

# **Лабораторная работа №1**

**Работа с git**

Астраханцева А. А.

# Содержание

1	Цель работы	4
2	Выполнение лабораторной работы	5
3	Выводы	42

## Список иллюстраций

2.1	Параметры установки окончаний строк . . . . .	5
2.2	Создание файла, репозитория и добавление файлов в репозиторий	6
2.3	Изменение файла, создание комита . . . . .	7
2.4	Изменение файла и создание коммита . . . . .	9
2.5	Просмотр истории . . . . .	9
2.6	Варианты просмотра лога . . . . .	10
2.7	Возвращение к версии репозитория используя хэш . . . . .	11
2.8	Создание тэгов и переключение по ним . . . . .	12
2.9	Переключение по тэгам . . . . .	13
2.10	Имена тэгов в логе . . . . .	13
2.11	Отмена индексации файла . . . . .	16
2.12	Отмена коммитов . . . . .	17
2.13	Удаление коммитов . . . . .	19
2.14	Удаление тэга . . . . .	20
2.15	Изменение коммита . . . . .	21
2.16	Перемещение файла средствами git . . . . .	23
2.17	Структура git . . . . .	24
2.18	Содержание git/HEAD . . . . .	25
2.19	Создание ветки, редактирование файлов . . . . .	28
2.20	Просмотр лога . . . . .	29
2.21	Создание отличий в ветках . . . . .	30
2.22	Просмотр графа . . . . .	31
2.23	Просмотр графа . . . . .	32
2.24	Сброс ветки . . . . .	33
2.25	Перебазирование . . . . .	34
2.26	Клонирование репозитория . . . . .	35
2.27	История репозитория . . . . .	36
2.28	Origin . . . . .	37
2.29	Извлечение изменений . . . . .	38
2.30	Различные команды git . . . . .	40

# 1 Цель работы

Приобретение навыков работы с git и markdown.

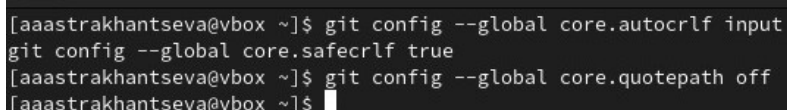
## 2 Выполнение лабораторной работы

### 1. Подготовка

Настройка `core.autocrlf` с параметрами `true` и `input` делает все переводы строк текстовых файлов в главном репозитории одинаковыми.

`core.autocrlf true` - git автоматически конвертирует CRLF->LF при коммите и обратно LF->CRLF при выгрузке кода из репозитория на файловую систему (используют в Windows). `core.autocrlf input` - конвертация CRLF в LF только при коммитах (используют в Mac/Linux).

Если `core.safecrlf` установлен в `true` или `warn`, git проверяет, если преобразование является обратимым для текущей настройки `core.autocrlf`. `core.safecrlf true` - отвержение необратимого преобразования lf<->crlf. Полезно, когда специфические бинарники похожие на текстовые файлы. `core.safecrlf warn` - печать только предупреждение, но принимает необратимый переход. (рис. 2.1).



```
[aaastrakhantseva@vbox ~]$ git config --global core.autocrlf input
git config --global core.safecrlf true
[aaastrakhantseva@vbox ~]$ git config --global core.quotepath off
[aaastrakhantseva@vbox ~]$
```

Рис. 2.1: Параметры установки окончаний строк

### 2. Создание проекта

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`.

```
mkdir hello
```

```
cd hello
touch hello.html
echo "Hello, World!" > hello.html
```

Создадим git репозиторий из этого каталога, выполнив команду:

```
git init
```

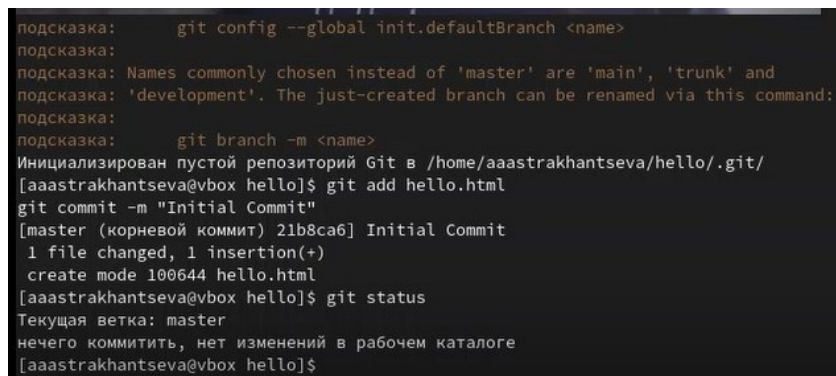
Добавим файл в репозиторий:

```
git add hello.html
git commit -m "Initial Commit"
```

Проверим текщее состояние репозитория с помощью команды:

```
git status
```

Команда проверки состояния сообщит, что коммитить нечего. Это означает, что в репозитории хранится текущее состояние рабочего каталога, и нет никаких изменений, ожидающих записи. (рис. 2.2).



```
подсказка:      git config --global init.defaultBranch <name>
подсказка:
подсказка: Names commonly chosen instead of 'master' are 'main', 'trunk' and
подсказка: 'development'. The just-created branch can be renamed via this command:
подсказка:
подсказка:      git branch -m <name>
Инициализирован пустой репозиторий Git в /home/aaastrakhantseva/hello/.git/
[aaastrakhantseva@vbox hello]$ git add hello.html
git commit -m "Initial Commit"
[master (корневой коммит) 21b8ca6] Initial Commit
1 file changed, 1 insertion(+)
 create mode 100644 hello.html
[aaastrakhantseva@vbox hello]$ git status
Текущая ветка: master
нечего коммитить, нет изменений в рабочем каталоге
[aaastrakhantseva@vbox hello]$
```

Рис. 2.2: Создание файла, репозитория и добавление файлов в репозиторий

### 3. Внесение изменений

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

<h1>Hello, World!</h1>

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 вам подсказку о том, что нужно делать дальше. Если нужно добавить эти изменения в репозиторий, используем команду git add. В противном случае используем команду

```
git checkout
```

для отмены изменений.

#### 4. Индексация изменений

Теперь выполним команду git, чтобы проиндексировать изменения. Проверим состояние.

```
git add hello.html
```

```
git status
```

Изменения файла hello.html были проиндексированы. Это означает, что git теперь знает об изменении, но изменение пока не записано в репозиторий. Следующий коммит будет включать в себя проиндексированные изменения (рис. 2.3).

```
[aaastrakhantseva@vbox hello]$ gedit hello.html
[aaastrakhantseva@vbox hello]$ git status
Текущая ветка: master
Изменения, которые не в индексе для коммита:
  (используйте «git add <файл>...», чтобы добавить файл в индекс)
  (используйте «git restore <файл>...», чтобы отменить изменения в рабочем каталоге)
    изменено:      hello.html

индекс пуст (используйте «git add» и/или «git commit -a»)
[aaastrakhantseva@vbox hello]$ git add hello.html
git status
Текущая ветка: master
Изменения, которые будут включены в коммит:
  (используйте «git restore --staged <файл>...», чтобы убрать из индекса)
    изменено:      hello.html
[aaastrakhantseva@vbox hello]$
```

Рис. 2.3: Изменение файла, создание коммита

Сделаем коммит и проверим состояние.

```
git commit
```

```
git status
```

Рабочий каталог чистый, можно продолжить работу. Добавим стандартные теги страницы. Изменим страницу «Hello, World», чтобы она содержала стандартные теги

и

:

```
<html>
```

```
<body>
```

```
<h1>Hello, World!</h1>
```

```
</body>
```

```
</html>
```

Теперь добавим это изменение в индекс git.

```
git add hello.html
```

Теперь добавим заголовки HTML (секцию  
) к странице «Hello, World»:

```
<html>
```

```
<head>
```

```
</head>
```

```
<body>
```

```
<h1>Hello, World!</h1>
```

```
</body>
```

```
</html>
```

Проверим текущий статус:



git status

Теперь hello.html указан дважды в состоянии. Первое изменение (добавление стандартных тегов) проиндексировано и готово к коммиту. Второе изменение (добавление заголовков HTML) является непроиндексированным (рис. 2.4).

```
[aaastrakhantseva@vbox hello]$ gedit hello.html
[aaastrakhantseva@vbox hello]$ git add hello.html
[aaastrakhantseva@vbox hello]$ gedit hello.html
[aaastrakhantseva@vbox hello]$ git status
Текущая ветка: master
Изменения, которые будут включены в коммит:
(используйте «git restore --staged <файл>...», чтобы убрать из индекса)
    изменено:      hello.html

Изменения, которые не в индексе для коммита:
(используйте «git add <файл>...», чтобы добавить файл в индекс)
(используйте «git restore <файл>...», чтобы отменить изменения в рабочем каталоге)
    изменено:      hello.html

[aaastrakhantseva@vbox hello]$
```

