/objects/01`. Смотрим в один из каталогов с именем из 2 букв. Видим файл с именем из 38 символов. Выполним: `cat .git/config`. Это файл конфигурации, создающийся для каждого конкретного проекта. Записи в этом файле будут перезаписывать записи в файле `.gitconfig` вашего главного каталога, по крайней мере в рамках этого проекта. Выполним: `ls .git/refs`, `ls .git/refs/heads`, `ls .git/refs/tags` и `cat .git/refs/tags/v1`. Выполним: `cat .git/HEAD`. Файл HEAD содержит ссылку на текущую ветку, в данный момент это ветка `main` (рис. fig. 3.17).

```

[arziyazetdinov@server.arziyazetdinov.net hello]$ ls -C .git/objects
00 11 2c 46 52 63 6e 8a a3 b3 bd d8 info
0b 25 2d 4e 54 68 81 9b ab b5 cb fa pack
[arziyazetdinov@server.arziyazetdinov.net hello]$ ls -C .git/objects/<dir>
bash: syntax error near unexpected token `newline'
[arziyazetdinov@server.arziyazetdinov.net hello]$ ls -C .git/objects/
00 11 2c 46 52 63 6e 8a a3 b3 bd d8 info
0b 25 2d 4e 54 68 81 9b ab b5 cb fa pack
[arziyazetdinov@server.arziyazetdinov.net hello]$ ls -C .git/objects/00
5524a67923c719f956232e2584275451bc0bba
[arziyazetdinov@server.arziyazetdinov.net hello]$ cat .git/config
[core]
    repositoryformatversion = 0
    filemode = true
    bare = false
    logallrefupdates = true
[arziyazetdinov@server.arziyazetdinov.net hello]$ ls .git/refs
heads tags
[arziyazetdinov@server.arziyazetdinov.net hello]$ ls .git/refs/heads
master
[arziyazetdinov@server.arziyazetdinov.net hello]$
ls .git/refs/tags
v1 v1-beta
[arziyazetdinov@server.arziyazetdinov.net hello]$

```

Рис. 3.17: Git внутри: Каталог .git

3.15 Работа непосредственно с объектами git

Поиск последнего коммита: git log --max-count=1. Вывод последнего коммита с помощью SHA1 хэша: git cat-file -t и git cat-file -p . Поиск дерева, используем SHA1 хэш из строки «дерева», из списка выше: git cat-file -p . Вывод каталога lib: git cat-file -p . Вывод файла hello.html: git cat-file -p (рис. fig. 3.18).

```

[arziyazetdinov@server.arziyazetdinov.net hello]$
ls .git/refs/tags
v1 v1-beta
[arziyazetdinov@server.arziyazetdinov.net hello]$ cat .git/refs/tags/v1
2580f5e8e53166dbc2fdb891ae84de9a21b8b2a3
[arziyazetdinov@server.arziyazetdinov.net hello]$ cat .git/HEAD
ref: refs/heads/master
[arziyazetdinov@server.arziyazetdinov.net hello]$ git log --max-count=1
commit 815ba8ac361b4cb889cccd7a0c3alebb2c89f59 (HEAD -> master)
Author: almazco <azia_04@mail.ru>
Date:   Fri Jun 6 16:16:45 2025 +0000

    Added index.html.
[arziyazetdinov@server.arziyazetdinov.net hello]$ git cat-file -t <hash>^C
[arziyazetdinov@server.arziyazetdinov.net hello]$ ^C
[arziyazetdinov@server.arziyazetdinov.net hello]$ git log --max-count=1

```

Рис. 3.18: Работа с объектами git

3.16 Создание ветки

Создадим ветку «style»: git checkout -b style и git status. Добавим файл стилей style.css: touch lib/style.css, git add lib/style.css и git commit -m “Added css stylesheet” (рис. fig. 3.19).

```
[arziyazetdinov@server.arziyazetdinov.net hello]$ git checkout -b style
Switched to a new branch 'style'
[arziyazetdinov@server.arziyazetdinov.net hello]$ git status
On branch style
nothing to commit, working tree clean
[arziyazetdinov@server.arziyazetdinov.net hello]$ touch lib/style.css
[arziyazetdinov@server.arziyazetdinov.net hello]$ nano style.css
[arziyazetdinov@server.arziyazetdinov.net hello]$ git add lib/style.css
[arziyazetdinov@server.arziyazetdinov.net hello]$ git commit -m "Added css style
sheet"
[style 027e001] Added css stylesheet
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 lib/style.css
[arziyazetdinov@server.arziyazetdinov.net hello]$ nano
```

Рис. 3.19: Создание ветки style и файла style.css

Обновим файл hello.html, чтобы использовать стили style.css (рис. fig. 3.20).

