

Лабораторная работа №1

Работа с git

Аскеров Александр Эдуардович

Содержание

1 Цель работы	10
2 Задание	11
3 Теоретическое введение	12
4 Выполнение лабораторной работы	13
4.1 Подготовка	13
4.1.1 Установка имени и электронной почты	13
4.1.2 Параметры установки окончаний строк	13
4.1.3 Установка отображения unicode	13
4.2 Создание проекта	14
4.2.1 Создадим страницу «Hello World»	14
4.2.2 Создание репозитория	14
4.2.3 Добавление файла в репозиторий	14
4.2.4 Проверка состояния репозитория	15
4.3 Внесение изменений	15
4.3.1 Изменим страницу «Hello World»	15
4.4 Индексация изменений	16
4.4.1 Коммит изменений	16
4.4.2 Добавим стандартные теги страницы	17
4.4.3 История	20
4.4.4 Получение старых версий	21
4.4.5 Создание тегов версий	23
4.4.6 Переключение по имени тега	24
4.4.7 Просмотр тегов с помощью команды tag	25
4.5 Отмена локальных изменений (до индексации)	25
4.5.1 Переключимся на ветку master	25
4.5.2 Изменим hello.html	26
4.5.3 Проверим состояние	26
4.5.4 Отмена изменений в рабочем каталоге	27
4.6 Отмена проиндексированных изменений (перед коммитом)	27
4.6.1 Изменим файл и проиндексируем изменения	27
4.6.2 Проверим состояние	28
4.6.3 Выполним сброс буферной зоны	28
4.6.4 Переключимся на версию коммита	29

4.7	Отмена коммитов	29
4.7.1	Отмена коммитов	29
4.7.2	Изменим файл и сделаем коммит	29
4.7.3	Сделаем коммит с новыми изменениями, отменяющими предыдущие	30
4.7.4	Проверим лог	31
4.8	Удаление коммитов из ветки	31
4.8.1	Команда git reset	32
4.8.2	Проверим нашу историю	32
4.8.3	Для начала отметим эту ветку	33
4.8.4	Сброс коммитов к предшествующему коммиту Oops	34
4.8.5	Ничего никогда не теряется	34
4.8.6	Опасность сброса	35
4.9	Удаление тега oops	36
4.9.1	Удаление тега oops	36
4.10	Внесение изменений в коммиты	36
4.10.1	Изменим страницу, а затем сделаем коммит	36
4.10.2	Необходим email	37
4.10.3	Изменим предыдущий коммит	38
4.10.4	Просмотр истории	38
4.11	Перемещение файлов	39
4.11.1	Переместим файл hello.html в каталог lib	39
4.12	Второй способ перемещения файлов	39
4.12.1	Коммит в новый каталог	39
4.13	Подробнее о структуре	40
4.13.1	Добавление index.html	40
4.14	Git внутри: Каталог .git	41
4.14.1	Каталог .git	41
4.14.2	База данных объектов	41
4.14.3	Углубляемся в базу данных объектов	42
4.14.4	Config File	42
4.14.5	Ветки и теги	43
4.14.6	Файл HEAD	43
4.15	Работа непосредственно с объектами git	44
4.15.1	Поиск последнего коммита	44
4.15.2	Вывод последнего коммита с помощью SHA1 хэша	44
4.15.3	Поиск дерева	44
4.15.4	Вывод каталога lib	45
4.15.5	Вывод файла hello.html	45
4.15.6	Исследуйте самостоятельно	45
4.16	Создание ветки	46
4.16.1	Создадим ветку	46
4.16.2	Добавим файл стилей style.css	46
4.16.3	Изменим основную страницу	47

4.16.4 Изменим index.html	48
4.17 Навигация по веткам	48
4.17.1 Переключение на ветку master	49
4.17.2 Вернёмся к ветке style	50
4.18 Изменения в ветке master	50
4.18.1 Создадим файл README в ветке master	51
4.19 Сделаем коммит изменений README.md в ветку master	51
4.19.1 Просмотр отличающихся веток	51
4.19.2 Просмотрим текущие ветки	51
4.20 Слияние	52
4.20.1 Слияние веток	52
4.21 Создание конфликта	54
4.21.1 Вернёмся в master и создадим конфликт	54
4.21.2 Просмотр веток	54
4.22 Разрешение конфликтов	56
4.22.1 Слияние master с веткой style	56
4.22.2 Решение конфликта	56
4.22.3 Сделаем коммит решения конфликта	57
4.22.4 1.22.4 Перебазирование как альтернатива слиянию	57
4.23 Сброс ветки style	57
4.23.1 Сброс ветки style	57
4.23.2 Проверим ветку	59
4.24 Сброс ветки master	60
4.24.1 Сброс ветки master	60
4.25 Перебазирование	63
4.25.1 Слияние VS перебазирование	64
4.26 Слияние в ветку master	65
4.26.1 Слияние style в master	65
4.26.2 Просмотрим логи	65
4.27 Клонирование репозиториев	66
4.27.1 Перейдём в рабочий каталог	66
4.27.2 Создадим клон репозитория hello	67
4.28 Просмотр клонированного репозитория	67
4.28.1 Взглянем на клонированный репозиторий	67
4.28.2 Просмотрим историю репозитория	68
4.28.3 Удалённые ветки	69
4.29 Что такое origin?	69
4.30 Удалённые ветки	70
4.30.1 Список удалённых веток	70
4.31 Изменение оригинального репозитория	70
4.31.1 Внесём изменения в оригинальный репозиторий hello	71
4.31.2 Извлечение изменений	71
4.31.3 Проверим README.md	73

4.32 Слияние извлечённых изменений	73
4.32.1 Сольём извлечённые изменения в локальную ветку master .	73
4.32.2 Ещё раз проверим файл README.md	74
4.33 Добавление ветки наблюдения	74
4.33.1 Добавим локальную ветку, которая отслеживает удалённую ветку	75
4.34 Чистые репозитории	75
4.35 Создадим чистый репозиторий	76
4.36 Добавление удалённого репозитория	76
4.37 Отправка изменений	76
4.38 Извлечение общих изменений	77
5 Выводы	79
Список литературы	80

Список иллюстраций

4.1 Установка имени и электронной почты	13
4.2 Параметры установки окончаний строк	13
4.3 Избежание нечитаемых строк	14
4.4 Страница с Hello World	14
4.5 Создание нового репозитория	14
4.6 Добавление файла в репозиторий	15
4.7 Текущее состояние репозитория	15
4.8 Содержимое hello.html	15
4.9 Состояние рабочего каталога	16
4.10 Индексация изменений	16
4.11 Коммит	17
4.12 Комментарий	17
4.13 Проверка состояния	17
4.14 Содержимое hello.html	18
4.15 Добавление изменения в индекс git	18
4.16 Содержимое hello.html	18
4.17 Текущий статус	19
4.18 Коммит и проверка состояния	19
4.19 Добавление второго изменения в индекс и проверка состояния	20
4.20 Коммит	20
4.21 Список произведённых изменений	21
4.22 Хэши предыдущих версий	22
4.23 Содержимое hello.html при первом коммите	22
4.24 Возврат к последней версии в ветке master	23
4.25 Тег первой версии	23
4.26 Переключение на предыдущую версию	24
4.27 Тег версии v1-beta	24
4.28 Переключение между двумя отмеченными версиями	24
4.29 Доступные теги	25
4.30 Просмотр тегов через лог	25
4.31 Переход на ветку master	26
4.32 Добавление комментария в hello.html	26
4.33 Состояние рабочего каталога	26
4.34 Переключение версии	27
4.35 Добавление комментария в hello.html	28
4.36 Индексация изменения	28
4.37 Проверка состояния	28

4.38 Сброс буферной зоны к HEAD	29
4.39 Переключение на версию коммита	29
4.40 Отредактированный hello.html	30
4.41 Коммит	30
4.42 Возврат	30
4.43 Редактирование комментария	30
4.44 Журнал	31
4.45 Журнал	33
4.46 Тег для коммита	33
4.47 Сброс ветки	34
4.48 Все коммиты	35
4.49 Удаление тега	36
4.50 Комментарий	37
4.51 Коммит	37
4.52 Редактирование hello.html	37
4.53 Изменение комментария	38
4.54 Журнал	38
4.55 Перенос страницы в каталог lib	39
4.56 Коммит	40
4.57 Файл index.html	40
4.58 Содержимое index.html	40
4.59 Добавление файла и коммит	40
4.60 Результат работы файла index.html	41
4.61 Вся информация о git	41
4.62 База данных объектов	41
4.63 Просмотр одного из каталогов из objects	42
4.64 Содержимое файла конфигурации	42
4.65 Содержимое тега v1	43
4.66 Содержимое файла HEAD	43
4.67 Последний коммит в репозитории	44
4.68 Вывод последнего коммита с помощью SHA1 хэша	44
4.69 Дерево каталогов	45
4.70 Вывод каталога lib	45
4.71 Вывод файла hello.html	45
4.72 Переключение на новую ветку style	46
4.73 Новый CSS-файл	46
4.74 Стиль для заголовка	47
4.75 Добавление файла в репозиторий и коммит	47
4.76 Редактирование файла hello.html	47
4.77 Обновление файла в репозитории и коммит	48
4.78 Редактирование файла index.html	48
4.79 Обновление файла в репозитории и коммит	48
4.80 Журнал	49
4.81 Переключение ветки	50

4.82 Переключение ветки	50
4.83 Переключение ветки	51
4.84 Новый файл README.md	51
4.85 Добавление файла в репозиторий и коммит	51
4.86 Просмотр веток и их отличий	52
4.87 Слияние веток	53
4.88 Дерево коммитов	53
4.89 Переключение ветки	54
4.90 Редактирование файла hello.html	54
4.91 Обновление файла в репозитории и коммит	54
4.92 Дерево коммитов	55
4.93 Конфликт при слиянии веток	56
4.94 Содержимое hello.html	56
4.95 Содержимое hello.html	57
4.96 Обновление файла в репозитории и коммит	57
4.97 Переключение на ветку style	58
4.98 Дерево коммитов	58
4.99 Перемещение к другому коммиту	59
4.100 Дерево коммитов	60
4.101 Переключение на ветку master	61
4.102 Дерево коммитов	61
4.103 Перемещение к другому коммиту	62
4.104 Дерево коммитов	62
4.105 Перебазирование	63
4.106 Журнал	64
4.107 Слияние веток	65
4.108 Журнал	66
4.109 Переход в рабочий каталог	67
4.110 Клон репозитория	67
4.111 Содержимое клонированного репозитория	67
4.112 Журнал	68
4.113 Удалённый репозиторий	69
4.114 Подробная информация об удалённом репозитории	69
4.115 Ветки в репозитории	70
4.116 Просмотр всех веток	70
4.117 Перемещение	71
4.118 Редактируем README	71
4.119 Обновление файла в репозитории и коммит	71
4.120 Перекачиваем файлы	72
4.121 Журнал	72
4.122 Содержимое README	73
4.123 Слияние веток	74
4.124 Версия README-файла	74
4.125 git pull	74

4.12	Отслеживание удалённой ветки	75
4.12	Чистый репозиторий	76
4.12	Добавление репозитория hello.git	76
4.12	Редактирование README.md	77
4.13	Коммит	77
4.13	Отправление изменений в общий репозиторий	77
4.13	Перемещение	78
4.13	Извлечение общих изменений	78

1 Цель работы

Приобрести практические навыки работы с системой управления версиями Git.

2 Задание

Изучить работу Git.

3 Теоретическое введение

Git – распределённая система управления версиями. Проект был создан Линусом Торвальдсом для управления разработкой ядра Linux, первая версия выпущена 7 апреля 2005 года.