Рис. 2.4: Изменение файла и создание коммита

Получим список произведенных изменений: (рис. 2.5)

git log

```
[aaastrakhantseva@vbox hello]$ git log
commit fec284002dbe94e7d9b9763e6d8ae21f3166d935 (HEAD -> master)
Author: aaastrakhantseva <nastyamolot04@gmail.com>
Date:   Tue Feb 18 12:37:16 2025 +0300

    Added HTML header

commit db744faf629f458374c78f9b3a7b9437292a6190
Author: aaastrakhantseva <nastyamolot04@gmail.com>
Date:   Tue Feb 18 12:36:00 2025 +0300

    Added standard HTML page tags

commit 0edd1db19da4e72d60ea9a90f35eb5a79e12881d
Author: aaastrakhantseva <nastyamolot04@gmail.com>
Date:   Tue Feb 18 12:31:52 2025 +0300

    Added h1 tag

commit 21b8ca6397950fe52ebee60ff270a8a2bc8f618
Author: aaastrakhantseva <nastyamolot04@gmail.com>
Date:   Tue Feb 18 12:26:59 2025 +0300

    Initial Commit
[aaastrakhantseva@vbox hello]$
```

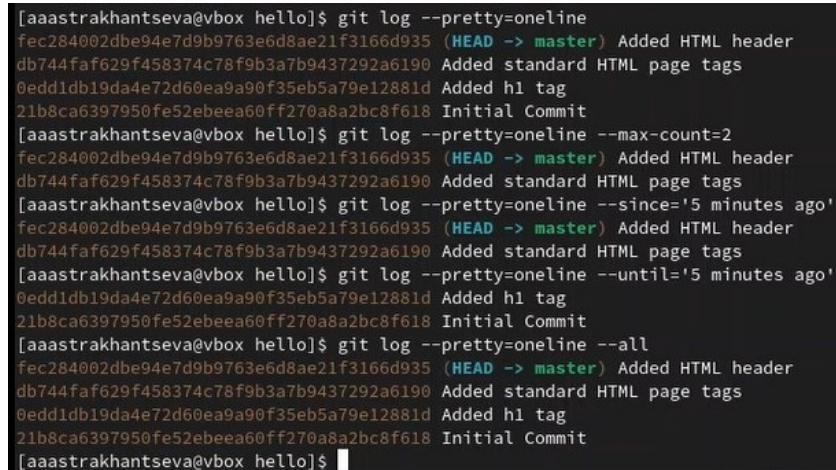
Рис. 2.5: Просмотр истории

Еще различные методы для вывода истории в более удобном виде: (рис. 2.6).

```
git log --pretty=oneline
git log --pretty=oneline --max-count=2
git log --pretty=oneline --since='5 minutes ago'
git log --pretty=oneline --until='5 minutes ago'
git log --pretty=oneline --author=<your name>
git log --pretty=oneline --all
```

Справочную страницу можно посмотреть с помощью:

```
man git-log
```



```
[aaastrakhantseva@vbox hello]$ git log --pretty=oneline
fec284002dbe94e7d9b9763e6d8ae21f3166d935 (HEAD -> master) Added HTML header
db744faf629f458374c78f9b3a7b9437292a6190 Added standard HTML page tags
0edd1db19da4e72d60ea9a90f35eb5a79e12881d Added h1 tag
21b8ca6397950fe52ebee60ff270a8a2bc8f618 Initial Commit
[aaastrakhantseva@vbox hello]$ git log --pretty=oneline --max-count=2
fec284002dbe94e7d9b9763e6d8ae21f3166d935 (HEAD -> master) Added HTML header
db744faf629f458374c78f9b3a7b9437292a6190 Added standard HTML page tags
[aaastrakhantseva@vbox hello]$ git log --pretty=oneline --since='5 minutes ago'
fec284002dbe94e7d9b9763e6d8ae21f3166d935 (HEAD -> master) Added HTML header
db744faf629f458374c78f9b3a7b9437292a6190 Added standard HTML page tags
[aaastrakhantseva@vbox hello]$ git log --pretty=oneline --until='5 minutes ago'
0edd1db19da4e72d60ea9a90f35eb5a79e12881d Added h1 tag
21b8ca6397950fe52ebee60ff270a8a2bc8f618 Initial Commit
[aaastrakhantseva@vbox hello]$ git log --pretty=oneline --all
fec284002dbe94e7d9b9763e6d8ae21f3166d935 (HEAD -> master) Added HTML header
db744faf629f458374c78f9b3a7b9437292a6190 Added standard HTML page tags
0edd1db19da4e72d60ea9a90f35eb5a79e12881d Added h1 tag
21b8ca6397950fe52ebee60ff270a8a2bc8f618 Initial Commit
[aaastrakhantseva@vbox hello]$
```

Рис. 2.6: Варианты просмотра лога

Гит предоставляем возможность откатить проект к старой версии. 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. Он должен быть в последней строке данных. Используем этот хэш-код (достаточно первых 7 знаков) в команде ниже. Затем проверим содержимое файла `hello.html`: (рис. 2.7).

```
git checkout <hash>
```

```
cat hello.html
```

```
Date: Tue Feb 18 12:31:52 2025 +0300

Added h1 tag

commit 21b8ca6397950fe52ebee60ff270a8a2bc8f618
Author: aaastrakhantseva <nastyamolot04@gmail.com>
Date: Tue Feb 18 12:26:59 2025 +0300

Initial Commit
[aaastrakhantseva@vbox hello]$ git checkout 21b8ca6397950fe
Примечание: переключение на «21b8ca6397950fe».

Вы сейчас в состоянии «отсоединённого указателя HEAD». Можете осмотреться,
внести экспериментальные изменения и зафиксировать их, также можете
отменить любые коммиты, созданные в этом состоянии, не затрагивая другие
ветки, переключившись обратно на любую ветку.

Если хотите создать новую ветку для сохранения созданных коммитов, можете
сделать это (сейчас или позже), используя команду switch с параметром -c.
Например:

git switch -c <новая-ветка>

Или отмените эту операцию с помощью:

git switch -

Отключите этот совет, установив переменную конфигурации
advice.detachedHead в значение false

HEAD сейчас на 21b8ca6 Initial Commit
[aaastrakhantseva@vbox hello]$ cat hello.html
Hello, World!
[aaastrakhantseva@vbox hello]$
```

Рис. 2.7: Возвращение к версии репозитория используя хэш

Давайте назовем текущую версию страницы hello первой (v1). Создадим тег первой версии

```
git tag v1
```

Теперь текущая версия страницы называется v1. Давайте создадим тег для версии, которая идет перед текущей версией и назовем его v1-beta. В первую очередь нам надо переключиться на предыдущую версию. Вместо поиска до хэш, мы будем использовать ^, обозначающее «родитель v1». Вместо обозначения v1^ можно использовать v1~1. Это обозначение можно определить как «первую версию предшествующую v1».

```
git checkout v1^  
cat hello.html
```