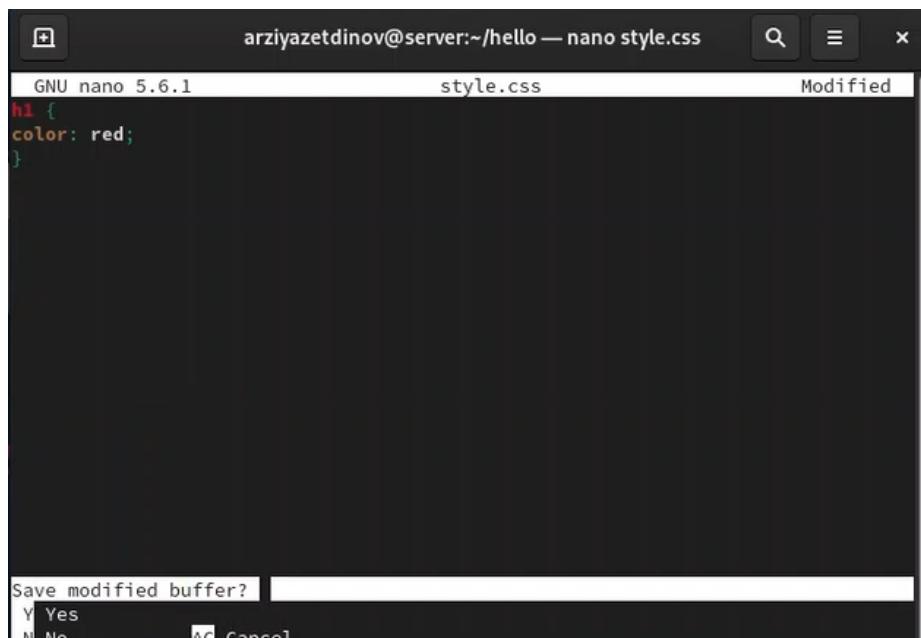


Рис. 3.20: Файл hello.html

Выполним: git add lib/hello.html и git commit -m “Hello uses style.css”. Изменим index.html, чтобы он тоже использовал style.css. Выполним: git add index.html и git commit -m “Updated index.html” (рис. fig. 3.21).

```
GNU nano 5.6.1          hello.html
<!-- Author: Almaz R. Ziyazetdinov (azia_04@mail.ru)--&gt;
&lt;html&gt;
&lt;head&gt;
&lt;/head&gt;
&lt;body&gt;
&lt;h1&gt;Hello, World!&lt;/h1&gt;
&lt;/body&gt;
&lt;/html&gt;

[ Read 8 lines ]
^G Help      ^O Write Out  ^W Where Is  ^K Cut      ^T Execute  ^C Location
^X Exit     ^R Read File  ^\ Replace   ^U Paste    ^J Justify  ^_ Go To Line</pre>
```