4 Выполнение лабораторной работы

4.1 Подготовка

4.1.1 Установка имени и электронной почты

В случае если ранее git не использовался, для начала нам необходимо осуществить установку. Выполним следующие команды, чтобы git узнал наше имя и электронную почту.

```
mensch@aeaskerov:~$ git config --global user.name "aeaskerov"
mensch@aeaskerov:~$ git config --global user.email [REDACTED]
mensch@aeaskerov:~$ █
```

Рис. 4.1: Установка имени и электронной почты

4.1.2 Параметры установки окончаний строк

Укажем параметры установки окончаний строк со значениями true и input.

```
mensch@aeaskerov:~$ git config --global core.autocrlf input
mensch@aeaskerov:~$ git config --global core.safecrlf true
```

Рис. 4.2: Параметры установки окончаний строк

4.1.3 Установка отображения unicode

Установим соответствующий флаг, чтобы избежать нечитаемых строк.

```
mensch@aeaskerov:~$ git config --global core.quotepath off  
mensch@aeaskerov:~$
```

Рис. 4.3: Избежание нечитаемых строк

4.2 Создание проекта

4.2.1 Создадим страницу «Hello World»

Начнём работу в пустом рабочем каталоге с создания пустого каталога с именем hello, затем войдём в него и создадим там файл с именем hello.html.

```
mensch@aeaskerov:~$ mkdir hello  
mensch@aeaskerov:~$ cd hello  
mensch@aeaskerov:~/hello$ touch hello.html  
mensch@aeaskerov:~/hello$ echo "Hello World!" > hello.html  
mensch@aeaskerov:~/hello$
```

Рис. 4.4: Страница с Hello World

4.2.2 Создание репозитория

Чтобы создать git репозиторий из этого каталога, выполним команду git init.

```
mensch@aeaskerov:~/hello$ git init  
hint: Using 'master' as the name for the initial branch. This default branch name  
hint: is subject to change. To configure the initial branch name to use in all  
hint: 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/mensch/hello/.git/  
mensch@aeaskerov:~/hello$
```

Рис. 4.5: Создание нового репозитория

4.2.3 Добавление файла в репозиторий

Добавим файл в репозиторий.

```
mensch@aeaskerov:~/hello$ git add hello.html
mensch@aeaskerov:~/hello$ git commit -m "Initial Commit"
[master (root-commit) 025e34a] Initial Commit
 1 file changed, 1 insertion(+)
 create mode 100644 hello.html
mensch@aeaskerov:~/hello$
```

Рис. 4.6: Добавление файла в репозиторий

4.2.4 Проверка состояния репозитория

Проверим текущее состояние репозитория.

```
mensch@aeaskerov:~/hello$ git status
On branch master
nothing to commit, working tree clean
mensch@aeaskerov:~/hello$
```

Рис. 4.7: Текущее состояние репозитория

Видим, что коммитить нечего – в репозитории хранится текущее состояние рабочего каталога, и нет никаких изменений, ожидающих записи.

4.3 Внесение изменений

4.3.1 Изменим страницу «Hello World»

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

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

Рис. 4.8: Содержимое hello.html

Проверим состояние рабочего каталога.

```
mensch@aeaskerov:~/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")
mensch@aeaskerov:~/hello$
```

Рис. 4.9: Состояние рабочего каталога

Видно, что git знает, что файл hello.html был изменён, но при этом эти изменения ещё не зафиксированы в репозитории.

4.4 Индексация изменений

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

```
mensch@aeaskerov:~/hello$ git add hello.html
mensch@aeaskerov:~/hello$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    modified:   hello.html

mensch@aeaskerov:~/hello$
```

Рис. 4.10: Индексация изменений

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.

4.4.1 Коммит изменений

Сделаем коммит и проверим состояние.

```
mensch@aeaskerov:~/hello$ git commit
```



Рис. 4.11: Коммит

Откроется редактор.

В первой строке введём комментарий: «Added h1 tag». Сохраним файл и выйдем из редактора.

```
mensch@aeaskerov:~/hello  
GNU nano 7.2          /home/mensch/hello/.git/COMMIT_EDITMSG *  
Added h1 tag  
# 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. 4.12: Комментарий

Теперь ещё раз проверим состояние.

```
mensch@aeaskerov:~/hello$ git status  
On branch master  
nothing to commit, working tree clean  
mensch@aeaskerov:~/hello$
```

Рис. 4.13: Проверка состояния

Рабочий каталог чистый, можно продолжить работу.

4.4.2 Добавим стандартные теги страницы

Изменим страницу «Hello World», чтобы она содержала стандартные теги `html` и `body`.

```
<html>
  <body>
    <h1>Hello World!</h1>
  </body>
</html>
```

Рис. 4.14: Содержимое hello.html

Теперь добавим это изменение в индекс git.

```
mensch@aeaskerov:~/hello$ git add hello.html
mensch@aeaskerov:~/hello$
```

Рис. 4.15: Добавление изменения в индекс git

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

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

Рис. 4.16: Содержимое hello.html

Проверим текущий статус.

```
mensch@aeaskerov:~/hello$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    modified:   hello.html

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

mensch@aeaskerov:~/hello$
```

Рис. 4.17: Текущий статус

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 мы делали коммит сейчас, заголовки не были бы сохранены в репозиторий.

Произведём коммит проиндексированного изменения (значение по умолчанию), а затем ещё раз проверим состояние.

```
mensch@aeaskerov:~/hello$ git commit -m "Added standard HTML page tags"
[master 37e4800] Added standard HTML page tags
 1 file changed, 5 insertions(+), 1 deletion(-)
mensch@aeaskerov:~/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")
mensch@aeaskerov:~/hello$
```

Рис. 4.18: Коммит и проверка состояния

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.

```
mensch@aeaskerov:~/hello$ git add .
mensch@aeaskerov:~/hello$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    modified:   hello.html

mensch@aeaskerov:~/hello$
```

Рис. 4.19: Добавление второго изменения в индекс и проверка состояния

Второе изменение проиндексировано и готово к коммиту.

Сделаем коммит второго изменения.

```
mensch@aeaskerov:~/hello$ git commit -m "Added HTML header"
[master 6d6170e] Added HTML header
 1 file changed, 2 insertions(+)
mensch@aeaskerov:~/hello$
```

Рис. 4.20: Коммит

4.4.3 История

Получим список произведённых изменений.

```
mensch@aeaskerov:~/hello$ git log
commit 6d6170e41f31010a5324e8e1294bc5elbfc29dd2 (HEAD -> master)
Author: aeaskerov <iqwertydragoni@gmail.com>
Date:   Wed Feb 12 17:07:01 2025 +0300

    Added HTML header

commit 37e4800d3dc24b1a3523ef9c3b72a20b0ba19f0c
Author: aeaskerov <iqwertydragoni@gmail.com>
Date:   Wed Feb 12 17:06:02 2025 +0300

    Added standard HTML page tags

commit 0c8cb2a3d946388c341fa423d0cac83f400b4d9e
Author: aeaskerov <iqwertydragoni@gmail.com>
Date:   Wed Feb 12 17:02:16 2025 +0300

    Added h1 tag

commit 025e34ae5aaeece623c326a645109310a43130ec
Author: aeaskerov <iqwertydragoni@gmail.com>
Date:   Wed Feb 12 16:57:58 2025 +0300

    Initial Commit
mensch@aeaskerov:~/hello$
```

Рис. 4.21: Список произведённых изменений

4.4.4 Получение старых версий

Команда `checkout` позволяет вернуться назад в историю – она копирует любой снимок из репозитория в рабочий каталог.

Получим хэши предыдущих версий.

```

mensch@aeaskerov:~/hello$ git log
commit 6d6170e41f31010a5324e8e1294bc5elbfc29dd2 (HEAD -> master)
Author: aeaskerov <iqwertydragoni@gmail.com>
Date:   Wed Feb 12 17:07:01 2025 +0300

    Added HTML header

commit 37e4800d3dc24b1a3523ef9c3b72a20b0ba19f0c
Author: aeaskerov <iqwertydragoni@gmail.com>
Date:   Wed Feb 12 17:06:02 2025 +0300

    Added standard HTML page tags

commit 0c8cb2a3d946388c341fa423d0cac83f400b4d9e
Author: aeaskerov <iqwertydragoni@gmail.com>
Date:   Wed Feb 12 17:02:16 2025 +0300

    Added h1 tag

commit 025e34ae5aaeece623c326a645109310a43130ec
Author: aeaskerov <iqwertydragoni@gmail.com>
Date:   Wed Feb 12 16:57:58 2025 +0300

    Initial Commit
mensch@aeaskerov:~/hello$
```

Рис. 4.22: Хэши предыдущих версий

Изучим данные лога и найдём 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.

```

mensch@aeaskerov:~/hello$ git checkout 025e34ae
Note: switching to '025e34ae'.

You are in 'detached HEAD' state. You can look around, make experimental
changes and commit them, and you can discard any commits you make in this
state without impacting any branches by switching back to a branch.

If you want to create a new branch to retain commits you create, you may
do so (now or later) by using -c with the switch command. Example:

  git switch -c <new-branch-name>

Or undo this operation with:

  git switch -

Turn off this advice by setting config variable advice.detachedHead to false

HEAD is now at 025e34a Initial Commit
mensch@aeaskerov:~/hello$ cat hello.html
Hello World!
mensch@aeaskerov:~/hello$
```

Рис. 4.23: Содержимое hello.html при первом коммите

Вернёмся к последней версии в ветке master.

```
mensch@aeaskerov:~/hello$ git checkout master
Previous HEAD position was 025e34a Initial Commit
Switched to branch 'master'
mensch@aeaskerov:~/hello$ cat hello.html
<html>
  <head>
  </head>
  <body>
    <h1>Hello World!</h1>
  </body>
</html>
mensch@aeaskerov:~/hello$
```

Рис. 4.24: Возврат к последней версии в ветке master

master – имя ветки по умолчанию. Переключая имена веток, мы попадаем на последнюю версию выбранной ветки.

4.4.5 Создание тегов версий

Назовём текущую версию страницы hello первой (v1).

Создадим тег первой версии.

```
mensch@aeaskerov:~/hello$ git tag v1
mensch@aeaskerov:~/hello$
```

Рис. 4.25: Тег первой версии

Теперь текущая версия страницы называется v1.

После этого создадим тег для версии, которая идёт перед текущей версией и назовем её v1-beta. В первую очередь нам надо переключиться на предыдущую версию.

```
mensch@aeaskerov:~/hello$ git checkout v1^
Note: switching to 'v1^'.

You are in 'detached HEAD' state. You can look around, make experimental
changes and commit them, and you can discard any commits you make in this
state without impacting any branches by switching back to a branch.

If you want to create a new branch to retain commits you create, you may
do so (now or later) by using -c with the switch command. Example:

  git switch -c <new-branch-name>

Or undo this operation with:

  git switch -

Turn off this advice by setting config variable advice.detachedHead to false

HEAD is now at 37e4800 Added standard HTML page tags
mensch@aeaskerov:~/hello$ cat hello.html
<html>
  <body>
    <h1>Hello World!</h1>
  </body>
</html>
mensch@aeaskerov:~/hello$
```

Рис. 4.26: Переключение на предыдущую версию

Это версия с тегами html и body, но ещё пока без head. Давайте сделаем её версией v1-beta.

```
mensch@aeaskerov:~/hello$ git tag v1-beta
mensch@aeaskerov:~/hello$
```

Рис. 4.27: Тег версии v1-beta

4.4.6 Переключение по имени тега

Теперь попробуем попереключаться между двумя отмеченными версиями.

```
mensch@aeaskerov:~/hello$ git checkout v1
Previous HEAD position was 37e4800 Added standard HTML page tags
HEAD is now at 6d6170e Added HTML header
mensch@aeaskerov:~/hello$ git checkout v1-beta
Previous HEAD position was 6d6170e Added HTML header
HEAD is now at 37e4800 Added standard HTML page tags
mensch@aeaskerov:~/hello$
```

Рис. 4.28: Переключение между двумя отмеченными версиями

4.4.7 Просмотр тегов с помощью команды tag

Мы можем увидеть, какие теги доступны, используя команду git tag.

```
mensch@aeaskerov:~/hello$ git tag
v1
v1-beta
mensch@aeaskerov:~/hello$
```

Рис. 4.29: Доступные теги

Мы также можем посмотреть теги в логе.