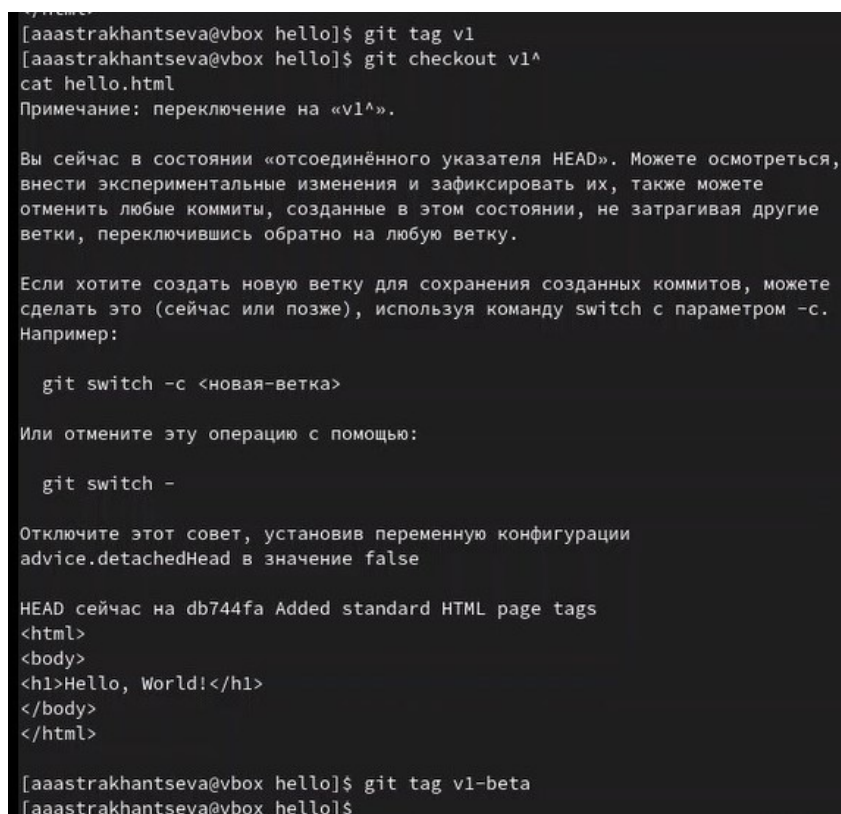
Это версия с тегами

и

, но еще пока без

. Давайте сделаем ее версией v1-beta (рис. 2.8):

```
git tag v1-beta
```



```
[aaastrakhantseva@vbox hello]$ git tag v1  
[aaastrakhantseva@vbox hello]$ git checkout v1^  
cat hello.html  
Примечание: переключение на «v1^».  
  
Вы сейчас в состоянии «отсоединённого указателя HEAD». Можете осмотреться,  
внести экспериментальные изменения и зафиксировать их, также можете  
отменить любые коммиты, созданные в этом состоянии, не затрагивая другие  
ветки, переключившись обратно на любую ветку.  
  
Если хотите создать новую ветку для сохранения созданных коммитов, можете  
сделать это (сейчас или позже), используя команду switch с параметром -с.  
Например:  
  
    git switch -с <новая-ветка>  
  
Или отмените эту операцию с помощью:  
  
    git switch -  
  
Отключите этот совет, установив переменную конфигурации  
advice.detachedHead в значение false  
  
HEAD сейчас на db744fa Added standard HTML page tags  
<html>  
<body>  
<h1>Hello, World!</h1>  
</body>  
</html>  
  
[aaastrakhantseva@vbox hello]$ git tag v1-beta  
[aaastrakhantseva@vbox hello]$
```

Рис. 2.8: Создание тэгов и переключение по ним

Теперь попробуем попереключаться между двумя отмеченными версиями.

```
git checkout v1  
git checkout v1-beta
```

Можем увидеть, какие теги доступны, используя команду (рис. 2.9).:

git tag

```
[aaastrakhantseva@vbox hello]$ git tag v1-beta
[aaastrakhantseva@vbox hello]$ git checkout v1
git checkout v1-beta
Предыдущая позиция HEAD была db744fa Added standard HTML page tags
HEAD сейчас на fec2840 Added HTML header
Предыдущая позиция HEAD была fec2840 Added HTML header
HEAD сейчас на db744fa Added standard HTML page tags
[aaastrakhantseva@vbox hello]$ git tag
v1
v1-beta
[aaastrakhantseva@vbox hello]$
```

Рис. 2.9: Переключение по тэгам

Можем посмотреть теги в логе (рис. 2.10).:

git log master --all

Можем видеть теги (v1 и v1-beta) в логе вместе с именем ветки (master). Кроме того HEAD показывает коммит, на который мы переключились (на данный момент это v1-beta).

```
[aaastrakhantseva@vbox hello]$ git log master --all
commit fec284002dbe94e7d9b9763e6d8ae21f3166d935 (tag: v1, master)
Author: aaastrakhantseva <nastyamolot04@gmail.com>
Date: Tue Feb 18 12:37:16 2025 +0300

    Added HTML header

commit db744faf629f458374c78f9b3a7b9437292a6190 (HEAD, tag: v1-beta)
Author: aaastrakhantseva <nastyamolot04@gmail.com>
Date: Tue Feb 18 12:36:00 2025 +0300

    Added standard HTML page tags

commit 0edd1db19da4e72d60ea9a90f35eb5a79e12881d
Author: aaastrakhantseva <nastyamolot04@gmail.com>
Date: Tue Feb 18 12:31:52 2025 +0300

    Added h1 tag

commit 21b8ca6397950fe52ebaea60ff270a8a2bc8f618
Author: aaastrakhantseva <nastyamolot04@gmail.com>
Date: Tue Feb 18 12:26:59 2025 +0300

    Initial Commit
[aaastrakhantseva@vbox hello]$
```

Рис. 2.10: Имена тэгов в логе

## 5. Отмена локальных изменений (до индексации)

Убедимся, что вы находимся на последнем коммите ветки master, прежде чем продолжить работу.

```
git checkout master
```

## 6. Отмена проиндексированных изменений (перед коммитом)

Внесем изменение в файл hello.html в виде нежелательного комментария:

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

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
```

```
cat hello.html
```

Команда git status показывает нам, что не было произведено никаких изменений, не зафиксированных в рабочем каталоге.

Внесем изменение в файл hello.html в виде нежелательного комментария

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

Проиндексируем это изменение и проверим состояние:

```
git add hello.html
git status
```

Состояние показывает, что изменение было проиндексировано и готово к коммиту. К счастью, вывод состояния показывает нам именно то, что мы должны сделать для отмены индексации изменения:

```
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 нежелательный комментарий. Мы можем использовать команду `git checkout`, чтобы удалить нежелательные изменения в рабочем каталоге (рис. 2.11).



```

[aaastrakhantseva@vbox hello]$ gedit hello.html
[aaastrakhantseva@vbox hello]$ git status
Текущая ветка: master
Изменения, которые не в индексе для коммита:
  (используйте «git add <файл>...», чтобы добавить файл в индекс)
  (используйте «git restore <файл>...», чтобы отменить изменения в рабочем каталоге)
    изменено:      hello.html

индекс пуст (используйте «git add» и/или «git commit -a»)
[aaastrakhantseva@vbox hello]$ git checkout hello.html
git status
cat hello.html
Updated 1 path from the index
Текущая ветка: master
нечего коммитить, нет изменений в рабочем каталоге
<html>
<head>
</head>
<body>
<h1>Hello, World!</h1>
</body>
</html>
[aaastrakhantseva@vbox hello]$ gedit hello.html
[aaastrakhantseva@vbox hello]$ git add hello.html
[aaastrakhantseva@vbox hello]$ git status
Текущая ветка: master
Изменения, которые будут включены в коммит:
  (используйте «git restore --staged <файл>...», чтобы убрать из индекса)
    изменено:      hello.html

[aaastrakhantseva@vbox hello]$ git reset HEAD hello.html
Неприндексированные изменения после сброса:
M       hello.html
[aaastrakhantseva@vbox hello]$

```

Рис. 2.11: Отмена индексации файла

Переключимся на версию коммита:

```

git checkout hello.html
git status

```

Наш рабочий каталог опять чист.

## 7. Отмена коммитов

Иногда нужно отменить коммит, мы сделаем это путем создания нового коммита, отменяющего нежелательные изменения. Изменим файл `hello.html` на следующий:

```

<html>
<head>
</head>
<body>

```



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

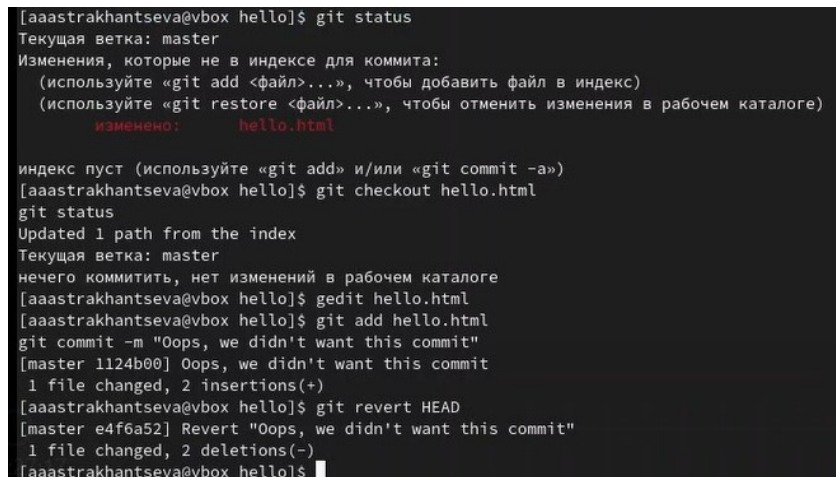
Выполним:

```
git add hello.html
git commit -m "Oops, we didn't want this commit"
```

Чтобы отменить коммит, нам необходимо сделать коммит, который удаляет изменения, сохраненные нежелательным коммитом.

```
git revert HEAD
```

Перейдем в редактор, где мы можем отредактировать коммит-сообщение по умолчанию или оставить все как есть. Сохраним и закроем файл. Так как мы отменили самый последний произведенный коммит, мы смогли использовать HEAD в качестве аргумента для отмены. Мы можем отменить любой произвольной коммит в истории, указав его хэш-значение (рис. 2.12).



```
[aaastrakhantseva@vbox hello]$ git status
Текущая ветка: master
Изменения, которые не в индексе для коммита:
  (используйте «git add <файл>...», чтобы добавить файл в индекс)
  (используйте «git restore <файл>...», чтобы отменить изменения в рабочем каталоге)
изменено:      hello.html

индекс пуст (используйте «git add» и/или «git commit -a»)
[aaastrakhantseva@vbox hello]$ git checkout hello.html
git status
Updated 1 path from the index
Текущая ветка: master
нечего коммитить, нет изменений в рабочем каталоге
[aaastrakhantseva@vbox hello]$ gedit hello.html
[aaastrakhantseva@vbox hello]$ git add hello.html
git commit -m "Oops, we didn't want this commit"
[master 1124b00] Oops, we didn't want this commit
 1 file changed, 2 insertions(+)
[aaastrakhantseva@vbox hello]$ git revert HEAD
[master e4f6a52] Revert "Oops, we didn't want this commit"
 1 file changed, 2 deletions(-)
[aaastrakhantseva@vbox hello]$
```

Рис. 2.12: Отмена коммитов

## 8. Удаление коммитов из ветки

Давайте сделаем быструю проверку нашей истории коммитов. Выполним:

```
git log
```

Мы видим, что два последних коммита в этой ветке — «Oops» и «Revert Oops». Давайте удалим их с помощью сброса.