Рис. 3.21: Файлы hello.html и index.html

3.17 Навигация по веткам

Теперь в вашем проекте есть две ветки: git log --all (рис. fig. 3.22).

```
arziyazetdinov@server:~/hello — git log --all
1 file changed, 2 insertions(+)
[arziyazetdinov@server.arziyazetdinov.net hello]$ nano index.html
[arziyazetdinov@server.arziyazetdinov.net hello]$ nano index.html
[arziyazetdinov@server.arziyazetdinov.net hello]$ git add index.html
[arziyazetdinov@server.arziyazetdinov.net hello]$ git commit -m "Updated index.html"
[style 11a7f3a] Updated index.html
1 file changed, 4 insertions(+)
[arziyazetdinov@server.arziyazetdinov.net hello]$ git log --all
commit 11a7f3ad7befc5a040a3d3e1843ae4a3bdcca (HEAD -> style)
Author: almazco <azia_04@mail.ru>
Date:   Fri Jun 6 16:40:27 2025 +0000
        Updated index.html

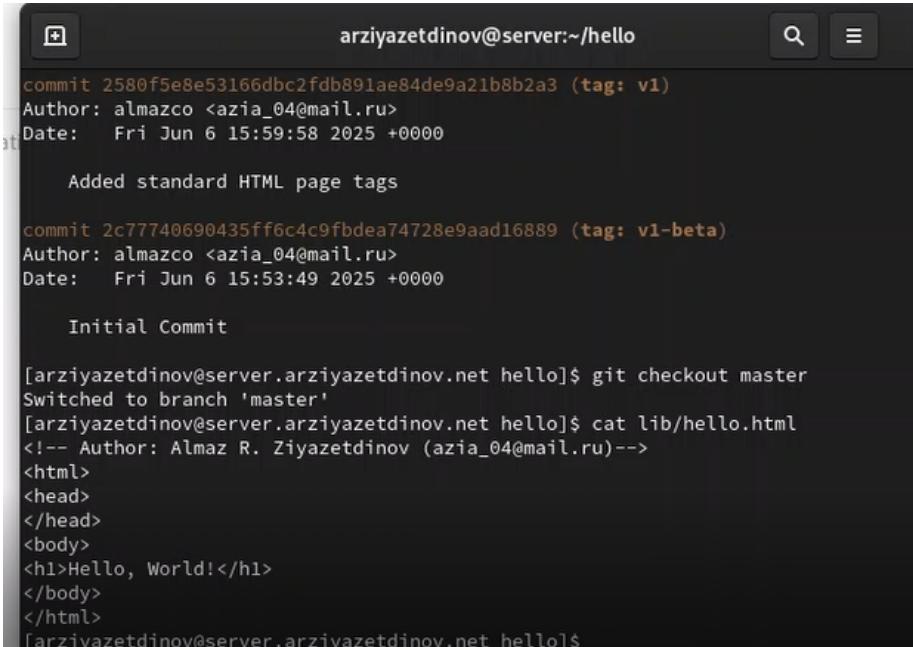
commit d1b0d99999530aedd08ab034d50ab177ca4eb9ad
Author: almazco <azia_04@mail.ru>
Date:   Fri Jun 6 16:39:26 2025 +0000
        Hello uses style.css

commit 027e001fdbdc4a6605b792c1e7e7841d490e4c38
Author: almazco <azia_04@mail.ru>
Date:   Fri Jun 6 16:25:03 2025 +0000
```

Рис. 3.22: Логи

Переключение на ветку main: git checkout main и cat lib/hello.html. Сейчас мы находимся на ветке main. Это заметно по тому, что файл hello.html

не использует стили style.css. Вернемся к ветке style: git checkout style и cat lib/hello.html. Содержимое lib/hello.html подтверждает, что мы вернулись на ветку style (рис. fig. 3.23).



```
arziyazetdinov@server:~/hello
commit 2580f5e8e53166dbc2fdb891ae84de9a21b8b2a3 (tag: v1)
Author: almazco <azia_04@mail.ru>
Date:   Fri Jun 6 15:59:58 2025 +0000

    Added standard HTML page tags

commit 2c77740690435ff6c4c9fbdea74728e9aad16889 (tag: v1-beta)
Author: almazco <azia_04@mail.ru>
Date:   Fri Jun 6 15:53:49 2025 +0000

    Initial Commit

[arziyazetdinov@server.arziyazetdinov.net hello]$ git checkout master
Switched to branch 'master'
[arziyazetdinov@server.arziyazetdinov.net hello]$ cat lib/hello.html
<!-- Author: Almaz R. Ziyazetdinov (azia_04@mail.ru)-->
<html>
<head>
</head>
<body>
<h1>Hello, World!</h1>
</body>
</html>
[arziyazetdinov@server.arziyazetdinov.net hello]$
```