```
mensch@aeaskerov:~/hello$ git log master --all
commit 6d6170e41f31010a5324e8e1294bc5e1bfc29dd2 (tag: v1, master)
Author: aeaskerov <iqwertydragoni@gmail.com>
Date:   Wed Feb 12 17:07:01 2025 +0300

    Added HTML header

commit 37e4800d3dc24b1a3523ef9c3b72a20b0ba19f0c (HEAD, tag: v1-beta)
Author: aeaskerov <iqwertydragoni@gmail.com>
Date:   Wed Feb 12 17:06:02 2025 +0300

    Added standard HTML page tags

commit 0c8cb2a3d946388c341fa423d0cac83f400b4d9e
Author: aeaskerov <iqwertydragoni@gmail.com>
Date:   Wed Feb 12 17:02:16 2025 +0300

    Added h1 tag

commit 025e34ae5aaeece623c326a645109310a43130ec
Author: aeaskerov <iqwertydragoni@gmail.com>
Date:   Wed Feb 12 16:57:58 2025 +0300

    Initial Commit
mensch@aeaskerov:~/hello$
```

Рис. 4.30: Просмотр тегов через лог

4.5 Отмена локальных изменений (до индексации)

4.5.1 Переключимся на ветку master

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

```
mensch@aeaskerov:~/hello$ git checkout master
Previous HEAD position was 37e4800 Added standard HTML page tags
Switched to branch 'master'
mensch@aeaskerov:~/hello$
```

Рис. 4.31: Переход на ветку master

4.5.2 Изменим hello.html

Иногда случается, что мы изменили файл в рабочем каталоге, и хотим отменить последние коммиты. С этим справится команда `git checkout`.

Внесём 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>
```

Рис. 4.32: Добавление комментария в `hello.html`

4.5.3 Проверим состояние

Сначала проверим состояние рабочего каталога.

```
mensch@aeaskerov:~/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")
mensch@aeaskerov:~/hello$
```

Рис. 4.33: Состояние рабочего каталога

Мы видим, что файл `hello.html` был изменен, но еще не проиндексирован.

4.5.4 Отмена изменений в рабочем каталоге

Используем команду git checkout для переключения версии файла hello.html в репозитории.

```
mensch@aeaskerov:~/hello$ git checkout hello.html
Updated 1 path from the index
mensch@aeaskerov:~/hello$ git status
On branch master
nothing to commit, working tree clean
mensch@aeaskerov:~/hello$ cat hello.html
<html>
  <head>
  </head>
  <body>
    <h1>Hello World!</h1>
  </body>
</html>
mensch@aeaskerov:~/hello$
```

Рис. 4.34: Переключение версии

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

4.6 Отмена проиндексированных изменений (перед коммитом)

4.6.1 Изменим файл и проиндексируем изменения

Внесём 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>
```

Рис. 4.35: Добавление комментария в hello.html

Проиндексируем это изменение.

```
mensch@aeaskerov:~/hello$ git add hello.html
mensch@aeaskerov:~/hello$
```

Рис. 4.36: Индексация изменения

4.6.2 Проверим состояние

Проверим состояние нежелательного изменения.

```
mensch@aeaskerov:~/hello$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    modified:   hello.html

mensch@aeaskerov:~/hello$
```

Рис. 4.37: Проверка состояния

Состояние показывает, что изменение было проиндексировано и готово к коммиту.

4.6.3 Выполним сброс буферной зоны

К счастью, вывод состояния показывает нам именно то, что мы должны сделать для отмены индексации изменения.

```
mensch@aeaskerov:~/hello$ git reset HEAD hello.html
Unstaged changes after reset:
M      hello.html
mensch@aeaskerov:~/hello$
```

Рис. 4.38: Сброс буферной зоны к HEAD

Команда `git reset` сбрасывает буферную зону к HEAD. Это очищает буферную зону от изменений, которые мы только что проиндексировали.

4.6.4 Переключимся на версию коммита

```
mensch@aeaskerov:~/hello$ git checkout hello.html
Updated 1 path from the index
mensch@aeaskerov:~/hello$ git status
On branch master
nothing to commit, working tree clean
mensch@aeaskerov:~/hello$
```

Рис. 4.39: Переключение на версию коммита

Наш рабочий каталог опять чист.

4.7 Отмена коммитов

4.7.1 Отмена коммитов

Иногда мы понимаем, что новые коммиты являются неверными, и хотим их отменить. Есть несколько способов решения этого вопроса, здесь мы будем использовать самый безопасный. Мы отменим коммит путём создания нового коммита, отменяющего нежелательные изменения.

4.7.2 Изменим файл и сделаем коммит

Изменим файл `hello.html` на следующий.

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

Рис. 4.40: Отредактированный hello.html

Выполним следующие команды.

```
mensch@aeaskerov:~/hello$ git add hello.html
mensch@aeaskerov:~/hello$ git commit -m "Oops, we didn't want this commit"
[master 615b911] Oops, we didn't want this commit
  1 file changed, 1 insertion(+)
mensch@aeaskerov:~/hello$
```

Рис. 4.41: Коммит

4.7.3 Сделаем коммит с новыми изменениями, отменяющими предыдущие

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

```
mensch@aeaskerov:~/hello$ git revert HEAD
```

Рис. 4.42: Возврат

Перейдём в редактор, где мы можем отредактировать коммит-сообщение по умолчанию или оставить все как есть. Сохраним и закроем файл.

```
mensch@aeaskerov:~/hello
GNU nano 7.2          /home/mensch/hello/.git/COMMIT_EDITMSG *
Revert "We want this commit"

This reverts commit 615b911df1e21445a27b01c5d6fdc8ef6e49bcfa.
```

Рис. 4.43: Редактирование комментария

4.7.4 Проверим лог

Проверка лога показывает нежелательные и отменённые коммиты в наш репозиторий.

```
mensch@aeaskerov:~/hello$ git log
commit 57696b51d69b32fbad9f40bfb712b35283ff5970 (HEAD -> master)
Author: aeaskerov <iqwertydragoni@gmail.com>
Date:   Wed Feb 12 17:20:37 2025 +0300

    Revert "We want this commit"

    This reverts commit 615b911df1e21445a27b01c5d6fdc8ef6e49bcfa.

commit 615b911df1e21445a27b01c5d6fdc8ef6e49bcfa
Author: aeaskerov <iqwertydragoni@gmail.com>
Date:   Wed Feb 12 17:20:20 2025 +0300

    Oops, we didn't want this commit

commit 6d6170e41f31010a5324e8e1294bc5e1bfc29dd2 (tag: v1)
Author: aeaskerov <iqwertydragoni@gmail.com>
Date:   Wed Feb 12 17:07:01 2025 +0300

    Added HTML header

commit 37e4800d3dc24b1a3523ef9c3b72a20b0ba19f0c (tag: v1-beta)
Author: aeaskerov <iqwertydragoni@gmail.com>
Date:   Wed Feb 12 17:06:02 2025 +0300

    Added standard HTML page tags

commit 0c8cb2a3d946388c341fa423d0cac83f400b4d9e
mensch@aeaskerov:~/hello$
```

Рис. 4.44: Журнал

4.8 Удаление коммитов из ветки

`git revert` является мощной командой, которая позволяет отменить любые коммиты в репозиторий. Однако, и оригиналный и «отменённый» коммиты видны в истории ветки (при использовании команды `git log`).

Часто мы делаем коммит, и сразу понимаем, что это была ошибка. Было бы неплохо иметь команду «возврата», которая позволила бы нам сделать вид, что неправильного коммита никогда и не было. Команда «возврата» даже предотвратила бы появление нежелательного коммита в истории `git log`.

4.8.1 Команда git reset

При получении ссылки на коммит (т.е. хэш, ветка или имя тега), команда git reset:

- перепишет текущую ветку, чтобы она указывала на нужный коммит;
- опционально сбросит буферную зону для соответствия с указанным коммитом;
- опционально сбросит рабочий каталог для соответствия с указанным коммитом.

4.8.2 Проверим нашу историю

Давайте сделаем быструю проверку нашей истории коммитов. Выполним следующее.

```
mensch@aeaskerov:~/hello$ git log
commit 57696b51d69b32fbad9f40bf712b35283ff5970 (HEAD -> master)
Author: aeaskerov <iqwertydragoni@gmail.com>
Date:   Wed Feb 12 17:20:37 2025 +0300

    Revert "We want this commit"

    This reverts commit 615b911df1e21445a27b01c5d6fdc8ef6e49bcfa.

commit 615b911df1e21445a27b01c5d6fdc8ef6e49bcfa
Author: aeaskerov <iqwertydragoni@gmail.com>
Date:   Wed Feb 12 17:20:20 2025 +0300

    Oops, we didn't want this commit

commit 6d6170e41f31010a5324e8e1294bc5e1bfc29dd2 (tag: v1)
Author: aeaskerov <iqwertydragoni@gmail.com>
Date:   Wed Feb 12 17:07:01 2025 +0300

    Added HTML header

commit 37e4800d3dc24b1a3523ef9c3b72a20b0ba19f0c (tag: v1-beta)
Author: aeaskerov <iqwertydragoni@gmail.com>
Date:   Wed Feb 12 17:06:02 2025 +0300

    Added standard HTML page tags

commit 0c8cb2a3d946388c341fa423d0cac83f400b4d9e
Author: aeaskerov <iqwertydragoni@gmail.com>
Date:   Wed Feb 12 17:02:16 2025 +0300

mensch@aeaskerov:~/hello$ █
```

Рис. 4.45: Журнал

Мы видим, что два последних коммита в этой ветке – «Oops» и «Revert Oops».

Удалим их с помощью сброса.

4.8.3 Для начала отметим эту ветку

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

```
mensch@aeaskerov:~/hello$ git tag oops
mensch@aeaskerov:~/hello$
```

Рис. 4.46: Тег для коммита

4.8.4 Сброс коммитов к предшествующим коммиту Oops

Глядя на историю лога, мы видим, что коммит с тегом «v1» является коммитом, предшествующим ошибочному коммиту. 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).

```
mensch@aeaskerov:~/hello$ git reset --hard v1
HEAD is now at 6d6170e Added HTML header
mensch@aeaskerov:~/hello$ git log
commit 6d6170e41f31010a5324e8e1294bc5e1bfc29dd2 (HEAD -> master, tag: v1)
Author: aeaskerov <iqwertydragoni@gmail.com>
Date:   Wed Feb 12 17:07:01 2025 +0300

    Added HTML header

commit 37e4800d3dc24b1a3523ef9c3b72a20b0ba19f0c (tag: v1-beta)
Author: aeaskerov <iqwertydragoni@gmail.com>
Date:   Wed Feb 12 17:06:02 2025 +0300

    Added standard HTML page tags

commit 0c8cb2a3d946388c341fa423d0cac83f400b4d9e
Author: aeaskerov <iqwertydragoni@gmail.com>
Date:   Wed Feb 12 17:02:16 2025 +0300

    Added h1 tag

commit 025e34ae5aaeece623c326a645109310a43130ec
Author: aeaskerov <iqwertydragoni@gmail.com>
Date:   Wed Feb 12 16:57:58 2025 +0300

    Initial Commit
mensch@aeaskerov:~/hello$
```

Рис. 4.47: Сброс ветки

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.

4.8.5 Ничего никогда не теряется

Что же случается с ошибочными коммитами? Оказывается, что коммиты всё ещё находятся в репозитории. На самом деле, мы всё ещё можем на них ссылаться. Например, в начале этого урока мы создали для отменённого коммита тег «oops». Посмотрим на все коммиты.