Но прежде чем удалить коммиты, давайте отметим последний коммит тегом, чтобы потом можно было его найти.

```
git tag oops
```

Глядя на историю лога, мы 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  
git log
```

Наша ветка master теперь указывает на коммит v1, а коммитов Oops и Revert Oops в ветке уже нет. Параметр `--hard` указывает, что рабочий каталог должен быть обновлен в соответствии с новым head ветки (рис. 2.13).

```

[aastrakhantseva@vbox hello]$ git tag oops
[aastrakhantseva@vbox hello]$ git reset --hard v1
git log
Указатель HEAD сейчас на коммите fec2840 Added HTML header
commit fec284002dbe94e7d9b9763e6d8ae21f3166d935 (HEAD -> master, tag: v1)
Author: aastrakhantseva <nastyamolot04@gmail.com>
Date: Tue Feb 18 12:37:16 2025 +0300

    Added HTML header

commit db744faf629f458374c78f9b3a7b9437292a6190 (tag: v1-beta)
Author: aastrakhantseva <nastyamolot04@gmail.com>
Date: Tue Feb 18 12:36:00 2025 +0300

    Added standard HTML page tags

commit 0edd1db19da4e72d60ea9a90f35eb5a79e12881d
Author: aastrakhantseva <nastyamolot04@gmail.com>
Date: Tue Feb 18 12:31:52 2025 +0300

    Added h1 tag

commit 21b8ca6397950fe52ebaea60ff270a8a2bc8f618
Author: aastrakhantseva <nastyamolot04@gmail.com>
Date: Tue Feb 18 12:26:59 2025 +0300

    Initial Commit
[aastrakhantseva@vbox hello]$ █

```

Рис. 2.13: Удаление коммитов

## 9. Удаление тега

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

```

git tag -d oops
git log --all

```

Тег «oops» больше не будет отображаться в репозитории (рис. 2.14).

```
Initial Commit
[aaastrakhantseva@vbox hello]$ git tag -d oops
git log --all
Метка «oops» удалена (была e4f6a52)
commit fec284002dbe94e7d9b9763e6d8ae21f3166d935 (HEAD -> master, tag: v1)
Author: aaastrakhantseva <nastyamolot04@gmail.com>
Date: Tue Feb 18 12:37:16 2025 +0300

    Added HTML header

commit db744faf629f458374c78f9b3a7b9437292a6190 (tag: v1-beta)
Author: aaastrakhantseva <nastyamolot04@gmail.com>
Date: Tue Feb 18 12:36:00 2025 +0300

    Added standard HTML page tags

commit 0edd1db19da4e72d60ea9a90f35eb5a79e12881d
Author: aaastrakhantseva <nastyamolot04@gmail.com>
Date: Tue Feb 18 12:31:52 2025 +0300

    Added h1 tag

commit 21b8ca6397950fe52ebaea60ff270a8a2bc8f618
Author: aaastrakhantseva <nastyamolot04@gmail.com>
Date: Tue Feb 18 12:26:59 2025 +0300

    Initial Commit
[aaastrakhantseva@vbox hello]$
```

Рис. 2.14: Удаление тэга

## 10. Внесение изменений в коммиты

Добавим в страницу комментарий автора:

```
<!-- Author: Anastasiia A. Astrakhantseva -->
<html>
<head>
</head>
<body>
<h1>Hello, World!</h1>
</body>
</html>
```

Выполним:

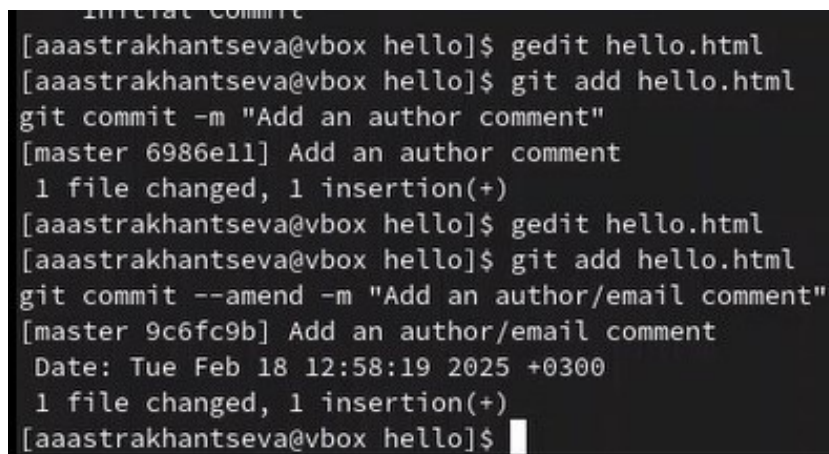
```
git add hello.html
git commit -m "Add an author comment"
```

После совершения коммита мы понимаем, что любой хороший комментарий должен включать электронную почту автора. Обновим страницу hello, включив в нее email.

```
<!-- Author: Anastasiia A. Astrakhantseva (1132226437@pfur.ru) -->
<html>
<head>
</head>
<body>
<h1>Hello, World!</h1>
</body>
</html>
```

Мы действительно не хотим создавать отдельный коммит только ради электронной почты. Давайте изменим предыдущий коммит, включив в него адрес электронной почты. Выполним (рис. 2.15).

```
git add hello.html
git commit --amend -m "Add an author/email comment"
```



```
Integrate commit
[aaastrakhantseva@vbox hello]$ gedit hello.html
[aaastrakhantseva@vbox hello]$ git add hello.html
git commit -m "Add an author comment"
[master 6986e11] Add an author comment
1 file changed, 1 insertion(+)
[aaastrakhantseva@vbox hello]$ gedit hello.html
[aaastrakhantseva@vbox hello]$ git add hello.html
git commit --amend -m "Add an author/email comment"
[master 9c6fc9b] Add an author/email comment
Date: Tue Feb 18 12:58:19 2025 +0300
1 file changed, 1 insertion(+)
[aaastrakhantseva@vbox hello]$
```

Рис. 2.15: Изменение коммита

## 11. Перемещение файлов

Сейчас мы собираемся создать структуру нашего репозитория. Давайте перенесем страницу в каталог lib.

```
mkdir lib
git mv hello.html lib
git status
```

Перемещая файлы с помощью `git mv`, мы информируем `git` о 2 вещах: - Что файл `hello.html` был удален. - Что файл `lib/hello.html` был создан. - Оба эти факта сразу же проиндексированы и готовы к коммиту. Команда `gitstatus` сообщает, что файл был перемещен.

Давайте сделаем коммит этого перемещения:

```
git commit -m "Moved hello.html to lib"
```

Добавим файл `index.html` в наш репозиторий

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

Добавим файл и сделаем коммит (рис. 2.16):

```
git add index.html
git commit -m "Added index.html."
```

```
[aaastrakhantseva@vbox hello]$ mkdir lib
git mv hello.html lib
git status
Текущая ветка: master
Изменения, которые будут включены в коммит:
  (используйте «git restore --staged <файл>...», чтобы убрать из индекса)
    переименовано: hello.html -> lib/hello.html

[aaastrakhantseva@vbox hello]$ git commit -m "Moved hello.html to lib"
[master f111c82] Moved hello.html to lib
 1 file changed, 0 insertions(+), 0 deletions(-)
 rename hello.html => lib/hello.html (100%)
[aaastrakhantseva@vbox hello]$ gedit index.html
[aaastrakhantseva@vbox hello]$ git add index.html
git commit -m "Added index.html."
[master 564107c] Added index.html.
 1 file changed, 6 insertions(+)
 create mode 100644 index.html
[aaastrakhantseva@vbox hello]$
```

Рис. 2.16: Перемещение файла средствами git

## 12. Git внутри: Каталог .git

Выполним:

```
ls -C .git
```

Это каталог, в котором хранится вся информация git. Выполним:

```
ls -C .git/objects
```

Мы должны увидеть 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/c6
```