Рис. 3.23: Навигация по веткам

3.18 Изменения в ветке main

Пока вы меняли ветку style, кто-то решил обновить ветку main. Они добавили файл README.md. Создадим файл README в ветке main: git checkout main. Создадим файл README.md: echo “This is the Hello World example from the git tutorial.” > README.md. Сделаем коммит изменений README.md в ветку main: git add README.md и git commit -m “Added README”. Используем следующую лог-команду для просмотра веток и их отличий: git log –graph –all (рис. fig. 3.24).

```
arziyazetdinov@server:~/hello — git log --graph --all
* commit d8d23c7b5d82c64131426ba58619c10c2470a381 (HEAD -> master)
  Author: almazco <azia_04@mail.ru>
  Date:   Fri Jun 6 16:44:33 2025 +0000

    Added README

* commit 11a7f3ad7bef9bc5a040a3d3e1843ae4a3bdcca (style)
  Author: almazco <azia_04@mail.ru>
  Date:   Fri Jun 6 16:40:27 2025 +0000

    Updated index.html

* commit d1b0d99999530aed08ab034d50ab177ca4eb9ad
  Author: almazco <azia_04@mail.ru>
  Date:   Fri Jun 6 16:39:26 2025 +0000

    Hello uses style.css

* commit 027e001fdbdc4a6605b792c1e7e7841d490e4c38
  Author: almazco <azia_04@mail.ru>
  Date:   Fri Jun 6 16:25:03 2025 +0000

    Added css stylesheet
```

Рис. 3.24: Изменения в ветке main

3.19 Слияние

Давай~~х~~те вернемся к ветке style и сольем main c style: git checkout style, git merge main, git log –graph –all (рис. fig. 3.25).

```
arziyazetdinov@server:~/hello
Moved hello.html to lib

* commit b53b15b9c1d3acd08bea6d595d40e4b8b192a118
  Author: almazco <azia_04@mail.ru>
  Date:   Fri Jun 6 16:12:32 2025 +0000

    Add an author/email comment

* commit 2580f5e8e53166dbc2fdb891ae84de9a21b8b2a3 (tag: v1)
  Author: almazco <azia_04@mail.ru>
  Date:   Fri Jun 6 15:59:58 2025 +0000

    Added standard HTML page tags

* commit 2c77740690435ff6c4c9fbdea74728e9aad16889 (tag: v1-beta)
  Author: almazco <azia_04@mail.ru>
  Date:   Fri Jun 6 15:53:49 2025 +0000

    Initial Commit
[arziyazetdinov@server.arziyazetdinov.net hello]$ git checkout style
Switched to branch 'style'
[arziyazetdinov@server.arziyazetdinov.net hello]$ git rebase master
Successfully rebased and updated refs/heads/style.
[arziyazetdinov@server.arziyazetdinov.net hello]$
```

Рис. 3.25: Слияние main c style

3.20 Создание конфликта

Вернемся в main и создадим конфликт. Файл lib/hello.html

Выполним: git add lib/hello.html и git commit -m ‘Life is great’. Просмотр веток: git log –graph –all. После коммита «Added README» ветка main была объединена с веткой style, но в настоящее время в main есть дополнительный коммит, который не был слит с style. Последнее изменение в main конфликтует с некоторыми изменениями в style

3.21 Разрешение конфликтов

Слияние main с веткой style. Теперь вернемся к ветке style и попытаемся объединить ее с новой веткой main: git checkout style, git merge main

Откроем lib/hello.html. Первый раздел — версия текущей ветки (style). Второй раздел — версия ветки main.

Решение конфликта. Внесем изменения в lib/hello.html

Сделаем коммит решения конфликта: git add lib/hello.html и git commit -m “Merged main fixed conflict”.