```
mensch@aeaskerov:~/hello$ git log --all
commit 57696b51d69b32fbad9f40bfb712b35283ff5970 (tag: oops)
Author: aeaskerov <iqwertydragoni@gmail.com>
Date:   Wed Feb 12 17:20:37 2025 +0300

    Revert "We want this commit"

    This reverts commit 615b911df1e21445a27b01c5d6fdc8ef6e49bcfa.

commit 615b911df1e21445a27b01c5d6fdc8ef6e49bcfa
Author: aeaskerov <iqwertydragoni@gmail.com>
Date:   Wed Feb 12 17:20:20 2025 +0300

    Oops, we didn't want this commit

commit 6d6170e41f31010a5324e8e1294bc5e1bfc29dd2 (HEAD -> master, tag: v1)
Author: aeaskerov <iqwertydragoni@gmail.com>
Date:   Wed Feb 12 17:07:01 2025 +0300

    Added HTML header

commit 37e4800d3dc24bla3523ef9c3b72a20b0ba19f0c (tag: v1-beta)
Author: aeaskerov <iqwertydragoni@gmail.com>
Date:   Wed Feb 12 17:06:02 2025 +0300

    Added standard HTML page tags

commit 0c8cb2a3d946388c341fa423d0cac83f400b4d9e
Author: aeaskerov <iqwertydragoni@gmail.com>
Date:   Wed Feb 12 17:02:16 2025 +0300

    Added h1 tag

commit 025e34ae5aaeece623c326a645109310a43130ec
Author: aeaskerov <iqwertydragoni@gmail.com>
mensch@aeaskerov:~/hello$
```

Рис. 4.48: Все коммиты

Мы видим, что ошибочные коммиты не исчезли. Они всё ещё находятся в репозитории. Просто они отсутствуют в ветке master. Если бы мы не отметили их тегами, они по-прежнему находились бы в репозитории, но не было бы никакой возможности ссылаться на них, кроме как при помощи их хэш-имён. Коммиты, на которые нет ссылок, остаются в репозитории до тех пор, пока не будет запущен сборщик мусора.

4.8.6 Опасность сброса

Сброс в локальных ветках, как правило, безопасен. Последствия любой «аварии» как правило, можно восстановить простым сбросом с помощью нужного коммита. Однако, если ветка «расшарена» на удалённых репозиториях, сброс может сбить с толку других пользователей ветки.

4.9 Удаление тега oops

4.9.1 Удаление тега oops

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

```
mensch@aeaskerov:~/hello$ git tag -d oops
Deleted tag 'oops' (was 57696b5)
mensch@aeaskerov:~/hello$ git log --all
commit 6d6170e41f31010a5324e8e1294bc5e1bfc29dd2 (HEAD -> master, tag: v1)
Author: aeaskerov <iqwertydragoni@gmail.com>
Date:   Wed Feb 12 17:07:01 2025 +0300

    Added HTML header

commit 37e4800d3dc24b1a3523ef9c3b72a20b0ba19f0c (tag: v1-beta)
Author: aeaskerov <iqwertydragoni@gmail.com>
Date:   Wed Feb 12 17:06:02 2025 +0300

    Added standard HTML page tags

commit 0c8cb2a3d946388c341fa423d0cac83f400b4d9e
Author: aeaskerov <iqwertydragoni@gmail.com>
Date:   Wed Feb 12 17:02:16 2025 +0300

    Added h1 tag

commit 025e34ae5aaeece623c326a645109310a43130ec
Author: aeaskerov <iqwertydragoni@gmail.com>
Date:   Wed Feb 12 16:57:58 2025 +0300

    Initial Commit
mensch@aeaskerov:~/hello$
```

Рис. 4.49: Удаление тега

Тег «oops» больше не будет отображаться в репозитории.

4.10 Внесение изменений в коммиты

4.10.1 Изменим страницу, а затем сделаем коммит

Добавим в страницу комментарий автора.

```
<!-- Author: Alexander Askerov -->
<html>
  <head>
  </head>
  <body>
    <h1>Hello World!</h1>
  </body>
</html>
```

Рис. 4.50: Комментарий

Выполним следующее.

```
mensch@aeaskerov:~/hello$ git add hello.html
mensch@aeaskerov:~/hello$ git commit -m "Add an author comment"
[master fb9f590] Add an author comment
 1 file changed, 1 insertion(+)
mensch@aeaskerov:~/hello$
```

Рис. 4.51: Коммит

4.10.2 Необходим email

После совершения коммита к комментарию стоит добавить электронную почту автора.

Обновим страницу hello, включив в неё email.

```
<!-- Author: Alexander Askerov (1132226538@pfur.ru) -->
<html>
  <head>
  </head>
  <body>
    <h1>Hello World!</h1>
  </body>
</html>
```

Рис. 4.52: Редактирование hello.html

4.10.3 Изменим предыдущий коммит

Мы не хотим создавать отдельный коммит только ради электронной почты.
Поэтому изменим предыдущий коммит, включив в него адрес электронной почты.

```
mensch@aeaskerov:~/hello$ git add hello.html
mensch@aeaskerov:~/hello$ git commit --amend -m "Add an author/email comment"
[master 2293f4f] Add an author/email comment
  Date: Wed Feb 12 18:03:14 2025 +0300
  1 file changed, 1 insertion(+)
mensch@aeaskerov:~/hello$
```

Рис. 4.53: Изменение комментария

4.10.4 Просмотр истории

Выполним следующее.

```
mensch@aeaskerov:~/hello$ git log
commit 2293f4f90a699e8c290e133a4b502dbed61a9bcf (HEAD -> master)
Author: aeaskerov <iqwertydragoni@gmail.com>
Date:   Wed Feb 12 18:03:14 2025 +0300

  Add an author/email comment

commit 6d6170e41f31010a5324e8e1294bc5e1bfc29dd2 (tag: v1)
Author: aeaskerov <iqwertydragoni@gmail.com>
Date:   Wed Feb 12 17:07:01 2025 +0300

  Added HTML header

commit 37e4800d3dc24b1a3523ef9c3b72a20b0ba19f0c (tag: v1-beta)
Author: aeaskerov <iqwertydragoni@gmail.com>
Date:   Wed Feb 12 17:06:02 2025 +0300

  Added standard HTML page tags

commit 0c8cb2a3d946388c341fa423d0cac83f400b4d9e
Author: aeaskerov <iqwertydragoni@gmail.com>
Date:   Wed Feb 12 17:02:16 2025 +0300

  Added h1 tag

commit 025e34ae5aaeece623c326a645109310a43130ec
Author: aeaskerov <iqwertydragoni@gmail.com>
Date:   Wed Feb 12 16:57:58 2025 +0300

  Initial Commit
mensch@aeaskerov:~/hello$
```

Рис. 4.54: Журнал

Мы можем увидеть, что оригинальный коммит «автор» заменён коммитом «автор/email». Этого же эффекта можно достичь путём сброса последнего коммита в ветке, и повторного коммита новых изменений.

4.11 Перемещение файлов

4.11.1 Переместим файл hello.html в каталог lib

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

```
mensch@aeaskerov:~/hello$ mkdir lib
mensch@aeaskerov:~/hello$ git mv hello.html lib
mensch@aeaskerov:~/hello$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    renamed:    hello.html -> lib/hello.html

mensch@aeaskerov:~/hello$
```

Рис. 4.55: Перенос страницы в каталог lib

4.12 Второй способ перемещения файлов

Мы могли бы выполнить:

```
mkdir lib
mv hello.html lib
git add lib/hello.html
git rm hello.html
```

4.12.1 Коммит в новый каталог

Сделаем коммит этого перемещения.

```
mensch@aeaskerov:~/hello$ git commit -m "Moved hello.html to lib"
[master cfb05ab] Moved hello.html to lib
 1 file changed, 0 insertions(+), 0 deletions(-)
 rename hello.html => lib/hello.html (100%)
mensch@aeaskerov:~/hello$
```

Рис. 4.56: Коммит

4.13 Подробнее о структуре

4.13.1 Добавление index.html

Добавим файл index.html в наш репозиторий.

```
mensch@aeaskerov:~/hello$ touch index.html
mensch@aeaskerov:~/hello$
```

Рис. 4.57: Файл index.html

Укажем в нём следующее содержимое.

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

Рис. 4.58: Содержимое index.html

Добавим файл и сделаем коммит.

```
mensch@aeaskerov:~/hello$ git add index.html
mensch@aeaskerov:~/hello$ git commit -m "Added index.html."
[master c277b5f] Added index.html.
 1 file changed, 5 insertions(+)
 create mode 100644 index.html
mensch@aeaskerov:~/hello$
```

Рис. 4.59: Добавление файла и коммит

Теперь при открытии index.html, мы увидим кусок страницы hello в маленьком окошке.



Рис. 4.60: Результат работы файла index.html

4.14 Git внутри: Каталог .git

4.14.1 Каталог .git

Выполним следующее.

```
mensch@aeaskerov:~/hello$ ls -C .git  
branches config HEAD index logs ORIG_HEAD refs  
COMMIT_EDITMSG description hooks info objects packed-refs  
mensch@aeaskerov:~/hello$
```

Рис. 4.61: Вся информация о git

Это каталог, в котором хранится вся информация git.

4.14.2 База данных объектов

Выполним следующее.

```
mensch@aeaskerov:~/hello$ ls -C .git/objects  
00 0c 23 37 41 4c 58 6d 90 98 a9 af cf fb pack  
02 22 2f 3d 45 57 61 81 96 a1 ae c2 de info  
mensch@aeaskerov:~/hello$
```

Рис. 4.62: База данных объектов

Мы видим набор каталогов, имена которых состоят из 2 символов. Имена каталогов являются первыми двумя буквами хэша sha1 объекта, хранящегося в git.

4.14.3 Углубляемся в базу данных объектов

Выполним следующее.

```
mensch@aeaskerov:~/hello$ ls -C .git/objects/2f  
0fdb072a938dbdc2f15815d3465e321d425d30  
mensch@aeaskerov:~/hello$
```

Рис. 4.63: Просмотр одного из каталогов из objects

Здесь мы смотрим в один из каталогов с именем из 2 букв. В результате мы видим файлы с именами из 38 символов. Это файлы, содержащие объекты, хранящиеся в git. Они сжаты и закодированы, поэтому просмотр их содержимого может мало чем помочь.

4.14.4 Config File

Выполним следующее.

```
mensch@aeaskerov:~/hello$ cat .git/config  
[core]  
    repositoryformatversion = 0  
    filemode = true  
    bare = false  
    logallrefupdates = true  
mensch@aeaskerov:~/hello$
```

Рис. 4.64: Содержимое файла конфигурации

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

4.14.5 Ветки и теги

Выполним следующее.

```
mensch@aeaskerov:~/hello$ ls .git/refs  
heads tags  
mensch@aeaskerov:~/hello$ ls .git/refs/heads  
master  
mensch@aeaskerov:~/hello$ ls .git/refs/tags  
v1 v1-beta  
mensch@aeaskerov:~/hello$ cat .git/refs/tags/v1  
6d6170e41f31010a5324e8e1294bc5e1bfc29dd2  
mensch@aeaskerov:~/hello$
```

Рис. 4.65: Содержимое тега v1

Здесь каждый файл соответствует тегу, ранее созданному с помощью команды `git tag`. Его содержание – это всего лишь хэш коммита, привязанный к тегу.

Каталог `heads` практически аналогичен, но используется для веток, а не тегов. На данный момент у нас есть только одна ветка, так что всё, что мы увидим в этом каталоге – это ветка `master`.

4.14.6 Файл HEAD

Выполним следующее.

```
mensch@aeaskerov:~/hello$ cat .git/HEAD  
ref: refs/heads/master  
mensch@aeaskerov:~/hello$
```

Рис. 4.66: Содержимое файла HEAD

Файл `HEAD` содержит ссылку на текущую ветку, в данный момент это должна быть ветка `master`.

4.15 Работа непосредственно с объектами git

4.15.1 Поиск последнего коммита

Выполним следующее.

```
mensch@aeaskerov:~/hello$ git log --max-count=1
commit c277b5f878513318c817ba8cf514430f47f4ae96 (HEAD -> master)
Author: aeaskerov <iqwertydragoni@gmail.com>
Date:   Wed Feb 12 18:17:56 2025 +0300

    Added index.html.
mensch@aeaskerov:~/hello$
```

Рис. 4.67: Последний коммит в репозитории

Эта команда показывает последний коммит в репозиторий. SHA1 хэш у разных пользователей, вероятно, отличается от этого, но отображаемый результат похож на этот.

4.15.2 Вывод последнего коммита с помощью SHA1 хэша

Выполним следующее.

```
mensch@aeaskerov:~/hello$ git cat-file -t c277b5f
commit
mensch@aeaskerov:~/hello$ git cat-file -p c277b5f
tree 58e6fe41b6681e6b52c20c253b4832c08946530c
parent cfb05ab92398429329bd286d13ec33b662cc2c8c
author aeaskerov <iqwertydragoni@gmail.com> 1739373476 +0300
committer aeaskerov <iqwertydragoni@gmail.com> 1739373476 +0300

    Added index.html.
mensch@aeaskerov:~/hello$
```

Рис. 4.68: Вывод последнего коммита с помощью SHA1 хэша

4.15.3 Поиск дерева

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

Выполним следующее.

```
mensch@aeaskerov:~/hello$ git cat-file -p 58e6fe4
100644 blob aeeb538cd0610f43516914c29faaae04e80a870d    index.html
040000 tree a943f8b6a35af6878f3e57fb57766e0f29afd7f8    lib
mensch@aeaskerov:~/hello$
```

Рис. 4.69: Дерево каталогов

4.15.4 Вывод каталога lib

Выполним следующее.

```
mensch@aeaskerov:~/hello$ git cat-file -p a943f8b
100644 blob 232459e185f9088bd2b850537c2cc4dcffc2974    hello.html
mensch@aeaskerov:~/hello$
```

Рис. 4.70: Вывод каталога lib

4.15.5 Вывод файла hello.html

Выполним следующее.

