

Отчет по лабораторной работе №1

Нефедова Наталия Николаевна

Содержание

1 Цель работы	7
2 Подготовка	8
3 Выполнение лабораторной работы	9
4 1.1.2 Параметры установки окончаний строки	10
5 1.2 Создание проекта	11
6 1.3 Внесение изменений	13
7 1.4 Индексация изменений	14
8 1.4.1 Коммит изменений	15
9 1.4.3 История	18
10 Получение старых версий	19
11 1.4.5 Создание тегов версий	21
12 1.4.6 Переключение по имени тега	22
13 1.4.7 Просмотр тегов с помощью команды tag	23
14 1.5 Отмена локальных изменений (до индексации)	24
15 1.6 Отмена проиндексированных изменений (перед коммитом)	26
16 1.7 Отмена коммитов	28
17 1.8.3 Для начала отметьте эту ветку	31
18 1.9 Удаление тега oops	33
19 1.10 Внесение изменений в коммиты	34
20 1.11 Перемещение файлов	36

21 1.12 Второй способ перемещения файлов	37
22 1.14 Git внутри: Каталог .git	39
23 1.16 Создание ветки	42
24 1.20 Слияние	46
25 1.21 Создание конфликта	47
26 1.22 Разрешение конфликтов	48
27 1.23 Сброс ветки style	49
28 1.24 Сброс ветки master	50
29 1.25 Перебазирование	51
30 1.27 Клонирование репозиторием	54
31 1.28 Просмотр колонизированного репозитория	55
32 1.29 Что такое origin?	57
33 1.30 Удаленные ветки	58
34 1.31 Внесите изменения в оригиналый репозиторий hello	59
35 1.32 Слияние извлеченных изменений	61
36 1.33 Добавление ветки наблюдения	62
37 1.34 Чистые репозитории	63
38 1.35 Создайте чистый репозиторий	64
39 1.36 Добавление удаленного репозитория	65
40 1.37 Отправка изменений	66
41 1.38 Извлечение общих изменений	67
42 Выводы	68
Список литературы	69

Список иллюстраций

5.1 Создайте страницу «Hello, World»	11
5.2 Добавление файла в репозиторий	11
6.1 Внесение изменений	13
6.2 git status	13
7.1 Индексация изменений	14
8.1 Коммит изменений	15
8.2 git status	16
8.3 коммит второго изменения	17
9.1 git log	18
12.1 Переключение по имени тега	22
13.1 git log master –all	23
14.1 Изменим hello.html	24
14.2 Отмена изменений в рабочем каталоге	25
15.1 Состояние	26
15.2 Переключитесь на версию коммита	27
16.1 Измените файл и сделайте коммит	28
16.2 Сделайте коммит с новыми изменениями	29
17.1 Сброс коммитов к предшествующим коммиту Oops	31
17.2 ошибочные коммиты не исчезли	32
18.1 Удаление тега oops	33
19.1 Внесение изменений в коммиты	34
19.2 Добавим кормит с комментарием	34
19.3 Добавим email	35
19.4 git log	35
20.1 Перемещение файла файл hello.html в каталог lib	36
21.1 Добавление index.html	37
21.2 страница hello в маленьком окошке	38

22.1 Один из каталогов с именем из 2 букв.	39
22.2 Ветки и теги	40
22.3 cat .git/HEAD	40
22.4 git log –max-count=1	40
22.5 Вывод последнего коммита с помощью SHA1	41
22.6 Поиск дерева	41
23.1 Создайте ветку	42
23.2 Добавьте файл стилей style.css	42
23.3 Обновим hello.html	43
23.4 Сделаем комет изменений	43
23.5 Навигация по веткам	43
23.6 git checkout - на ветку master	44
23.7 Вернемся к ветке style	44
23.8 Сделаем коммит изменений README.md	44
23.9 Просмотр отличающихся веток	45
24.1 Слияние веток	46
25.1 Создание конфликта	47
26.1 Решение конфликта	48
27.1 Сброс ветки style	49
28.1 Сброс ветки master	50
29.1 Перебазирование	51
29.2 Перебазирование 2	51
29.3 Слияние в ветку master	52
29.4 Просмотр логов	53
30.1 Клонирование репозитория	54
31.1 Клонированный репозиторий	55
31.2 Все логи	56
32.1 Удаленные репозитории	57
33.1 git branch - просмотр локальных веток	58
33.2 git branch -a просмотр всех веток	58
34.1 Внесите изменения в оригинальный репозиторий	59
34.2 Извлечение изменений	60
34.3 README.md не изменился	60
35.1 Слияние извлеченных изменений	61

36.1 Добавим локальную ветку	62
38.1 Создайте чистый репозиторий	64
39.1 Добавление удаленного репозитория	65
40.1 Отправка изменений в репозиторий	66
41.1 Извлечение общих изменений	67