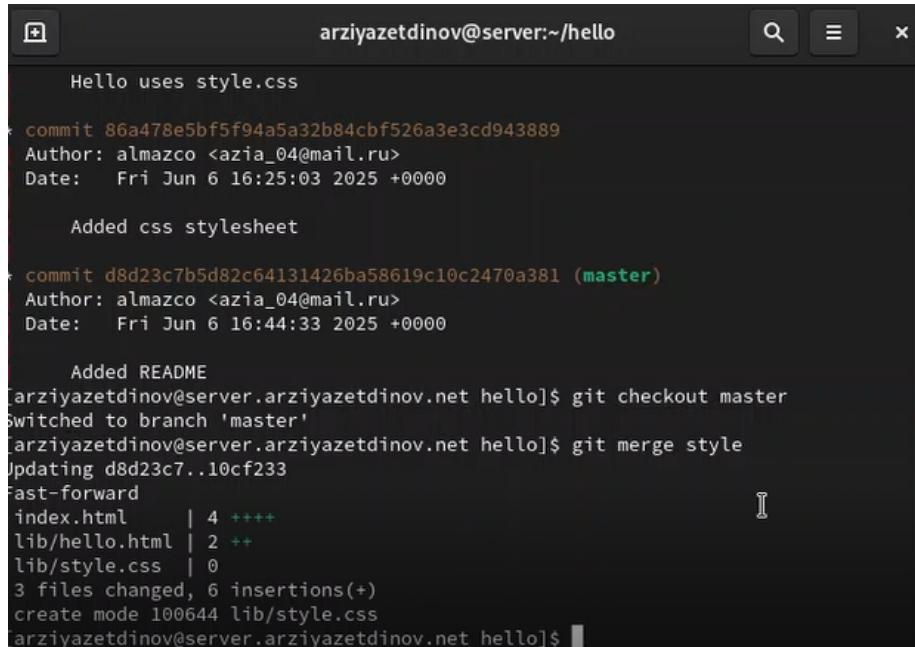
3.22 Сброс ветки style

Вернемся на ветку style к точке перед тем, как мы слили ее с веткой main: git checkout style и git log –graph

Мы видим, что коммит «Updated index.html» был последним на ветке style перед слиянием. Давайте сбросим ветку style к этому коммиту: git reset –hard . Поищем лог ветки style. У нас в истории больше нет коммитов слияния: git log –graph –all .

3.23 Сброс ветки main

Сброс ветки main: git checkout main, git log –graph (рис. fig. 3.26).



```
arziyazetdinov@server:~/hello
Hello uses style.css

commit 86a478e5bf5f94a5a32b84cbf526a3e3cd943888
Author: almazco <azia_04@mail.ru>
Date:   Fri Jun 6 16:25:03 2025 +0000

    Added css stylesheet

commit d8d23c7b5d82c64131426ba58619c10c2470a381 (master)
Author: almazco <azia_04@mail.ru>
Date:   Fri Jun 6 16:44:33 2025 +0000

    Added README
[arziyazetdinov@server.arziyazetdinov.net hello]$ git checkout master
switched to branch 'master'
[arziyazetdinov@server.arziyazetdinov.net hello]$ git merge style
Updating d8d23c7..10cf233
Fast-forward
 index.html      | 4 +---
 lib/hello.html  | 2 ++
 lib/style.css   | 0
 3 files changed, 6 insertions(+)
 create mode 100644 lib/style.css
[arziyazetdinov@server.arziyazetdinov.net hello]$
```

Рис. 3.26: Ветка main

Коммит «Added README» идет непосредственно перед коммитом конфликтующего интерактивного режима. Мы сбросим ветку main к коммиту «Added README»: git reset –hard и git log –graph –all

Мы вернулись в точку до первого слияния и хотим перенести изменения из ветки main в нашу ветку style: git checkout style, git rebase main и git log –graph

3.24 Слияние в ветку main

Мы поддерживали соответствие ветки style с веткой main (с помощью rebase), теперь давайте сольем изменения style в ветку main: git checkout main и git merge style . Посмотрим логи. Теперь ветки style и main идентичны.

3.25 Клонирование репозиториев

Переходим в рабочий каталог aamishina и сделаем клон репозитория hello: cd .. pwd ls, git clone hello cloned_hello, ls. Давайте взглянем на клонированный репозиторий: cd cloned_hello, ls. Видим список всех файлов на верхнем уровне оригинального репозитория README.md, index.html и lib. Просмотрим историю репозитория: git log –all. Видим список всех коммитов в новом репозитории, и он совпадает с историей коммитов в оригинальном репозитории. Единственная разница в названиях веток

3.26 Что такое origin?

Выполним: git remote. Видим, что клонированный репозиторий знает об имени по умолчанию удаленного репозитория. Посмотрим более подробную информацию: git remote show origin. Давайте посмотрим на ветки, доступные в нашем клонированном репозитории: git branch. Как видим, в списке только ветка main. Для того, чтобы увидеть все ветки, вводим команду: git branch -a (рис. fig. 3.27).

```
Date: Fri Jun 6 16:44:33 2025 +0000
      Added README
[arziyazetdinov@server.arziyazetdinov.net hello]$ cd
[arziyazetdinov@server.arziyazetdinov.net ~]$ cd ..
[arziyazetdinov@server.arziyazetdinov.net home]$ pwd
/home
[arziyazetdinov@server.arziyazetdinov.net home]$ ls
arziyazetdinov user vagrant
[arziyazetdinov@server.arziyazetdinov.net home]$ cd /hello
bash: cd: /hello: No such file or directory
[arziyazetdinov@server.arziyazetdinov.net home]$ cd hello
bash: cd: hello: No such file or directory
[arziyazetdinov@server.arziyazetdinov.net home]$ cd
```

Рис. 3.27: Информация о репозитории и ветках

3.27 Изменение оригинального репозитория

Внесем изменения в оригинальный репозиторий hello: cd ../hello. Внесем изменения в файл README.md: This is the Hello World example from the git tutorial. Сделаем коммит: git add README и git commit -m "Changed README in original repo"

Теперь в оригинальном репозитории есть более поздние изменения, которых нет в клонированной версии.