Смотрим в один из каталогов с именем из 2 букв. Увидим файлы с именами из 38 символов. Это файлы, содержащие объекты, хранящиеся в git. Они сжаты и закодированы, поэтому просмотр их содержимого нам мало чем поможет. 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 (рис. 2.17):

```
ls .git/refs
```

```
[aaastrakhantseva@vbox hello]$ ls -C .git
branches  COMMIT_EDITMSG  config  description  HEAD  hooks  index  info  logs  objects  ORIG_HEAD  packed-refs  refs
[aaastrakhantseva@vbox hello]$ ls -C .git/objects
09 10 11 20 52 5e 69 8a b2 bf c6 e4 f7 fe  pack
0e 11 2d 4f 56 62 72 9c b3 c3 d0 f1 f9  info
[aaastrakhantseva@vbox hello]$ ls -C .git/objects
09 10 11 20 52 5e 69 8a b2 bf c6 e4 f7 fe  pack
0e 11 2d 4f 56 62 72 9c b3 c3 d0 f1 f9  info
[aaastrakhantseva@vbox hello]$ ls -C .git/objects/<dir>
bash: синтаксическая ошибка рядом с неожиданным маркером «newline»
[aaastrakhantseva@vbox hello]$ ls -C .git/objects/c6
b5be25f013151c362dec53ac3fba26c05f12a
[aaastrakhantseva@vbox hello]$ ls -laC .git/objects/c6
- - - b5be25f013151c362dec53ac3fba26c05f12a
[aaastrakhantseva@vbox hello]$ cat .git/config
[core]
    repositoryformatversion = 0
    filemode = true
    bare = false
    logallrefupdates = true
[aaastrakhantseva@vbox hello]$ ls .git/refs
ls .git/refs/heads
ls .git/refs/tags
cat .git/refs/tags/v1
master
v1 v1-beta
fec284002dbe94e7d9b9763e6d8ae21f3166d935
[aaastrakhantseva@vbox hello]$
```

Рис. 2.17: Структура git

Выполним:

```
cat .git/HEAD
```

Файл HEAD содержит ссылку на текущую ветку, в данный момент это должна быть ветка master (рис. 2.18):

Выполним:

```
git log --max-count=1
```

### 13. Работа непосредственно с объектами git

Эта команда должна показать последний коммит в репозиторий:



```
[aaastrakhantseva@vbox hello]$ cat .git/HEAD
ref: refs/heads/master
[aaastrakhantseva@vbox hello]$
git log --max-count=1[aaastrakhantseva@vbox hello]$ git log --max-count=1
commit 564107c4ec0d5a8a8b6edf773b3090d2b4a10635 (HEAD -> master)
Author: aaastrakhantseva <nastyamolot04@gmail.com>
Date: Tue Feb 18 13:03:07 2025 +0300

    Added index.html.
[aaastrakhantseva@vbox hello]$
```

Рис. 2.18: Содержание git/HEAD

Выполним:

```
git cat-file -t <hash>
```

```
git cat-file -p <hash>
```

Мы можем вывести дерево каталогов, ссылка на который идет в коммите. Это должно быть описание файлов (верхнего уровня) в нашем проекте (для конкретного коммита). Используйте SHA1 хэш из строки «дерева», из списка выше.

Выполним:

```
git cat-file -p <treehash>
```

Выполним:

```
git cat-file -p <libhash>
```

1.15.5 Вывод файла hello.html Выполним:

```
git cat-file -p <hellohash>
```

Исследуем git репозиторий вручную самостоятельно. Смотрите, удастся ли вам найти оригинальный файл hello.html с самого первого коммита вручную по ссылкам SHA1 хэша в последнем коммите (рис. ??, ??):

```

git log --max-count=1[aastrakhantseva@vbox hello]$ git log --max-count=1
commit 564107c4ec0d5a8a8b6edf773b3090d2b4a10635 (HEAD -> master)
Author: aastrakhantseva <nastyamolot04@gmail.com>
Date: Tue Feb 18 13:03:07 2025 +0300

    Added index.html.
[aastrakhantseva@vbox hello]$ git cat-file -t 564107c4ec0d
commit
[aastrakhantseva@vbox hello]$ git cat-file -p 564107c4ec0d
tree b3fa35711ee4d257f866a55a4d2991c5376fd71b
parent fillc8288669c5892b84851a38ed8c7341afd93f
author aastrakhantseva <nastyamolot04@gmail.com> 1739872987 +0300
committer aastrakhantseva <nastyamolot04@gmail.com> 1739872987 +0300

Added index.html.
[aastrakhantseva@vbox hello]$ git cat-file -p b3fa35711ee4d257f866a55a4d2991c5376fd71b
100644 blob 723e799ebff9306dbfda097463c709db4f8cd367    index.html
040000 tree 105ece18ef1ac46bbaa848ab52836efcfe3e9db    lib
[aastrakhantseva@vbox hello]$ git cat-file -p lib
fatal: Not a valid object name lib
[aastrakhantseva@vbox hello]$ git cat-file -p 040000
fatal: Not a valid object name 040000
[aastrakhantseva@vbox hello]$ git cat-file -p 723e799ebf
<html>
<body>
<iframe src="lib/hello.html" width="200" height="200" />
</body>
</html>

```

```

Initial Commit
[aastrakhantseva@vbox hello]$ git cat-file -t 21b8ca63
commit
[aastrakhantseva@vbox hello]$ git cat-file -p 21b8ca63
tree 523cd318fc3b74ad3852cd34a74720ad6f7dd422
author aastrakhantseva <nastyamolot04@gmail.com> 1739872987 +0300
committer aastrakhantseva <nastyamolot04@gmail.com> 1739872987 +0300

Initial Commit
[aastrakhantseva@vbox hello]$ git cat-file -p 23cd318fc3b74ad3852cd34a74720ad6f7dd422
fatal: Not a valid object name 23cd318fc3b74ad3852cd34a74720ad6f7dd422
[aastrakhantseva@vbox hello]$ git cat-file -p 523cd318fc3b74ad3852cd34a74720ad6f7dd422
100644 blob 8ab686eafeb1f44702738c8b0f24f2567c36da6d
[aastrakhantseva@vbox hello]$ git cat-file -p 8ab686eafeb1f44702738c8b0f24f2567c36da6d
Hello, World!
[aastrakhantseva@vbox hello]$ git checkout -b style
git status
Переключились на новую ветку «style»
Текущая ветка: style
нечего коммитить, нет изменений в рабочем каталоге
[aastrakhantseva@vbox hello]$ gedit style.css

```

## 14. Создание ветки

Пора сделать наш hello world более выразительным. Так как это может занять некоторое время, лучше переместить эти изменения в отдельную ветку, чтобы изолировать их от изменений в ветке master. Давайте назовем нашу новую ветку «style». Выполним:

```
git checkout -b style
```

```
git status
```

```
git checkout -b <имя_ветки> является шорткатом для git branch
```

Обратим внимание, что команда git status сообщает о том, что вы находитесь в ветке «style». Добавим файл стилей style.css

```

h1 {
color: red;
}

```

Выполним:

```
git add lib/style.css
git commit -m "Added css stylesheet"
```

Выполним:

Обновим файл hello.html, чтобы использовать стили style.css.

```
<!-- Author: Anastasiia A. Astrakhantseva (1132226437@pfur.ru) -->
<html>
<head>
<link type="text/css" rel="stylesheet"
media="all" href="style.css" />
18
</head>
<body>
<h1>Hello, World!</h1>
</body>
</html>
```

Выполним:

```
git add lib/hello.html
git commit -m "Hello uses style.css"
```

Обновим файл index.html, чтобы он тоже использовал style.css

```
<html>
<head>
<link type="text/css" rel="stylesheet"
media="all" href="lib/style.css" />
</head>
<body>
<iframe src="lib/hello.html" width="200" height="200" />
```

```
</body>
</html>
```

Выполним (рис. 2.19):

```
git add index.html
git commit -m "Updated index.html"
```

```
[aaastrakhantseva@vbox hello]$ gedit style.css
[aaastrakhantseva@vbox hello]$ ls
index.html  lib  style.css
[aaastrakhantseva@vbox hello]$ mv style.css lib
[aaastrakhantseva@vbox hello]$ ls
index.html  lib
[aaastrakhantseva@vbox hello]$ ls lib
hello.html  style.css
[aaastrakhantseva@vbox hello]$ git add lib/style.css
git commit -m "Added css stylesheet"
[style e20612c] Added css stylesheet
1 file changed, 4 insertions(+)
create mode 100644 lib/style.css
[aaastrakhantseva@vbox hello]$ ls
index.html  lib
[aaastrakhantseva@vbox hello]$ ls lib
hello.html  style.css
[aaastrakhantseva@vbox hello]$ gedit lib/hello.html
[aaastrakhantseva@vbox hello]$ git add lib/hello.html
git commit -m "Hello uses style.css"
[style 9f8feef] Hello uses style.css
1 file changed, 2 insertions(+), 1 deletion(-)
[aaastrakhantseva@vbox hello]$ gedit
[aaastrakhantseva@vbox hello]$ gedit index.html
[aaastrakhantseva@vbox hello]$ git add index.html
git commit -m "Updated index.html"
[style 168b698] Updated index.html
1 file changed, 3 insertions(+)
```