```
mensch@aeaskerov:~/hello$ git cat-file -p 232459e
<!-- Author: Alexander Askerov (1132226538@pfur.ru) -->
<html>
  <head>
  </head>
  <body>
    <h1>Hello World!</h1>
  </body>
</html>
mensch@aeaskerov:~/hello$
```

Рис. 4.71: Вывод файла hello.html

4.15.6 Исследуйте самостоятельно

Исследуем git репозиторий вручную самостоятельно. Посмотрим, удастся ли нам найти оригиналный файл hello.html с самого первого коммита вручную по ссылкам SHA1 хэша в последнем коммите.

4.16 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.

4.16.1 Создадим ветку

Назовём нашу новую ветку «style».

Выполним следующее.

```
mensch@aeaskerov:~/hello$ git checkout -b style
Switched to a new branch 'style'
mensch@aeaskerov:~/hello$ git status
On branch style
nothing to commit, working tree clean
mensch@aeaskerov:~/hello$
```

Рис. 4.72: Переключение на новую ветку style

Команда git status сообщает о том, что мы находимся в ветке «style».

4.16.2 Добавим файл стилей style.css

Выполним следующее.

```
mensch@aeaskerov:~/hello$ touch lib/style.css
mensch@aeaskerov:~/hello$
```

Рис. 4.73: Новый CSS-файл

Укажем в нём следующее содержимое.

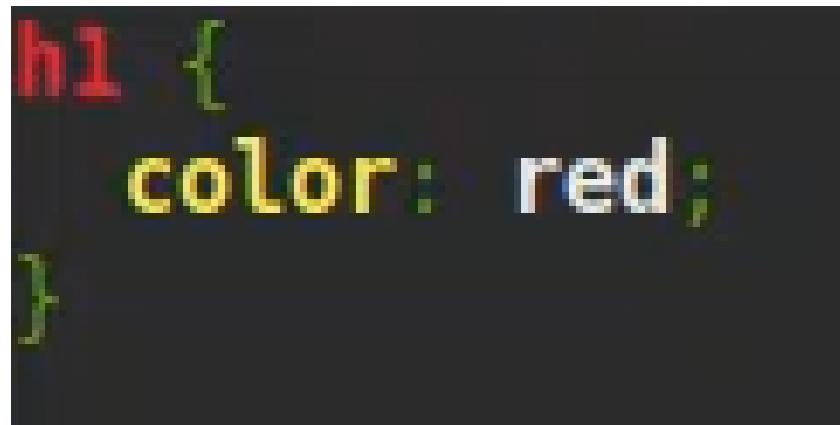


Рис. 4.74: Стиль для заголовка

Выполним следующее.

```
mensch@aeaskerov:~/hello$ git add lib/style.css
mensch@aeaskerov:~/hello$ git commit -m "Added css stylesheet"
[style c2ab60] Added css stylesheet
 1 file changed, 3 insertions(+)
  create mode 100644 lib/style.css
mensch@aeaskerov:~/hello$
```

Рис. 4.75: Добавление файла в репозиторий и коммит

4.16.3 Изменим основную страницу

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

```
<!-- Author: Alexander Askerov (1132226538@pfur.ru) -->
<html>
  <head>
    <link type="text/css" rel="stylesheet" media="all" href="style.css" />
  </head>
  <body>
    <h1>Hello World!</h1>
  </body>
</html>
```

Рис. 4.76: Редактирование файла hello.html

Выполним следующее.

```
mensch@aeaskerov:~/hello$ git add lib/hello.html
mensch@aeaskerov:~/hello$ git commit -m "Hello uses style.css"
[style 6345e56] Hello uses style.css
 1 file changed, 1 insertion(+)
mensch@aeaskerov:~/hello$
```

Рис. 4.77: Обновление файла в репозитории и коммит

4.16.4 Изменим index.html

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>
```

Рис. 4.78: Редактирование файла index.html

Выполним следующее.

```
mensch@aeaskerov:~/hello$ git add index.html
mensch@aeaskerov:~/hello$ git commit -m "Updated index.html"
[style 1578440] Updated index.html
 1 file changed, 3 insertions(+)
mensch@aeaskerov:~/hello$
```

Рис. 4.79: Обновление файла в репозитории и коммит

4.17 Навигация по веткам

Теперь в нашем проекте есть две ветки.

Выполним следующее.

```
mensch@aeaskerov:~/hello$ git log --all
commit 1578440679b283128ef0bbd1ab3fe76858ce3687 (HEAD -> style)
Author: aeaskerov <iqwertydragoni@gmail.com>
Date:   Wed Feb 12 19:13:55 2025 +0300

    Updated index.html

commit 6345e56c0375c1ed1d8e58a3cbba141e971d23c8
Author: aeaskerov <iqwertydragoni@gmail.com>
Date:   Wed Feb 12 19:12:00 2025 +0300

    Hello uses style.css

commit c2abe60591dc615b3e54b68715c70387d20f2aa0
Author: aeaskerov <iqwertydragoni@gmail.com>
Date:   Wed Feb 12 19:09:37 2025 +0300

    Added css stylesheet

commit c277b5f878513318c817ba8cf514430f47f4ae96 (master)
Author: aeaskerov <iqwertydragoni@gmail.com>
Date:   Wed Feb 12 18:17:56 2025 +0300

    Added index.html.

commit cfb05ab92398429329bd286d13ec33b662cc2c8c
Author: aeaskerov <iqwertydragoni@gmail.com>
Date:   Wed Feb 12 18:11:56 2025 +0300

    Moved hello.html to lib

commit 2293f4f90a699e8c290e133a4b502dbed61a9bcf
Author: aeaskerov <iqwertydragoni@gmail.com>
Date:   Wed Feb 12 18:03:14 2025 +0300

    Add an author/email comment

mensch@aeaskerov:~/hello$ □
```

Рис. 4.80: Журнал

4.17.1 Переключение на ветку master

Используем команду `git checkout` для переключения между ветками.

```
mensch@aeaskerov:~/hello$ git checkout master
Switched to branch 'master'
mensch@aeaskerov:~/hello$ cat lib/hello.html
<!-- Author: Alexander Askerov (1132226538@pfur.ru) -->
<html>
  <head>
  </head>
  <body>
    <h1>Hello World!</h1>
  </body>
</html>
mensch@aeaskerov:~/hello$
```

Рис. 4.81: Переключение ветки

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

4.17.2 Вернёмся к ветке style

Выполним следующее.

```
mensch@aeaskerov:~/hello$ git checkout style
Switched to branch 'style'
mensch@aeaskerov:~/hello$ cat lib/hello.html
<!-- Author: Alexander Askerov (1132226538@pfur.ru) -->
<html>
  <head>
    <link type="text/css" rel="stylesheet" media="all" href="style.css" />
  </head>
  <body>
    <h1>Hello World!</h1>
  </body>
</html>
mensch@aeaskerov:~/hello$
```

Рис. 4.82: Переключение ветки

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

4.18 Изменения в ветке master

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

4.18.1 Создадим файл README в ветке master

Выполним следующее.

```
mensch@aeaskerov:~/hello$ git checkout master
Switched to branch 'master'
mensch@aeaskerov:~/hello$
```

Рис. 4.83: Переключение ветки

Создадим файл README.md.

```
mensch@aeaskerov:~/hello$ echo "This is the Hello World example from the git tutorial." > README.md
mensch@aeaskerov:~/hello$
```

Рис. 4.84: Новый файл README.md

4.19 Сделаем коммит изменений README.md в ветку master

master

Выполним следующее.

```
mensch@aeaskerov:~/hello$ git add README.md
mensch@aeaskerov:~/hello$ git commit -m "Added README"
[master 039d8f2] Added README
 1 file changed, 1 insertion(+)
 create mode 100644 README.md
mensch@aeaskerov:~/hello$
```

Рис. 4.85: Добавление файла в репозиторий и коммит

4.19.1 Просмотр отличающихся веток

4.19.2 Просмотрим текущие ветки

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

```
mensch@aeaskerov:~/hello$ git log --graph --all
* commit 039d8f2b70b76ebe8cabd4104c26dd2be6d796ff (HEAD -> master)
| Author: aeaskerov <iqwertydragoni@gmail.com>
| Date:   Wed Feb 12 19:17:16 2025 +0300
|
|     Added README
|
* commit 1578440679b283128ef0bbd1ab3fe76858ce3687 (style)
| Author: aeaskerov <iqwertydragoni@gmail.com>
| Date:   Wed Feb 12 19:13:55 2025 +0300
|
|     Updated index.html
|
* commit 6345e56c0375cled1d8e58a3cbbal41e971d23c8
| Author: aeaskerov <iqwertydragoni@gmail.com>
| Date:   Wed Feb 12 19:12:00 2025 +0300
|
|     Hello uses style.css
|
* commit c2abe60591dc615b3e54b68715c70387d20f2aa0
| Author: aeaskerov <iqwertydragoni@gmail.com>
| Date:   Wed Feb 12 19:09:37 2025 +0300
|
|     Added css stylesheet
|
* commit c277b5f878513318c817ba8cf514430f47f4ae96
| Author: aeaskerov <iqwertydragoni@gmail.com>
| Date:   Wed Feb 12 18:17:56 2025 +0300
|
|     Added index.html.
|
* commit cfb05ab92398429329bd286d13ec33b662cc2c8c
| Author: aeaskerov <iqwertydragoni@gmail.com>
| Date:   Wed Feb 12 18:11:56 2025 +0300
|
|     Moved hello.html to lib
|
* commit 2293f4f90a699e8c290e133a4b502dbed61a9bcf
| Author: aeaskerov <iqwertydragoni@gmail.com>
mensch@aeaskerov:~/hello$
```

Рис. 4.86: Просмотр веток и их отличий

4.20 Слияние

4.20.1 Слияние веток

Слияние переносит изменения из двух веток в одну. Вернёмся к ветке `style` и сольём `master` со `style`.

```
mensch@aeaskerov:~/hello$ git checkout style
Switched to branch 'style'
mensch@aeaskerov:~/hello$ git merge master
Merge made by the 'ort' strategy.
 README.md | 1 +
 1 file changed, 1 insertion(+)
 create mode 100644 README.md
mensch@aeaskerov:~/hello$
```

Рис. 4.87: Слияние веток

```
mensch@aeaskerov:~/hello$ git log --graph --all
*   commit da1d173cf1f8c2f89b979924346d4983c6ffd4 (HEAD -> style)
|   Merge: 1578440 039d8f2
|   Author: aeaskerov <iqwertydragoni@gmail.com>
|   Date:   Wed Feb 12 19:19:13 2025 +0300
|
|       Merge branch 'master' into style
|
* commit 039d8f2b70b76ebe8cabd4104c26dd2be6d796ff (master)
|   Author: aeaskerov <iqwertydragoni@gmail.com>
|   Date:   Wed Feb 12 19:17:16 2025 +0300
|
|       Added README
|
* commit 1578440679b283128ef0bbd1ab3fe76858ce3687
|   Author: aeaskerov <iqwertydragoni@gmail.com>
|   Date:   Wed Feb 12 19:13:55 2025 +0300
|
|       Updated index.html
|
* commit 6345e56c0375c1ed1d8e58a3cbba141e971d23c8
|   Author: aeaskerov <iqwertydragoni@gmail.com>
|   Date:   Wed Feb 12 19:12:00 2025 +0300
|
|       Hello uses style.css
|
* commit c2abe60591dc615b3e54b68715c70387d20f2aa0
|   Author: aeaskerov <iqwertydragoni@gmail.com>
|   Date:   Wed Feb 12 19:09:37 2025 +0300
|
|       Added css stylesheet
|
* commit c277b5f878513318c817ba8cf514430f47f4ae96
|   Author: aeaskerov <iqwertydragoni@gmail.com>
|   Date:   Wed Feb 12 18:17:56 2025 +0300
|
|       Added index.html.
|
* commit cfb05ab92398429329bd286d13ec33b662cc2c8c
|   Author: aeaskerov <iqwertydragoni@gmail.com>
mensch@aeaskerov:~/hello$
```

Рис. 4.88: Дерево коммитов

4.21 Создание конфликта

4.21.1 Вернёмся в master и создадим конфликт

Вернёмся в ветку master.

```
mensch@aeaskerov:~/hello$ git checkout master
Switched to branch 'master'
mensch@aeaskerov:~/hello$
```

Рис. 4.89: Переключение ветки

Внесём следующие изменения.

```
<!-- Author: Alexander Askerov (1132226538@pfur.ru) -->
<html>
  <head>
    <!-- no style -->
  </head>
  <body>
    <h1>Hello World! Life is great!</h1>
  </body>
</html>
```