Извлечение изменений: cd ../cloned_hello, git fetch, git log –all

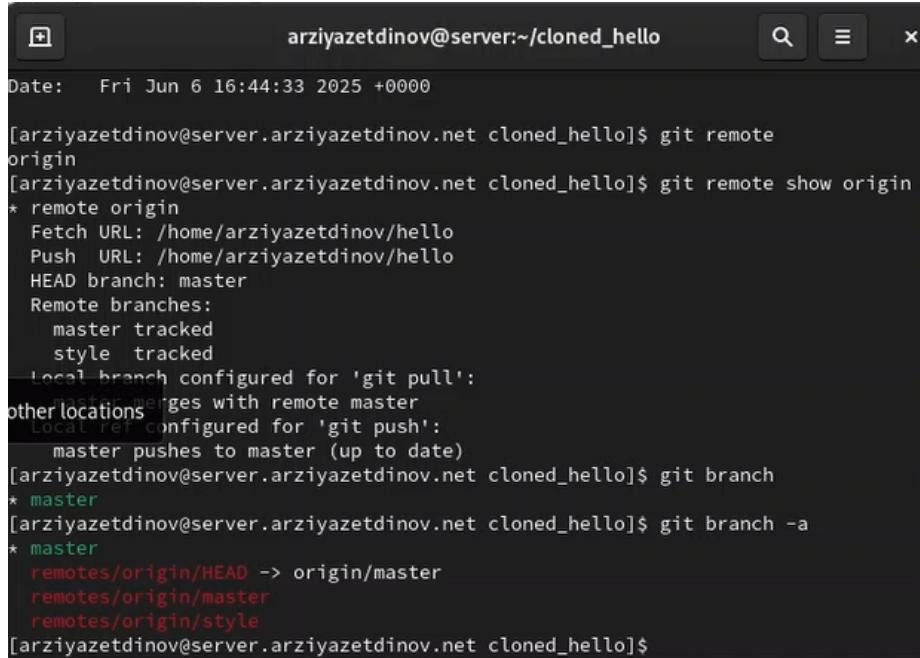
Сейчас мы находимся в репозитории cloned_hello. На данный момент в репозитории есть все коммиты из оригинального репозитория, но они не интегрированы в локальные ветки клонированного репозитория. Выводом является то, что команда git fetch будет извлекать новые коммиты из удаленного репозитория, но не будет сливать их с вашими наработками в локальных ветках. Проверим README.md, он не изменился

3.28 Слияние извлеченных изменений

Сольем извлеченные изменения в локальную ветку main: git merge origin/main. Еще раз проверим файл README.md. Сейчас мы видим изменения (рис. fig. ??).

3.29 Добавление ветки наблюдения

Добавим локальную ветку, которая отслеживает удаленную ветку: git branch –track style origin/style, git branch -a, git log –max-count=2. Теперь мы можем видеть ветку style в списке веток и логе



```
arziyazetdinov@server:~/cloned_hello
Date: Fri Jun 6 16:44:33 2025 +0000

[arziyazetdinov@server.arziyazetdinov.net cloned_hello]$ git remote
origin
[arziyazetdinov@server.arziyazetdinov.net cloned_hello]$ git remote show origin
* remote origin
  Fetch URL: /home/arziyazetdinov/hello
  Push URL: /home/arziyazetdinov/hello
  HEAD branch: master
  Remote branches:
    master tracked
    style tracked
  Local branch configured for 'git pull':
    master merges with remote master
  other locations
  Local ref configured for 'git push':
    master pushes to master (up to date)
[arziyazetdinov@server.arziyazetdinov.net cloned_hello]$ git branch
* master
[arziyazetdinov@server.arziyazetdinov.net cloned_hello]$ git branch -a
* master
  remotes/origin/HEAD -> origin/master
  remotes/origin/master
  remotes/origin/style
[arziyazetdinov@server.arziyazetdinov.net cloned_hello]$
```