Рис. 2.19: Создание ветки, редактирование файлов

## 15. Навигация по веткам

Теперь в проекте есть две ветки (рис. 2.20) Выполним:

```
git log --all
```

```
[aaastrakhantseva@vbox hello]$ git log --all
commit 168b698c6eeb6ed56cdbfe05c7a442008c0a1028 (HEAD -> style)
Author: aaastrakhantseva <nastyamolot04@gmail.com>
Date: Tue Feb 18 13:19:23 2025 +0300

    Updated index.html

commit 9f8feef4530d1156382af7a9cb2729b42e6dccdf
Author: aaastrakhantseva <nastyamolot04@gmail.com>
Date: Tue Feb 18 13:18:36 2025 +0300

    Hello uses style.css

commit e20612cf4cfe60851acce5a02fbefcc5d02473ca
Author: aaastrakhantseva <nastyamolot04@gmail.com>
Date: Tue Feb 18 13:17:46 2025 +0300

    Added css stylesheet

commit 564107c4ec0d5a8a8b6edf773b3090d2b4a10635 (master)
Author: aaastrakhantseva <nastyamolot04@gmail.com>
Date: Tue Feb 18 13:03:07 2025 +0300

    Added index.html.

commit f111c8288669c5892b84851a38ed8c7341afd93f
Author: aaastrakhantseva <nastyamolot04@gmail.com>
Date: Tue Feb 18 13:02:22 2025 +0300

    Moved hello.html to lib

commit 9c6fc9b11b37a79e5fa10d272790798818affe98
Author: aaastrakhantseva <nastyamolot04@gmail.com>
Date: Tue Feb 18 12:58:19 2025 +0300

    Add an author/email comment

commit fec284002dbe94e7d9b9763e6d8ae21f3166d935 (tag: v1)
Author: aaastrakhantseva <nastyamolot04@gmail.com>
Date: Tue Feb 18 12:37:16 2025 +0300
```

Рис. 2.20: Просмотр лога

Используем команду `git checkout` для переключения между ветками:

```
git checkout master
cat lib/hello.html
```

Сейчас мы находимся на ветке `master`. Это заметно по тому, что файл `hello.html` не использует стили `style.css`. Вернемся к ветке `style`. Выполним:

```
git checkout style
cat lib/hello.html
```

Содержимое lib/hello.html подтверждает, что мы вернулись на ветку style.

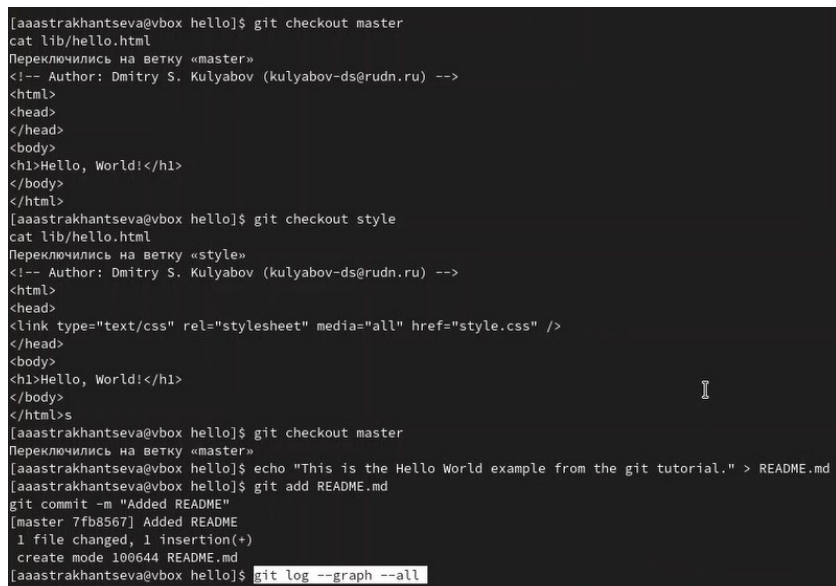
Пока вы меняли ветку style, кто-то решил обновить ветку master. Они добавили файл README.md.

Выполним:

```
git checkout master
echo "This is the Hello World example from the git tutorial." > README.md
```

Сделаем коммит изменений README.md в ветку master (рис. 2.21).

```
git add README.md
git commit -m "Added README"
```



```
[aaastrakhantseva@vbox hello]$ git checkout master
cat lib/hello.html
Переключились на ветку «master»
<!-- Author: Dmitry S. Kulyabov (kulyabov-ds@rudn.ru) -->
<html>
<head>
</head>
<body>
<h1>Hello, World!</h1>
</body>
</html>
[aaastrakhantseva@vbox hello]$ git checkout style
cat lib/hello.html
Переключились на ветку «style»
<!-- Author: Dmitry S. Kulyabov (kulyabov-ds@rudn.ru) -->
<html>
<head>
<link type="text/css" rel="stylesheet" media="all" href="style.css" />
</head>
<body>
<h1>Hello, World!</h1>
</body>
</html>
[aaastrakhantseva@vbox hello]$ git checkout master
Переключились на ветку «master»
[aaastrakhantseva@vbox hello]$ echo "This is the Hello World example from the git tutorial." > README.md
[aaastrakhantseva@vbox hello]$ git add README.md
git commit -m "Added README"
[master 7fb8567] Added README
1 file changed, 1 insertion(+)
create mode 100644 README.md
[aaastrakhantseva@vbox hello]$ git log --graph --all
```

Рис. 2.21: Создание отличий в ветках

Теперь у нас в репозитории есть две отличающиеся ветки. Используем лог-команду для просмотра веток и их отличий (рис. 2.22).

```
git log --graph --all
```



```
[aaastrakhantseva@vbox hello]$ git log --graph --all
* commit 7fb8567417f732a10742e68b42a0c5ad6f2cbb86 (HEAD -> master)
  Author: aaastrakhantseva <nastyamolot04@gmail.com>
  Date: Tue Feb 18 13:21:22 2025 +0300

    Added README

* commit 168b698c6eeb6ed56cdbfe05c7a442008c0a1028 (style)
  Author: aaastrakhantseva <nastyamolot04@gmail.com>
  Date: Tue Feb 18 13:19:23 2025 +0300

    Updated index.html

* commit 9f8feef4530d1156382af7a9cb2729b42e6dccdf
  Author: aaastrakhantseva <nastyamolot04@gmail.com>
  Date: Tue Feb 18 13:18:36 2025 +0300

    Hello uses style.css

* commit e20612cf4cfe60851acce5a02fbefcc5d02473ca
  Author: aaastrakhantseva <nastyamolot04@gmail.com>
  Date: Tue Feb 18 13:17:46 2025 +0300

    Added css stylesheet

* commit 564107c4ec0d5a8a8b6edf773b3090d2b4a10635
  Author: aaastrakhantseva <nastyamolot04@gmail.com>
  Date: Tue Feb 18 13:03:07 2025 +0300

    Added index.html.

* commit f11lc8288669c5892b84851a38ed8c7341afd93f
  Author: aaastrakhantseva <nastyamolot04@gmail.com>
  Date: Tue Feb 18 13:02:22 2025 +0300

    Moved hello.html to lib

* commit 9c6fc9b11b37a79e5fa10d272790798818affe98
  Author: aaastrakhantseva <nastyamolot04@gmail.com>
  Date: Tue Feb 18 12:58:19 2025 +0300
```

Рис. 2.22: Просмотр графа

## 16. Слияние

Слияние переносит изменения из двух веток в одну. Давайте вернемся к ветке style и сольем master с style (рис. 2.23).

```
git checkout style
git merge master
git log --graph --all
```

```
* commit acd8959b9573c9c45c8589010b6fad76af69ca23 (HEAD -> style)
Author: aaastrakhantseva <nastyamolot04@gmail.com>
Date: Tue Feb 18 13:28:27 2025 +0300

    Merged master fixed conflict.

* commit 777c75b09555d85f3d29127d1cfla0518a37c301
 \ Merge: 0af7381 25a490f
  | Author: aaastrakhantseva <nastyamolot04@gmail.com>
  | Date: Tue Feb 18 13:25:21 2025 +0300
  |
  | Merge branch 'master' into style
  |
* commit 25a490f19f11b5b3514073f9a9a2da5f5a416d95 (master)
 | Author: aaastrakhantseva <nastyamolot04@gmail.com>
 | Date: Tue Feb 18 13:24:29 2025 +0300
 |
 | Life is great
 |
* commit 0af738162dd8f475746dda445e2c42b5342ada3e
 \ Merge: 168b698 7fb8567
  | Author: aaastrakhantseva <nastyamolot04@gmail.com>
  | Date: Tue Feb 18 13:22:42 2025 +0300
  |
  | Merge branch 'master' into style
  |
* commit 7fb8567417f732a10742e68b42a0c5ad6f2cbb86
 | Author: aaastrakhantseva <nastyamolot04@gmail.com>
 | Date: Tue Feb 18 13:21:22 2025 +0300
 |
 | Added README
 |
* commit 168b698c6eeb6ed56cdbfe05c7a442008c0a1028
 | Author: aaastrakhantseva <nastyamolot04@gmail.com>
 | Date: Tue Feb 18 13:19:23 2025 +0300
 |
 | Updated index.html
 |
* commit 9f8feef4530d1156382af7a9cb2729b42e6dcccdf
 | Author: aaastrakhantseva <nastyamolot04@gmail.com>
 | Date: Tue Feb 18 13:18:36 2025 +0300
 |
 | Hello uses style.css
 |
:
```