Рис. 4.90: Редактирование файла hello.html

Выполним следующее.

```
mensch@aeaskerov:~/hello$ git add lib/hello.html
mensch@aeaskerov:~/hello$ git commit -m 'Life is great'
[master 9f5615b] Life is great
 1 file changed, 2 insertions(+), 1 deletion(-)
mensch@aeaskerov:~/hello$
```

Рис. 4.91: Обновление файла в репозитории и коммит

4.21.2 Просмотр веток

Выполним следующее.

```
mensch@aeaskerov:~/hello$ git log --graph --all
* commit 9f5615b2baebfb8d2ae8a904655a79ad09dd9425 (HEAD -> master)
  Author: aeaskerov <iqwertydragoni@gmail.com>
  Date:   Wed Feb 12 19:24:13 2025 +0300

    Life is great

* commit daf1d173cfdf8c2f89b979924346d4983c6ffd4 (style)
  Merge: 1578440 039d8f2
  Author: aeaskerov <iqwertydragoni@gmail.com>
  Date:   Wed Feb 12 19:19:13 2025 +0300

    Merge branch 'master' into style

* commit 039d8f2b70b76ebe8cabd4104c26dd2be6d796ff
  Author: aeaskerov <iqwertydragoni@gmail.com>
  Date:   Wed Feb 12 19:17:16 2025 +0300

    Added README

* commit 1578440679b283128ef0bbd1ab3fe76858ce3687
  Author: aeaskerov <iqwertydragoni@gmail.com>
  Date:   Wed Feb 12 19:13:55 2025 +0300

    Updated index.html

* commit 6345e56c0375c1ed1d8e58a3cbba141e971d23c8
  Author: aeaskerov <iqwertydragoni@gmail.com>
  Date:   Wed Feb 12 19:12:00 2025 +0300

    Hello uses style.css

* commit c2abe60591dc615b3e54b68715c70387d20f2aa0
  Author: aeaskerov <iqwertydragoni@gmail.com>
  Date:   Wed Feb 12 19:09:37 2025 +0300

    Added css stylesheet

* commit c277b5f878513318c817ba8cf514430f47f4ae96
  Author: aeaskerov <iqwertydragoni@gmail.com>
  Date:   Wed Feb 12 18:17:56 2025 +0300
mensch@aeaskerov:~/hello$
```

Рис. 4.92: Дерево коммитов

После коммита «Added README» ветка master была объединена с веткой style, но в настоящее время в master есть дополнительный коммит, который не был слит со style.

Последнее изменение в master конфликтует с некоторыми изменениями в style. На следующем шаге мы решим этот конфликт.

4.22 Разрешение конфликтов

4.22.1 Слияние master с веткой style

Теперь вернёмся к ветке `style` и попытаемся объединить её с новой веткой `master`.

Выполним следующее.

```
mensch@aeaskerov:~/hello$ git checkout style
Switched to branch 'style'
mensch@aeaskerov:~/hello$ git merge master
Auto-merging lib/hello.html
CONFLICT (content): Merge conflict in lib/hello.html
Automatic merge failed; fix conflicts and then commit the result.
mensch@aeaskerov:~/hello$
```

Рис. 4.93: Конфликт при слиянии веток

Если мы откроем `lib/hello.html`, то увидим следующее.

```
mensch@aeaskerov:~/hello
GNU nano 7.2                                lib/hello.html
<!-- Author: Alexander Askerov (1132226538@pfur.ru) -->
<html>
  <head>
    <<<<< HEAD
      <link type="text/css" rel="stylesheet" media="all" href="style.css" />
    =====
      <!-- no style -->
    >>>>> master
      </head>
      <body>
        <h1>Hello World! Life is great!</h1>
      </body>
    </html>
```

Рис. 4.94: Содержимое `hello.html`

Первый раздел – версия текущей ветки (`style`). Второй раздел – версия ветки `master`.

4.22.2 Решение конфликта

Нам необходимо вручную разрешить конфликт. Внесём изменения в `lib/hello.html` для достижения следующего результата.

```
mensch@aeaskerov: ~/hello
  GNU nano 7.2          lib/hello.html *
<!-- Author: Alexander Askerov (1132226538@pfur.ru) -->
<html>
  <head>
    <link type="text/css" rel="stylesheet" media="all" href="style.css" />
  </head>
  <body>
    <h1>Hello World! Life is great!</h1>
  </body>
</html>
```

Рис. 4.95: Содержимое hello.html

4.22.3 Сделаем коммит решения конфликта

Выполним следующее.

```
mensch@aeaskerov:~/hello$ git add lib/hello.html
mensch@aeaskerov:~/hello$ git commit -m "Merged master fixed conflict."
[style df7b2a0] Merged master fixed conflict.
mensch@aeaskerov:~/hello$
```

Рис. 4.96: Обновление файла в репозитории и коммит

4.22.4 1.22.4 Перебазирование как альтернатива слиянию

Рассмотрим различия между слиянием и перебазированием. Для того, чтобы это сделать, нам нужно вернуться в репозиторий в момент до первого слияния, а затем повторить те же действия, но с использованием перебазирования вместо слияния.

Мы будем использовать команду `reset` для возврата веток к предыдущему состоянию.

4.23 Сброс ветки style

4.23.1 Сброс ветки style

Вернёмся на ветку `style` к точке перед тем, как мы слили её с веткой `master`. Мы можем сбросить ветку к любому коммиту. По сути, это изменение указателя ветки на любую точку дерева коммитов.

В этом случае мы хотим вернуться в ветку style в точку перед слиянием с master. Нам необходимо найти последний коммит перед слиянием.

Выполним следующее.

```
mensch@aeaskerov:~/hello$ git checkout style
Already on 'style'
mensch@aeaskerov:~/hello$
```

Рис. 4.97: Переключение на ветку style

```
mensch@aeaskerov:~/hello$ git log --graph
*   commit df7b2a00326f1473d246063d990a0b399b4f11fe (HEAD -> style)
| \ Merge: da1d17 9f5615b
| | Author: aeaskerov <iqwertydragoni@gmail.com>
| | Date:   Wed Feb 12 19:27:41 2025 +0300
|
|       Merged master fixed conflict.
|
*   commit 9f5615b2baebfb8d2ae8a904655a79ad09dd9425 (master)
| \ Merge: da1d17 9f5615b
| | Author: aeaskerov <iqwertydragoni@gmail.com>
| | Date:   Wed Feb 12 19:24:13 2025 +0300
|
|       Life is great
|
*   commit da1d173cf1f8c2f89b979924346d4983c6ffd4
| \ Merge: 1578440 039d8f2
| | Author: aeaskerov <iqwertydragoni@gmail.com>
| | Date:   Wed Feb 12 19:19:13 2025 +0300
|
|       Merge branch 'master' into style
|
*   commit 039d8f2b70b76ebe8cabd4104c26dd2be6d796ff
| \ Author: aeaskerov <iqwertydragoni@gmail.com>
| | Date:   Wed Feb 12 19:17:16 2025 +0300
|
|       Added README
|
*   commit 1578440679b283128ef0bbdlab3fe76858ce3687
| \ Author: aeaskerov <iqwertydragoni@gmail.com>
| | Date:   Wed Feb 12 19:13:55 2025 +0300
|
|       Updated index.html
|
*   commit 6345e56c0375cled1d8e58a3cbba141e971d23c8
| \ Author: aeaskerov <iqwertydragoni@gmail.com>
| | Date:   Wed Feb 12 19:12:00 2025 +0300
|
|       Hello uses style.css
|
*   commit c2abe60591dc615b3e54b68715c70387d20f2aa0
| \ Author: aeaskerov <iqwertydragoni@gmail.com>
| | Date:   Wed Feb 12 19:09:37 2025 +0300
mensch@aeaskerov:~/hello$
```

Рис. 4.98: Дерево коммитов

Мы видим, что коммит «Updated index.html» был последним на ветке style перед слиянием. Сбросим ветку style к этому коммиту.

```
mensch@aeaskerov:~/hello$ git reset --hard 1578440
HEAD is now at 1578440 Updated index.html
mensch@aeaskerov:~/hello$
```

Рис. 4.99: Перемещение к другому коммиту

4.23.2 Проверим ветку

Поищем лог ветки style. У нас в истории больше нет коммитов слияний.

```
mensch@aeaskerov:~/hello$ git log --graph --all
* commit 9f5615b2baebfb8d2ae8a904655a79ad09dd9425 (master)
| Author: aeaskerov <iqwertydragoni@gmail.com>
| Date:   Wed Feb 12 19:24:13 2025 +0300
|
|     Life is great
|
* commit 039d8f2b70b76ebe8cabd4104c26dd2be6d796ff
| Author: aeaskerov <iqwertydragoni@gmail.com>
| Date:   Wed Feb 12 19:17:16 2025 +0300
|
|     Added README
|
* commit 1578440679b283128ef0bbd1ab3fe76858ce3687 (HEAD -> style)
| Author: aeaskerov <iqwertydragoni@gmail.com>
| Date:   Wed Feb 12 19:13:55 2025 +0300
|
|     Updated index.html
|
* commit 6345e56c0375c1ed1d8e58a3cbb141e971d23c8
| Author: aeaskerov <iqwertydragoni@gmail.com>
| Date:   Wed Feb 12 19:12:00 2025 +0300
|
|     Hello uses style.css
|
* commit c2abe60591dc615b3e54b68715c70387d20f2aa0
| Author: aeaskerov <iqwertydragoni@gmail.com>
| Date:   Wed Feb 12 19:09:37 2025 +0300
|
|     Added css stylesheet
|
* commit c277b5f878513318c817ba8cf514430f47f4ae96
| Author: aeaskerov <iqwertydragoni@gmail.com>
| Date:   Wed Feb 12 18:17:56 2025 +0300
|
|     Added index.html.
|
* commit cfb05ab92398429329bd286d13ec33b662cc2c8c
| Author: aeaskerov <iqwertydragoni@gmail.com>
| Date:   Wed Feb 12 18:11:56 2025 +0300
|
|     Moved hello.html to lib
|
mensch@aeaskerov:~/hello$
```

Рис. 4.100: Дерево коммитов

4.24 Сброс ветки master

4.24.1 Сброс ветки master

Добавив интерактивный режим в ветку master, мы внесли изменения, конфликтующие с изменениями в ветке style. Вернёмся в ветку master в точку перед внесением конфликтующих изменений. Это позволяет нам продемон-

стрировать работу команды git rebase, не беспокоясь о конфликтах.

```
mensch@aeaskerov:~/hello$ git checkout master
Switched to branch 'master'
mensch@aeaskerov:~/hello$
```

Рис. 4.101: Переключение на ветку master

```
mensch@aeaskerov:~/hello$ git log --graph
* commit 9f5615b2baebfb8d2ae8a904655a79ad09dd9425 (HEAD -> master)
  Author: aeaskerov <iqwertydragoni@gmail.com>
  Date:   Wed Feb 12 19:24:13 2025 +0300

    Life is great

* commit 039d8f2b70b76ebe8cabd4104c26dd2be6d796ff
  Author: aeaskerov <iqwertydragoni@gmail.com>
  Date:   Wed Feb 12 19:17:16 2025 +0300

    Added README

* commit c277b5f878513318c817ba8cf514430f47f4ae96
  Author: aeaskerov <iqwertydragoni@gmail.com>
  Date:   Wed Feb 12 18:17:56 2025 +0300

    Added index.html.

* commit cfb05ab92398429329bd286d13ec33b662cc2c8c
  Author: aeaskerov <iqwertydragoni@gmail.com>
  Date:   Wed Feb 12 18:11:56 2025 +0300

    Moved hello.html to lib

* commit 2293f4f90a699e8c290e133a4b502dbed61a9bcf
  Author: aeaskerov <iqwertydragoni@gmail.com>
  Date:   Wed Feb 12 18:03:14 2025 +0300

    Add an author/email comment

* commit 6d6170e41f31010a5324e8e1294bc5e1bfc29dd2 (tag: v1)
  Author: aeaskerov <iqwertydragoni@gmail.com>
  Date:   Wed Feb 12 17:07:01 2025 +0300

    Added HTML header

* commit 37e4800d3dc24b1a3523ef9c3b72a20b0ba19f0c (tag: v1-beta)
  Author: aeaskerov <iqwertydragoni@gmail.com>
  Date:   Wed Feb 12 17:06:02 2025 +0300

    Added standard HTML page tags

* commit 0c8cb2a3d946388c341fa423d0cac83f400b4d9e
mensch@aeaskerov:~/hello$
```