1 Цель работы

Научиться работать с git, провести базовую установку

2 Подготовка

1.1.1 Установка имени и электронной почты Если вы никогда ранее не использовали git, для начала вам необходимо осуществить установку. Выполните следующие команды, чтобы git узнал ваше имя и электронную почту. Если git уже установлен, можете переходить к разделу окончания строк.[1]

3 Выполнение лабораторной работы

4 1.1.2 Параметры установки окончаний строк

Для пользователей Unix/Mac:[1]

```
git config -global core.autocrlf input git config -global core.safecrlf true
```

1.1.3 Установка отображения unicode По умолчанию, git будет печатать не-ASCII символов в именах файлов в виде восьмеричных последовательностей nnn. Что бы избежать нечитаемых строк, установите соответствующий флаг.

```
git config -global core.quotepath off
```

5 1.2 Создание проекта

1.2.1 Создайте страницу «Hello, World» Начните работу в пустом рабочем каталоге с создания пустого каталога с именем hello, затем войдите в него и создайте там файл с именем hello.html.

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

```
[nata@MacBook-Air-Nata ~ % cd Desktop
[nata@MacBook-Air-Nata Desktop % cd Machine_learning
[nata@MacBook-Air-Nata Machine_learning % mkdir hello
[nata@MacBook-Air-Nata Machine_learning % cd hello
[nata@MacBook-Air-Nata hello % touch hello.html
[nata@MacBook-Air-Nata hello % echo "Hello, World!" > hello.html
```

Рис. 5.1: Создайте страницу «Hello, World»

1.2.2 Создание репозитория Чтобы создать git репозиторий из этого каталога, выполните команду git init.

```
git init
```

1.2.3 Добавление файла в репозиторий Добавим файл в репозиторий.

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

```
nata /Users/nata/Desktop/git% git add hello.html
nata /Users/nata/Desktop/git% git commit -m "Initial Commit"
[master (root-commit) 0eca96f] Initial Commit
 1 file changed, 0 insertions(+), 0 deletions(-)
   create mode 100644 hello.html
nata /Users/nata/Desktop/git%
```

Рис. 5.2: Добавление файла в репозиторий

1.2.4 Проверка состояния репозитория Используйте команду git status, чтобы проверить текущее состояние репозитория.

```
git status
[nata /Users/nata/Desktop/git% git status
On branch master
nothing to commit, working tree clean
```

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

6 1.3 Внесение изменений

1.3.1 Измените страницу «Hello, World» Добавим кое-какие HTML-теги к нашему приветствию. Измените содержимое файла hello.html на:

Hello, World!

Проверьте состояние рабочего каталога.



```
hello.html - .../Desktop/g
<> hello.html •
1 <h1>Hello, World!</h1>
```

Рис. 6.1: Внесение изменений

git status git знает, что файл hello.html был изменен, но при этом эти изменения еще не зафиксированы в репозитории. Также обратите внимание на то, что сообщение о состоянии дает вам подсказку о том, что нужно делать дальше. Если вы хотите добавить эти изменения в репозиторий, используйте команду `git add`. В противном случае используйте команду `git checkout` для отмены изменений.

```
[nata /Users/nata/Desktop/git/hello% vim hello.html
[nata /Users/nata/Desktop/git/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")
```

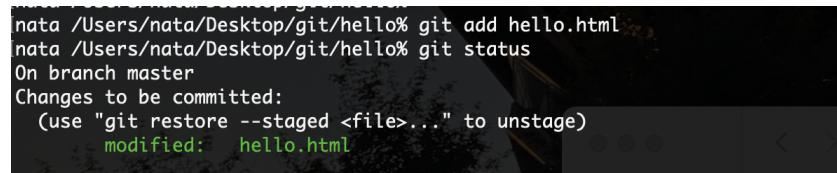
Рис. 6.2: git status

7 1.4 Индексация изменений

Теперь выполните команду git, чтобы проиндексировать изменения. Проверьте 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. Если вы решили, что не хотите коммитить изменения, команда состояния напомнит вам о том, что с помощью команды git reset можно снять индексацию этих изменений.



```
nata /Users/nata/Desktop/git/hello% git add hello.html
nata /Users/nata/Desktop/git/hello% git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    modified:   hello.html
```