Рис. 2.23: Просмотр графа

## 17. Сброс ветки

Нам необходимо найти последний коммит перед слиянием (рис. 2.24).

```
git checkout style
```

```
git log --graph
```



Мы видим, что коммит «Updated index.html» был последним на ветке style перед слиянием. Давайте сбросим ветку style к этому коммиту (рис. 2.24).

```
git reset --hard <hash>
```

```
ERROR: желось в виду path (с двумя дефисами):
[aastrakhantseva@vbox hello]$ git reset --hard 7fb8567417f732a10742e68b42a0c5ad6f2cbb86
Указатель HEAD сейчас на коммите 7fb8567 Added README
[aastrakhantseva@vbox hello]$ git log --graph --all
* commit 7fb8567417f732a10742e68b42a0c5ad6f2cbb86 (HEAD -> master)
  Author: aastrakhantseva <nastyamolot04@gmail.com>
  Date: Tue Feb 18 13:21:22 2025 +0300

    Added README

* commit 168b698c6eeb6ed56cdbcfe05c7a442008c0a1028 (style)
  Author: aastrakhantseva <nastyamolot04@gmail.com>
  Date: Tue Feb 18 13:19:23 2025 +0300

    Updated index.html

* commit 9f8feef4530d1156382af7a9cb2729b42e6dcccdf
  Author: aastrakhantseva <nastyamolot04@gmail.com>
  Date: Tue Feb 18 13:18:36 2025 +0300

    Hello uses style.css

* commit e20612cf4cfe60851acce5a02fbefcc5d02473ca
  Author: aastrakhantseva <nastyamolot04@gmail.com>
  Date: Tue Feb 18 13:17:46 2025 +0300

    Added css stylesheet

* commit 564107c4ec0d5a8a8b6edf773b3090d2b4a10635
  Author: aastrakhantseva <nastyamolot04@gmail.com>
  Date: Tue Feb 18 13:03:07 2025 +0300

    Added index.html.

* commit f111c828366945892b84851a38e08c7341afd93f
  Author: aastrakhantseva <nastyamolot04@gmail.com>
  Date: Tue Feb 18 13:02:22 2025 +0300

    Moved hello.html to lib

* commit 9c6fc9b11b37a79e5fa10d272790798818affe98
  Author: aastrakhantseva <nastyamolot04@gmail.com>
  Date: Tue Feb 18 12:58:19 2025 +0300
```

Рис. 2.24: Сброс ветки

## 17. 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. На этот раз для переноса изменений из ветки master мы будем использовать команду git rebase вместо слияния (рис. 2.25).

```
git checkout style
```

```
git rebase master
```

```
git log --graph
```

```
[aaastrakhantseva@vbox hello]$ git checkout style
git rebase master
git log --graph
Переключились на ветку «style»
Успешно перемещён и обновлён refs/heads/style.
* commit 4c71a73eba1ccc12ddf3145dba3face7988a08c9 (HEAD -> style)
  Author: aaastrakhantseva <nastyamolot04@gmail.com>
  Date:   Tue Feb 18 13:19:23 2025 +0300

    Updated index.html

* commit 462e8225a8103fd9fbb48aaecd2271d269496a76
  Author: aaastrakhantseva <nastyamolot04@gmail.com>
  Date:   Tue Feb 18 13:18:36 2025 +0300

    Hello uses style.css

* commit db77e239050e47dbe3aaa9beabd2a0822e98cabd
  Author: aaastrakhantseva <nastyamolot04@gmail.com>
  Date:   Tue Feb 18 13:17:46 2025 +0300

    Added css stylesheet

* commit 7fb856741f732a10742e68b42a0c5ad6f2cbb86 (master)
  Author: aaastrakhantseva <nastyamolot04@gmail.com>
  Date:   Tue Feb 18 13:21:22 2025 +0300

    Added README

* commit 564107c4ec0d5a8a8b6edf773b3090d2b4a10635
  Author: aaastrakhantseva <nastyamolot04@gmail.com>
  Date:   Tue Feb 18 13:03:07 2025 +0300

    Added index.html.

* commit f111c8288669c5892b84851a38ed8c7341afd93f
  Author: aaastrakhantseva <nastyamolot04@gmail.com>
  Date:   Tue Feb 18 13:02:22 2025 +0300

    Moved hello.html to lib
```

Рис. 2.25: Перебазирование

## 18. Клонирование репозитория

Перейдем в рабочий каталог и сделаем клон репозитория hello.

```
cd ..
```

```
pwd
```

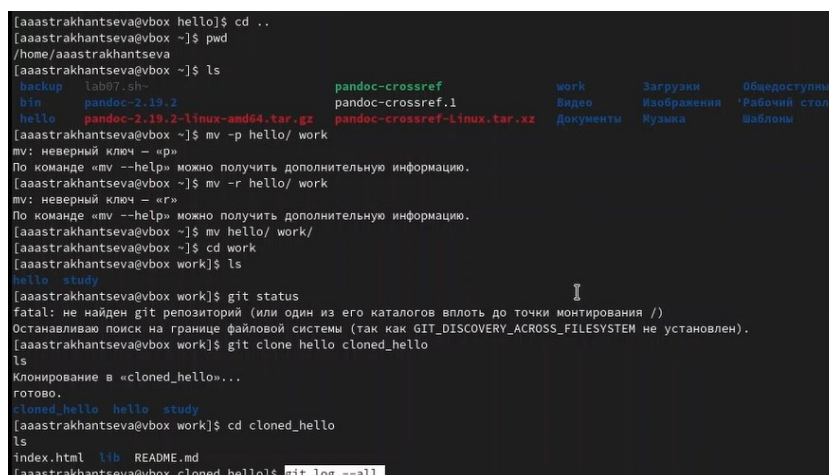
```
ls
```

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.

```
git clone hello cloned_hello  
ls
```

В рабочем каталоге теперь должно быть два репозитория: оригинальный репозиторий «hello» и клонированный репозиторий «cloned\_hello». Давайте взглянем на клонированный репозиторий (рис. 2.26).

```
cd cloned_hello  
ls
```



```
[aaastrakhantseva@vbox hello]$ cd ..  
[aaastrakhantseva@vbox ~]$ pwd  
/home/aaastrakhantseva  
[aaastrakhantseva@vbox ~]$ ls  
backup  lab97.sh  pandoc-crossref  work  Загрузки  Общедоступные  
bin     pandoc-2.19.2  pandoc-crossref.1  Видео  Изображения  'Рабочий стол'  
hello   pandoc-2.19.2-linux-amd64.tar.gz  pandoc-crossref-Linux.tar.xz  Документы  Музыка  Шаблоны  
[aaastrakhantseva@vbox ~]$ mv -p hello/ work  
mv: неверный ключ - «p»  
По команде «mv --help» можно получить дополнительную информацию.  
[aaastrakhantseva@vbox ~]$ mv -r hello/ work  
mv: неверный ключ - «r»  
По команде «mv --help» можно получить дополнительную информацию.  
[aaastrakhantseva@vbox ~]$ mv work/ work/  
[aaastrakhantseva@vbox ~]$ cd work  
[aaastrakhantseva@vbox work]$ ls  
hello  study  
[aaastrakhantseva@vbox work]$ git status  
fatal: не найден git репозиторий (или один из его каталогов вплоть до точки монтирования /).  
Останавливаю поиск на границе файловой системы (так как GIT_DISCOVERY_ACROSS_FILESYSTEM не установлен).  
[aaastrakhantseva@vbox work]$ git clone hello cloned_hello  
Клонирование в «cloned_hello»...  
готово.  
[aaastrakhantseva@vbox work]$ cd cloned_hello  
ls  
index.html  11k  README.md  
[aaastrakhantseva@vbox cloned_hello]$ git log --all
```

Рис. 2.26: Клонирование репозитория

Посмотрим историю репозитория

```
git log --all
```

Увидим список всех коммитов в новый репозиторий, и он должен (более или менее) совпадать с историей коммитов в оригинальном репозитории. Единственная разница должна быть в названиях веток (рис. 2.27).

```
index.html  README.md
[aastrakhantseva@ybox cloned_hello]$ git log --all
commit 4c71a73eb1ccc12ddf3145dba3face7988a08c9 (HEAD -> master, origin/style, origin/master, origin/HEAD)
Author: aastrakhantseva <nastyamolot04@gmail.com>
Date: Tue Feb 18 13:19:23 2025 +0300

    Updated index.html

commit 462e8225a8103fd9fbb48aaecd2271d269496a76
Author: aastrakhantseva <nastyamolot04@gmail.com>
Date: Tue Feb 18 13:18:36 2025 +0300

    Hello uses style.css

commit db77e239050e47dbe3aaa0beabd2a0822e98cabd
Author: aastrakhantseva <nastyamolot04@gmail.com>
Date: Tue Feb 18 13:17:46 2025 +0300

    Added css stylesheet

commit 7fb8567417f732a10742e68b42a0c5ad6f2cbb86
Author: aastrakhantseva <nastyamolot04@gmail.com>
Date: Tue Feb 18 13:21:22 2025 +0300

    Added README

commit 564107c4ec0d5a8a8b6edf773b3090d2b4a10635
Author: aastrakhantseva <nastyamolot04@gmail.com>
Date: Tue Feb 18 13:03:07 2025 +0300

    Added index.html.

commit f111c8288669c5892b84851a38ed8c7341afd93f
Author: aastrakhantseva <nastyamolot04@gmail.com>
Date: Tue Feb 18 13:02:22 2025 +0300

    Moved hello.html to lib
```