Рис. 4.102: Дерево коммитов

Коммит «Added README» идёт непосредственно перед коммитом конфлик-

тующего интерактивного режима. Мы сбросим ветку master к коммиту «Added README».

```
mensch@aeaskerov:~/hello$ git reset --hard 039d8f2
HEAD is now at 039d8f2 Added README
mensch@aeaskerov:~/hello$
```

Рис. 4.103: Перемещение к другому коммиту

```
mensch@aeaskerov:~/hello$ git log --graph --all
* commit 039d8f2b70b76ebe8cabd4104c26dd2be6d796ff (HEAD -> master)
| Author: aeaskerov <iqwertydragoni@gmail.com>
| Date:   Wed Feb 12 19:17:16 2025 +0300
|
|       Added README
|
* commit 1578440679b283128ef0bbd1ab3fe76858ce3687 (style)
| Author: aeaskerov <iqwertydragoni@gmail.com>
| Date:   Wed Feb 12 19:13:55 2025 +0300
|
|       Updated index.html
|
* commit 6345e56c0375c1ed1d8e58a3cbb141e971d23c8
| Author: aeaskerov <iqwertydragoni@gmail.com>
| Date:   Wed Feb 12 19:12:00 2025 +0300
|
|       Hello uses style.css
|
* commit c2abe60591dc615b3e54b68715c70387d20f2aa0
| Author: aeaskerov <iqwertydragoni@gmail.com>
| Date:   Wed Feb 12 19:09:37 2025 +0300
|
|       Added css stylesheet
|
* commit c277b5f878513318c817ba8cf514430f47f4ae96
| Author: aeaskerov <iqwertydragoni@gmail.com>
| Date:   Wed Feb 12 18:17:56 2025 +0300
|
|       Added index.html.
|
* commit cfb05ab92398429329bd286d13ec33b662cc2c8c
| Author: aeaskerov <iqwertydragoni@gmail.com>
| Date:   Wed Feb 12 18:11:56 2025 +0300
|
|       Moved hello.html to lib
|
* commit 2293f4f90a699e8c290e133a4b502dbed61a9bcf
| Author: aeaskerov <iqwertydragoni@gmail.com>
| Date:   Wed Feb 12 18:03:14 2025 +0300
|
|       Add an author/email comment
|
* commit 6d6170e41f31010a5324e8e1294bc5e1bfc29dd2 (tag: v1)
| Author: aeaskerov <iqwertydragoni@gmail.com>
mensch@aeaskerov:~/hello$
```

Рис. 4.104: Дерево коммитов

Лог выглядит, как будто репозиторий был перемотан назад во времени к точке до какого-либо слияния.

4.25 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 используем команду git rebase вместо слияния.

```
mensch@aeaskerov:~/hello$ git checkout style
Switched to branch 'style'
mensch@aeaskerov:~/hello$ git rebase master
Successfully rebased and updated refs/heads/style.
mensch@aeaskerov:~/hello$
```

Рис. 4.105: Перебазирование

```
mensch@aeaskerov:~/hello$ git log --graph
* commit 6859d193adfe6e243bc26cd0c975087f60da7fc6 (HEAD -> style)
  Author: aeaskerov <iqwertydragoni@gmail.com>
  Date:   Wed Feb 12 19:13:55 2025 +0300

    Updated index.html

* commit 48fa2a5cbc5d0cce747ec10eaa974b2f00497ffc
  Author: aeaskerov <iqwertydragoni@gmail.com>
  Date:   Wed Feb 12 19:12:00 2025 +0300

    Hello uses style.css

* commit fb48c3f08f847fe8a27255d8b09443f8728a676a
  Author: aeaskerov <iqwertydragoni@gmail.com>
  Date:   Wed Feb 12 19:09:37 2025 +0300

    Added css stylesheet

* commit 039d8f2b70b76ebe8cabd4104c26dd2be6d796ff (master)
  Author: aeaskerov <iqwertydragoni@gmail.com>
  Date:   Wed Feb 12 19:17:16 2025 +0300

    Added README

* commit c277b5f878513318c817ba8cf514430f47f4ae96
  Author: aeaskerov <iqwertydragoni@gmail.com>
  Date:   Wed Feb 12 18:17:56 2025 +0300

    Added index.html.

* commit cfb05ab92398429329bd286d13ec33b662cc2c8c
  Author: aeaskerov <iqwertydragoni@gmail.com>
  Date:   Wed Feb 12 18:11:56 2025 +0300

    Moved hello.html to lib

* commit 2293f4f90a699e8c290e133a4b502dbed61a9bcf
  Author: aeaskerov <iqwertydragoni@gmail.com>
  Date:   Wed Feb 12 18:03:14 2025 +0300

    Add an author/email comment

* commit 6d6170e41f31010a5324e8e1294bc5e1bfc29dd2 (tag: v1)
  Author: aeaskerov <iqwertydragoni@gmail.com>
  Date:   Wed Feb 12 17:07:01 2025 +0300
mensch@aeaskerov:~/hello$
```

Рис. 4.106: Журнал

4.25.1 Слияние VS перебазирование

Конечный результат перебазирования очень похож на результат слияния. Ветка `style` в настоящее время содержит все свои изменения, а также все изменения ветки `master`. Однако, дерево коммитов значительно отличается. Дерево

коммитов ветки style было переписано таким образом, что ветка master является частью истории коммитов. Это делает цепь коммитов линейной и гораздо более читабельной.

4.26 Слияние в ветку master

Мы поддерживали соответствие ветки style с веткой master (с помощью rebase), теперь же сольём изменения style в ветку master.

4.26.1 Слияние style в master

Выполним следующее.

```
mensch@aeaskerov:~/hello$ git checkout master
Switched to branch 'master'
mensch@aeaskerov:~/hello$ git merge style
Updating 039d8f2..6859d19
Fast-forward
 index.html      | 3 +++
 lib/hello.html  | 1 +
 lib/style.css   | 3 +++
 3 files changed, 7 insertions(+)
 create mode 100644 lib/style.css
mensch@aeaskerov:~/hello$
```

Рис. 4.107: Слияние веток

Поскольку последний коммит ветки master прямо предшествует последнему коммиту ветки style, git может выполнить ускоренное слияние-перемотку. При быстрой перемотке вперёд git просто передвигает указатель вперёд, таким образом указывая на тот же коммит, что и ветка style.

При быстрой перемотке конфликтов быть не может.

4.26.2 Просмотрим логи

Выполним следующее.

```
mensch@aeaskerov:~/hello$ git log
commit 6859d193adfe6e243bc26cd0c975087f60da7fc6 (HEAD -> master, style)
Author: aeaskerov <iqwertydragoni@gmail.com>
Date:   Wed Feb 12 19:13:55 2025 +0300

    Updated index.html

commit 48fa2a5cbc5d0cce747ec10eaa974b2f00497ff
Author: aeaskerov <iqwertydragoni@gmail.com>
Date:   Wed Feb 12 19:12:00 2025 +0300

    Hello uses style.css

commit fb48c3f08f847fe8a27255d8b09443f8728a676a
Author: aeaskerov <iqwertydragoni@gmail.com>
Date:   Wed Feb 12 19:09:37 2025 +0300

    Added css stylesheet

commit 039d8f2b70b76ebe8cabd4104c26dd2be6d796ff
Author: aeaskerov <iqwertydragoni@gmail.com>
Date:   Wed Feb 12 19:17:16 2025 +0300

    Added README

commit c277b5f878513318c817ba8cf514430f47f4ae96
Author: aeaskerov <iqwertydragoni@gmail.com>
Date:   Wed Feb 12 18:17:56 2025 +0300

    Added index.html.

commit cfb05ab92398429329bd286d13ec33b662cc2c8c
Author: aeaskerov <iqwertydragoni@gmail.com>
Date:   Wed Feb 12 18:11:56 2025 +0300

    Moved hello.html to lib

commit 2293f4f90a699e8c290e133a4b502dbed61a9bcf
Author: aeaskerov <iqwertydragoni@gmail.com>
Date:   Wed Feb 12 18:03:14 2025 +0300

    Add an author/email comment

commit 6d6170e41f31010a5324e8e1294bc5e1bfc29dd2 (tag: v1)
Author: aeaskerov <iqwertydragoni@gmail.com>
mensch@aeaskerov:~/hello$
```

Рис. 4.108: Журнал

Теперь ветки `style` и `master` идентичны.

4.27 Клонирование репозиториев

4.27.1 Перейдём в рабочий каталог

Перейдём в рабочий каталог и сделаем клон нашего репозитория `hello`.

```
mensch@aeaskerov:~/hello$ cd ..
mensch@aeaskerov:~$ pwd
/home/mensch
mensch@aeaskerov:~$ ls
Desktop Documents Downloads hello Music Pictures Public Templates Videos
mensch@aeaskerov:~$
```

Рис. 4.109: Переход в рабочий каталог

Сейчас мы находимся в рабочем каталоге. Здесь есть единственный репозиторий под названием «hello».

4.27.2 Создадим клон репозитория hello

Создадим клон репозитория.

```
mensch@aeaskerov:~$ git clone hello cloned_hello
Cloning into 'cloned_hello'...
done.
mensch@aeaskerov:~$ ls
cloned_hello  Documents  hello  Pictures  Templates
Desktop       Downloads  Music   Public    Videos
mensch@aeaskerov:~$
```

Рис. 4.110: Клон репозитория

В нашем рабочем каталоге теперь есть два репозитория: оригинальный репозиторий «hello» и клонированный репозиторий «cloned_hello».

4.28 Просмотр клонированного репозитория

4.28.1 Взглянем на клонированный репозиторий

```
mensch@aeaskerov:~$ cd cloned_hello
mensch@aeaskerov:~/cloned_hello$ ls
index.html  lib  README.md
mensch@aeaskerov:~/cloned_hello$
```

Рис. 4.111: Содержимое клонированного репозитория

Мы видим список всех файлов на верхнем уровне оригинального репозитория README.md, index.html и lib.

4.28.2 Просмотрим историю репозитория

Выполним следующее.

```
mensch@aeaskerov:~/cloned_hello$ git log --all
commit 6859d193adfe0e243bc26cd0c975087f60da7fc6 (HEAD -> master, origin/style, origin/master, origin/HEAD)
Author: aeaskerov <iqwertydragoni@gmail.com>
Date:   Wed Feb 12 19:13:55 2025 +0300

    Updated index.html

commit 48fa2a5cbc5d0cce747ec10eaa974b2f0049fffc
Author: aeaskerov <iqwertydragoni@gmail.com>
Date:   Wed Feb 12 19:12:00 2025 +0300

    Hello uses style.css

commit fb48c3f08f847fe8a27255d8b09443f8728a676a
Author: aeaskerov <iqwertydragoni@gmail.com>
Date:   Wed Feb 12 19:09:37 2025 +0300

    Added css stylesheet

commit 039d8f2b70b76ebe8cabd4104c26dd2be6d796ff
Author: aeaskerov <iqwertydragoni@gmail.com>
Date:   Wed Feb 12 19:17:16 2025 +0300

    Added README

commit c277b5f878513318c817ba8cf514430f47f4ae96
Author: aeaskerov <iqwertydragoni@gmail.com>
Date:   Wed Feb 12 18:17:56 2025 +0300

    Added index.html.

commit cfb05ab92398429329bd286d13ec33b662cc2c8c
Author: aeaskerov <iqwertydragoni@gmail.com>
Date:   Wed Feb 12 18:11:56 2025 +0300

    Moved hello.html to lib

commit 2293f4f90a699e8c290e133a4b502dbed61a9bcf
Author: aeaskerov <iqwertydragoni@gmail.com>
Date:   Wed Feb 12 18:03:14 2025 +0300

    Add an author/email comment

commit 6d6170e41f31010a5324e8e1294bc5e1bfc29dd2 (tag: v1)
mensch@aeaskerov:~/cloned_hello$
```

Рис. 4.112: Журнал

Мы видим список всех коммитов в новый репозиторий, и он (более или менее) совпадает с историей коммитов в оригинальном репозитории. Единственная разница должна быть в названиях веток.

4.28.3 Удалённые ветки

Мы видим ветку master (HEAD) в списке истории. Мы также видим ветки со странными именами (origin/master, origin/style и origin/HEAD).

4.29 Что такое origin?