Рис. 3.28: Добавление ветки наблюдения

3.30 Создаите чистый репозиторий

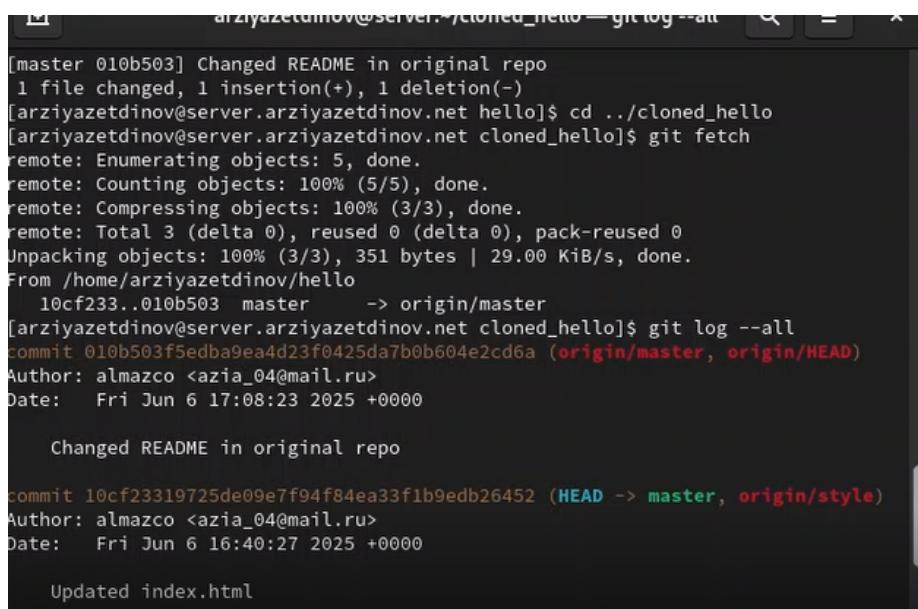
Создадим чистый репозиторий: cd .., git clone –bare hello hello.git, ls hello.git. Сейчас мы находимся в рабочем каталоге. Мы видим, что в репозитории hello.git нет рабочего каталога

3.31 Добавление удаленного репозитория

Давайте добавим репозиторий hello.git к нашему оригинальному репозиторию: cd hello и git remote add shared ../hello.git . Создадим изменения для отправки: файл README.md, сделаем его коммит. Теперь отправим изменения в общий репозиторий: git push shared main. Общим называется репозиторий, получающий отправленные нами изменения.

3.32 Извлечение общих изменений

Быстро переключаемся в клонированный репозиторий и извлекаем изменения, только что отправленные в общий репозиторий: cd .../cloned_hello. Сейчас мы находимся в репозитории cloned_hello: git remote add shared/hello.git, git branch -track shared main, git pull shared main, cat README.md (рис. fig. 3.29).



```
arziyazetdinov@server:~/cloned_hello$ git log --all
[master 010b503] Changed README in original repo
 1 file changed, 1 insertion(+), 1 deletion(-)
[arziyazetdinov@server.arziyazetdinov.net hello]$ cd ../cloned_hello
[arziyazetdinov@server.arziyazetdinov.net cloned_hello]$ git fetch
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), 351 bytes | 29.00 KiB/s, done.
From /home/arziyazetdinov/hello
  10cf233..010b503  master      -> origin/master
[arziyazetdinov@server.arziyazetdinov.net cloned_hello]$ git log --all
commit 010b503f5edba9ea4d23f0425da7b0b604e2cd6a (origin/master, origin/HEAD)
Author: almazco <azia_04@mail.ru>
Date:   Fri Jun 6 17:08:23 2025 +0000

    Changed README in original repo

commit 10cf23319725de09e7f94f84ea33f1b9edb26452 (HEAD -> master, origin/style)
Author: almazco <azia_04@mail.ru>
Date:   Fri Jun 6 16:40:27 2025 +0000

    Updated index.html
```

Рис. 3.29: Извлечение общих изменений

4 Выводы

В ходе выполнения данной лабораторной работы я приобрела практические навыки работы с системой управления версиями Git.

Список литературы