Копировать

Копировать как HTML

Вставить

☐ Только для чтения

Параметры

Создать окно

Создать вкладку

☒ Показывать панель меню

Рис. 2.27: История репозитория

## 19. Что такое origin?

Выполним:

```
git remote
```

Мы видим, что клонированный репозиторий знает об имени по умолчанию удаленного репозитория. Давайте посмотрим, можем ли мы получить более подробную информацию об имени по умолчанию:

```
git remote show origin
```

Удаленные репозитории обычно размещаются на отдельной машине, возможно, централизованном сервере. Однако, как мы видим здесь, они могут с тем же успехом указывать на репозиторий на той же машине. Нет ничего особенного в имени «origin», однако существует традиция использовать «origin» в качестве имени первичного централизованного репозитория (если таковой имеется).

Давайте посмотрим на ветки, доступные в нашем клонированном репозитории.

```
git branch
```

Как мы видим, в списке только ветка master. Где ветка style? Команда git branch выводит только список локальных веток по умолчанию.

Для того, чтобы увидеть все ветки, попробуем следующую команду:

```
git branch -a
```

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.

Внесем некоторые изменения в оригинальный репозиторий, чтобы затем попытаться извлечь и слить изменения из удаленной ветки в текущую

```
cd ../hello
```

Внесите следующие изменения в файл README.md (рис. 2.28).

```
[aaastrakhantseva@vbox cloned_hello]$ git remote
origin
[aaastrakhantseva@vbox cloned_hello]$ git remote show origin
* внешний репозиторий origin
  URL для извлечения: /home/aaastrakhantseva/work/hello
  URL для отправки: /home/aaastrakhantseva/work/hello
  HEAD ветка: master
  Внешние ветки:
    master отслеживается
    style отслеживается
  Локальная ветка, настроенная для «git pull»:
    master будет слита с внешней веткой master
  Локальная ссылка, настроенная для «git push»:
    master будет отправлена в master (уже актуальна)
[aaastrakhantseva@vbox cloned_hello]$ git branch
* master
[aaastrakhantseva@vbox cloned_hello]$ git branch -a
* master
  remotes/origin/HEAD -> origin/master
  remotes/origin/master
  remotes/origin/style
[aaastrakhantseva@vbox cloned_hello]$ cd ../hello/
[aaastrakhantseva@vbox hello]$ gedit README.md
```

Рис. 2.28: Origin

This is the Hello World example from the git tutorial.

Теперь добавьте это изменение и сделайте коммит

```
git add README
```

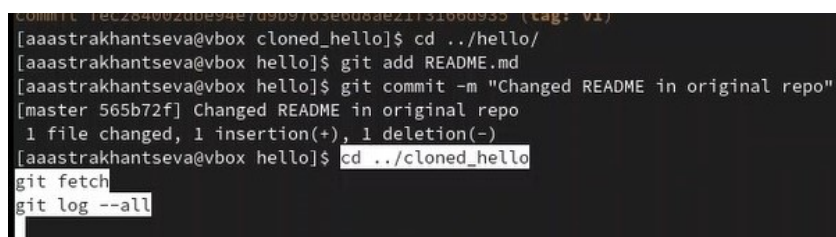
```
git commit -m "Changed README in original repo"
```

Теперь в оригинальном репозитории есть более поздние изменения, которых нет в клонированной версии. Далее мы извлечем и сольем эти изменения в клонированный репозиторий (рис. 2.29).

```
cd ../cloned_hello
```

```
git fetch
```

```
git log --all
```



```
commit fec2840020be94e7d9b9703e008ae21751000933 (tag: v1)
[aaastrakhantseva@vbox cloned_hello]$ cd ../hello/
[aaastrakhantseva@vbox hello]$ git add README.md
[aaastrakhantseva@vbox hello]$ git commit -m "Changed README in original repo"
[master 565b72f] Changed README in original repo
1 file changed, 1 insertion(+), 1 deletion(-)
[aaastrakhantseva@vbox hello]$ cd ../cloned_hello
git fetch
git log --all
```

Рис. 2.29: Извлечение изменений

## 20. Слияние извлеченных изменений

Сольем извлеченные изменения в локальную ветку master (рис. 2.29)

```
git merge origin/master
```

Еще раз проверим файл README.md Сейчас мы должны увидеть изменения.

Теперь давайте рассмотрим объединение fetch и merge в одну команду. Выполним:

```
git pull
```

эквивалентно двум следующим шагам:

```
git fetch
git merge origin/master
```

## 21. Добавление ветки наблюдения

Ветки, которые начинаются с `remotes/origin` являются ветками оригинального репозитория. Обратите внимание, что у вас больше нет ветки под названием `style`, но система контроля версий знает, что в оригинальном репозитории ветка `style` была. 1.33.1 Добавьте 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. 22. Чистые репозитории Чистые репозитории (без рабочих каталогов) обычно используются для расшаривания. Обычный `git`-репозиторий подразумевает, что вы будете использовать его как рабочую директорию, поэтому вместе с файлами проекта в актуальной версии, `git` хранит все служебные, «чисто-репозиторийские» файлы в поддиректории `.git`. В удаленных репозиториях нет смысла хранить рабочие файлы на диске (как это делается в рабочих копиях), а все что им действительно нужно — это дельты изменений и другие бинарные данные репозитория. Вот это и есть «чистый репозиторий». Создадим чистый репозиторий (рис. 2.30).

```
cd ..
git clone --bare hello hello.git
ls hello.git
```



```

[aaastrakhantseva@vbox cloned_hello]$ cat README.md
This is the Hello World example from the git tutorial.
[aaastrakhantseva@vbox cloned_hello]$ git merge origin/master
Обновление 4c71a73..565b72f
Fast-forward
 README.md | 2 +-
 1 file changed, 1 insertion(+), 1 deletion(-)
[aaastrakhantseva@vbox cloned_hello]$ cat README.md
This is the Hello World example from the git tutorial!
[aaastrakhantseva@vbox cloned_hello]$ git pull
Уже актуально.
[aaastrakhantseva@vbox cloned_hello]$ git fetch
git merge origin/master
Уже актуально.
[aaastrakhantseva@vbox cloned_hello]$ git branch --track style origin/style
git branch -a
git log --max-count=2
branch 'style' set up to track 'origin/style'.
* master
  style
  remotes/origin/HEAD -> origin/master
  remotes/origin/master
  remotes/origin/style
commit 565b72f85cab6e23784cd0b63ec0a7d6323b7653 (HEAD -> master, origin/master, origin/HEAD)
Author: aaastrakhantseva <nastyamolot04@gmail.com>
Date: Tue Feb 18 16:07:02 2025 +0300

    Changed README in original repo

commit 4c71a73e8a1ccc12ddf3145dba3face7988a08c9 (origin/style, style)
Author: aaastrakhantseva <nastyamolot04@gmail.com>
Date: Tue Feb 18 13:19:23 2025 +0300

    Updated index.html
[aaastrakhantseva@vbox cloned_hello]$ cd ..
git clone --bare hello hello.git
ls hello.git
Клонирование в голый репозиторий «hello.git»...
готово.
branches config description HEAD hooks info objects packed-refs refs
[aaastrakhantseva@vbox work]$

```

Рис. 2.30: Различные команды git

## 23. Отправка изменений

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. Файл README.md:

This is the Hello World example from the git tutorial.

(Changed in the original and pushed to shared)

Выполним:

```
git checkout master
```

```
git add README
```

```
git commit -m "Added shared comment to readme"
```



Теперь отправьте изменения в общий репозиторий. Выполним:

```
git push shared master
```

Общим называется репозиторий, получающий отправленные нами изменения.

1.38 Извлечение общих изменений Научиться извлекать изменения из общего репозитория. Быстро переключитесь в клонированный репозиторий и извлеките изменения, только что отправленные в общий репозиторий.

```
cd ../cloned_hello
```

Сейчас мы находимся в репозитории cloned\_hello. Выполните:

```
git remote add shared ../hello.git git branch --track shared master  
git pull shared master cat README.md
```

## 3 Выводы

В ходе выполнения лабораторной работы я приобрела навыки работы с git и markdown.