Выполним следующее.

```
mensch@aeaskerov:~/cloned_hello$ git remote  
origin  
mensch@aeaskerov:~/cloned_hello$
```

Рис. 4.113: Удалённый репозиторий

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

Выполним следующее.

```
mensch@aeaskerov:~/cloned_hello$ git remote show origin  
* remote origin  
  Fetch URL: /home/mensch/hello  
  Push URL: /home/mensch/hello  
  HEAD branch: master  
  Remote branches:  
    master tracked  
    style tracked  
  Local branch configured for 'git pull':  
    master merges with remote master  
  Local ref configured for 'git push':  
    master pushes to master (up to date)  
mensch@aeaskerov:~/cloned_hello$
```

Рис. 4.114: Подробная информация об удалённом репозитории

4.30 Удалённые ветки

Давайте посмотрим на ветки, доступные в нашем клонированном репозитории.

```
mensch@aeaskerov:~/cloned_hello$ git branch
* master
mensch@aeaskerov:~/cloned_hello$
```

Рис. 4.115: Ветки в репозитории

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

4.30.1 Список удалённых веток

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

```
mensch@aeaskerov:~/cloned_hello$ git branch -a
* master
  remotes/origin/HEAD -> origin/master
  remotes/origin/master
  remotes/origin/style
mensch@aeaskerov:~/cloned_hello$
```

Рис. 4.116: Просмотр всех веток

Git выводит все коммиты в оригинальный репозиторий, но ветки в удалённом репозитории не рассматриваются как локальные. Если мы хотим собственную ветку `style`, мы должны сами её создать. Скоро мы увидим, как это делается.

4.31 Изменение оригинального репозитория

Внесём некоторые изменения в оригинальный репозиторий, чтобы затем попытаться извлечь и слить изменения из удалённой ветки в текущую.

4.31.1 Внесём изменения в оригиналальный репозиторий hello

Выполним следующее.

```
mensch@aeaskerov:~/cloned_hello$ cd ../hello
mensch@aeaskerov:~/hello$
```

Рис. 4.117: Перемещение

Примечание: Сейчас мы находимся в репозитории hello

Внесём следующие изменения в файл README.md.

```
This is the Hello World example from the git tutorial.
This is the Hello World example from the git tutorial.
```

Рис. 4.118: Редактируем README

Выполним следующее.

```
mensch@aeaskerov:~/hello$ git add README.md
mensch@aeaskerov:~/hello$ git commit -m "Changed README in original repo"
[master d30f046] Changed README in original repo
 1 file changed, 1 insertion(+)
mensch@aeaskerov:~/hello$
```

Рис. 4.119: Обновление файла в репозитории и коммит

Теперь в оригиналльном репозитории есть более поздние изменения, которых нет в клонированной версии. Далее мы извлечём и сольём эти изменения в клонированный репозиторий.

4.31.2 Извлечение изменений

Научимся извлекать изменения из удалённого репозитория.

```
mensch@aeaskerov:~/hello$ cd ../cloned_hello
mensch@aeaskerov:~/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), 364 bytes | 364.00 KiB/s, done.
From /home/mensch/hello
  6859d19..d30f046  master      -> origin/master
mensch@aeaskerov:~/cloned_hello$
```

Рис. 4.120: Перекачиваем файлы

```
mensch@aeaskerov:~/cloned_hello$ git log --all
commit d30f04644513b186f5bd416318ec685e86f557cb (origin/master, origin/HEAD)
Author: aeaskerov <iqwertydragoni@gmail.com>
Date:   Wed Feb 12 19:54:04 2025 +0300

    Changed README in original repo

commit 6859d193adfe6e243bc26cd0c975087f60da7fc6 (HEAD -> master, origin/style)
Author: aeaskerov <iqwertydragoni@gmail.com>
Date:   Wed Feb 12 19:13:55 2025 +0300

    Updated index.html

commit 48fa2a5cbc5d0cce747ec10eaa974b2f00497ffc
Author: aeaskerov <iqwertydragoni@gmail.com>
Date:   Wed Feb 12 19:12:00 2025 +0300

    Hello uses style.css

commit fb48c3f08f847fe8a27255d8b09443f8728a676a
Author: aeaskerov <iqwertydragoni@gmail.com>
Date:   Wed Feb 12 19:09:37 2025 +0300

    Added css stylesheet

commit 039d8f2b70b76ebe8cabd4104c26dd2be6d796ff
Author: aeaskerov <iqwertydragoni@gmail.com>
Date:   Wed Feb 12 19:17:16 2025 +0300

    Added README

commit c277b5f878513318c817ba8cf514430f47f4ae96
Author: aeaskerov <iqwertydragoni@gmail.com>
Date:   Wed Feb 12 18:17:56 2025 +0300

    Added index.html.

commit cfb05ab92398429329bd286d13ec33b662cc2c8c
Author: aeaskerov <iqwertydragoni@gmail.com>
Date:   Wed Feb 12 18:11:56 2025 +0300

    Moved hello.html to lib

commit 2293f4f90a699e8c290e133a4b502dbed61a9bcf
Author: aeaskerov <iqwertydragoni@gmail.com>
Date:   Wed Feb 12 18:03:14 2025 +0300
mensch@aeaskerov:~/cloned_hello$
```

Рис. 4.121: Журнал

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.

В истории выше найдём коммит «Changed README in original repo». Обратим внимание, что коммит включает в себя коммиты «origin/master» и «origin/HEAD».

Теперь посмотрим на коммит «Updated index.html». Мы увидим, что локальная ветка master указывает на этот коммит, а не на новый коммит, который мы только что извлекли.

Выводом является то, что команда `git fetch` будет извлекать новые коммиты из удалённого репозитория, но не будет сливать их с нашими наработками в локальных ветках.

4.31.3 Проверим README.md

Мы можем продемонстрировать, что клонированный файл README.md не изменился.

```
mensch@aeaskerov:~/cloned_hello$ cat README.md
This is the Hello World example from the git tutorial.
mensch@aeaskerov:~/cloned_hello$
```

Рис. 4.122: Содержимое README

4.32 Слияние извлечённых изменений

4.32.1 Сольём извлечённые изменения в локальную ветку master

Выполним следующее.

```
mensch@aeaskerov:~/cloned_hello$ git merge origin/master
Updating 6859d19..d30f046
Fast-forward
 README.md | 1 +
 1 file changed, 1 insertion(+)
mensch@aeaskerov:~/cloned_hello$
```

Рис. 4.123: Слияние веток

4.32.2 Ещё раз проверим файл README.md

Сейчас мы увидим изменения.

```
mensch@aeaskerov:~/cloned_hello$ cat README.md
This is the Hello World example from the git tutorial.
This is the Hello World example from the git tutorial.
mensch@aeaskerov:~/cloned_hello$
```

Рис. 4.124: Версия README-файла

Хотя команда git fetch не сливает изменения, мы можем вручную слить изменения из удалённого репозитория.

Теперь давайте рассмотрим объединение fetch и merge в одну команду.

```
mensch@aeaskerov:~/cloned_hello$ git pull
Already up to date.
mensch@aeaskerov:~/cloned_hello$
```

Рис. 4.125: git pull

4.33 Добавление ветки наблюдения

Ветки, которые начинаются с remotes/origin являются ветками оригинального репозитория. Обратим внимание, что у нас больше нет ветки под названием style, но система контроля версий знает, что в оригинальном репозитории ветка style была.

4.33.1 Добавим локальную ветку, которая отслеживает удалённую ветку

Выполним следующее.

```
mensch@aeaskerov:~/cloned_hello$ git branch --track style origin/style
branch 'style' set up to track 'origin/style'.
mensch@aeaskerov:~/cloned_hello$ git branch -a
* master
  style
    remotes/origin/HEAD -> origin/master
    remotes/origin/master
    remotes/origin/style
mensch@aeaskerov:~/cloned_hello$ git log --max-count=2
commit d30f04644513b186f5bd416318ec685e86f557cb (HEAD -> master, origin/master, origin/HEAD)
)
Author: aeaskerov <iqwertydragoni@gmail.com>
Date:   Wed Feb 12 19:54:04 2025 +0300

    Changed README in original repo

commit 6859d193adfe6e243bc26cd0c975087f60da7fc6 (origin/style, style)
Author: aeaskerov <iqwertydragoni@gmail.com>
Date:   Wed Feb 12 19:13:55 2025 +0300

    Updated index.html
mensch@aeaskerov:~/cloned_hello$
```

Рис. 4.126: Отслеживание удалённой ветки

Теперь мы можем видеть ветку `style` в списке веток и логе.

4.34 Чистые репозитории

Чистые репозитории (без рабочих каталогов) обычно используются для расшаривания. Обычный git-репозиторий подразумевает, что мы будем использовать его как рабочую директорию, поэтому вместе с файлами проекта в актуальной версии, git хранит все служебные, «чисто-репозиториевые» файлы в поддиректории `.git`. В удалённых 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».

4.35 Создадим чистый репозиторий

```
mensch@aeaskerov:~/cloned_hello$ cd ..
mensch@aeaskerov:~$ git clone --bare hello hello.git
Cloning into bare repository 'hello.git'...
done.
mensch@aeaskerov:~$ ls hello.git
branches config description HEAD hooks info objects packed-refs refs
mensch@aeaskerov:~$
```

Рис. 4.127: Чистый репозиторий

Сейчас мы находимся в рабочем каталоге.

Как правило, репозитории, оканчивающиеся на `.git` являются чистыми репозиториями. Мы видим, что в репозитории `hello.git` нет рабочего каталога. Потому, это есть не что иное, как каталог `.git` нечистого репозитория.

4.36 Добавление удалённого репозитория

Давайте добавим репозиторий `hello.git` к нашему оригинальному репозиторию.

```
mensch@aeaskerov:~$ cd hello
mensch@aeaskerov:~/hello$ git remote add shared ../hello.git
mensch@aeaskerov:~/hello$
```

Рис. 4.128: Добавление репозитория `hello.git`

4.37 Отправка изменений

Так как чистые репозитории, как правило, расшариваются на каком-нибудь сетевом сервере, нам необходимо отправить наши изменения в другие репозитории. Начнём с создания изменения для отправки.

Отредактируем файл `README.md` и сделаем коммит.

```
mensch@aeaskerov: ~/hello
GNU nano 7.2                               README.md *
This is the Hello World example from the git tutorial.
This is the Hello World example from the git tutorial.
(Changed in the original and pushed to shared)
```

Рис. 4.129: Редактирование README.md

```
mensch@aeaskerov:~/hello$ git checkout master
M      README.md
Already on 'master'
mensch@aeaskerov:~/hello$ git add README.md
mensch@aeaskerov:~/hello$ git commit -m "Added shared comment to readme"
[master 528ffdc] Added shared comment to readme
 1 file changed, 1 insertion(+)
mensch@aeaskerov:~/hello$
```

Рис. 4.130: Коммит

Теперь отправим изменения в общий репозиторий.

```
mensch@aeaskerov:~/hello$ git push shared master
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 4 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 393 bytes | 393.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
To ../hello.git
  d30f046..528ffdc  master -> master
mensch@aeaskerov:~/hello$
```

Рис. 4.131: Отправление изменений в общий репозиторий

Общим называется репозиторий, получающий отправленные нами изменения.

4.38 Извлечение общих изменений

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

```
mensch@aeaskerov:~/hello$ cd ../cloned_hello
mensch@aeaskerov:~/cloned_hello$
```

Рис. 4.132: Перемещение

```
mensch@aeaskerov:~/cloned_hello$ git remote add shared ..../hello.git
mensch@aeaskerov:~/cloned_hello$ git branch --track shared master
branch 'shared' set up to track 'master'.
mensch@aeaskerov:~/cloned_hello$ git pull shared master
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), 373 bytes | 373.00 KiB/s, done.
From ..../hello
 * branch           master      -> FETCH HEAD
 * [new branch]     master      -> shared/master
Updating d30f046..528ffdc
Fast-forward
 README.md | 1 +
 1 file changed, 1 insertion(+)
mensch@aeaskerov:~/cloned_hello$ cat README.md
This is the Hello World example from the git tutorial.
This is the Hello World example from the git tutorial.
(Changed in the original and pushed to shared)
mensch@aeaskerov:~/cloned_hello$
```

Рис. 4.133: Извлечение общих изменений

5 Выводы

Получены навыки работы с системой git.

Список литературы

1. Git [Электронный ресурс]. URL: <https://ru.wikipedia.org/wiki/Git>.