Рис. 7.1: Индексация изменений

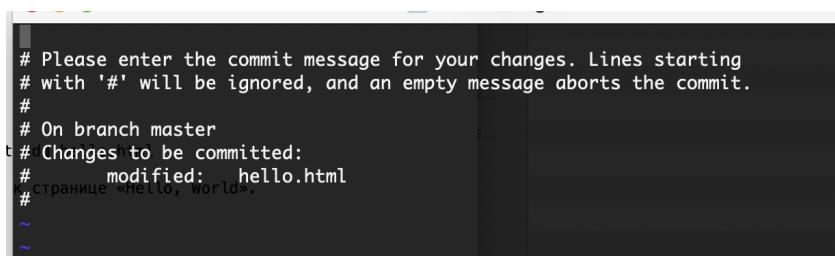
8 1.4.1 Коммит изменений

Когда вы ранее использовали `git commit` для коммита первоначальной версии файла `hello.html` в репозиторий, вы включили метку `-m`, которая делает комментарий в командной строке. Команда `commit` позволит вам интерактивно редактировать комментарии для коммита. Теперь давайте это проверим. Если вы опустите метку `-m` из командной строки, `git` перенесет вас в редактор по вашему выбору. Редактор выбирается из следующего списка (в порядке приоритета):

- переменная среды `GIT_EDITOR`
- параметр конфигурации `core.editor`
- переменная среды `VISUAL`
- переменная среды `EDITOR`

Сделайте коммит и проверьте состояние.

```
git commit
```



The screenshot shows a terminal window with a dark background. It displays the following text:

```
# 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
#   странице «Hello, World».
#
~
```

Рис. 8.1: Коммит изменений

1.4.2 Добавьте стандартные теги страницы

Hello, World!

Теперь добавьте это изменение в индекс git. `git add hello.html`

Теперь добавьте заголовки HTML (секцию) к странице «Hello, World».

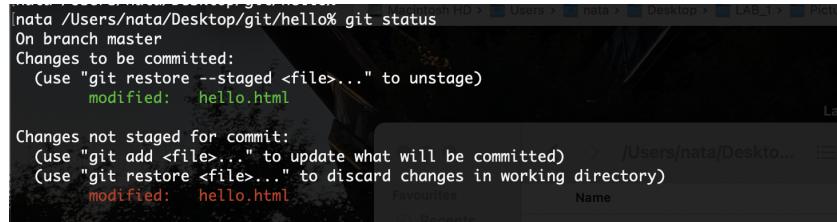
Hello, World!

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

Проверьте текущий

статус:

```
git status
```



```
nata /Users/nata/Desktop/git/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
```

Рис. 8.2: git status

git commit -m “Added standard HTML page tags” git status Состояние команды говорит о том, что hello.html имеет незафиксированные изменения, но уже не в буферной зоне.

Теперь добавьте второе изменение в индекс, а затем проверьте состояние с помощью команды git status.

```
git add . git status
```

В качестве файла для добавления, мы использовали текущий каталог (.). Это краткий и удобный путь для добавления всех изменений в файлы текущего каталога и его подкаталоги. Но поскольку он добавляет все, не лишним будет проверить состояние перед запуском add, просто чтобы убедиться, что вы не добавили какой-то файл, который добавлять было не нужно. Второе изменение было проиндексировано и готово к коммиту.

Сделайте коммит второго изменения

```
git commit -m “Added HTML header”
```

```
nata /Users/nata/Desktop/git/hello% git commit -m "Added standard HTML page tags"
[master e97efbd] Added standard HTML page tags
 1 file changed, 5 insertions(+)
nata /Users/nata/Desktop/git/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")
nata /Users/nata/Desktop/git/hello% git add .
nata /Users/nata/Desktop/git/hello% git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    modified:   hello.html

nata /Users/nata/Desktop/git/hello% git commit -m "Added HTML header"
[master 5cf00b3] Added HTML header
 1 file changed, 2 insertions(+)
nata /Users/nata/Desktop/git/hello%
```

Рис. 8.3: коммит второго изменения

9 1.4.3 История

Получим список произведенных изменений:

git log

```
nata /Users/nata/Desktop/git/hello% git log
commit 5cf00b38bb86db48d2cd7055928163f3830900e7 (HEAD -> master)
Author: nnnefedova <1132226537@pfur.ru>
Date:   Sat Feb 15 13:46:22 2025 +0300

    Added HTML header

commit e97efbd25755a87d910b7ddc4b58746cc84a51ba
Author: nnnefedova <1132226537@pfur.ru>
Date:   Sat Feb 15 13:45:23 2025 +0300

    Added standard HTML page tags

commit 2b315ace5b35329076bf2832536b4ef9ddf4f663 (application/javascript, renamed from hello.html)
Author: nnnefedova <1132226537@pfur.ru>
Date:   Sat Feb 15 13:36:11 2025 +0300

    Initial Commit
nata /Users/nata/Desktop/git/hello%
```

Рис. 9.1: git log

Однострочный формат истории:

git log --pretty=oneline Есть много вариантов отображения лога.

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=git log --pretty=oneline --all Справочная страница:

man git-log Инструмент gitk полезен в изучении истории изменений.

10 Получение старых версий

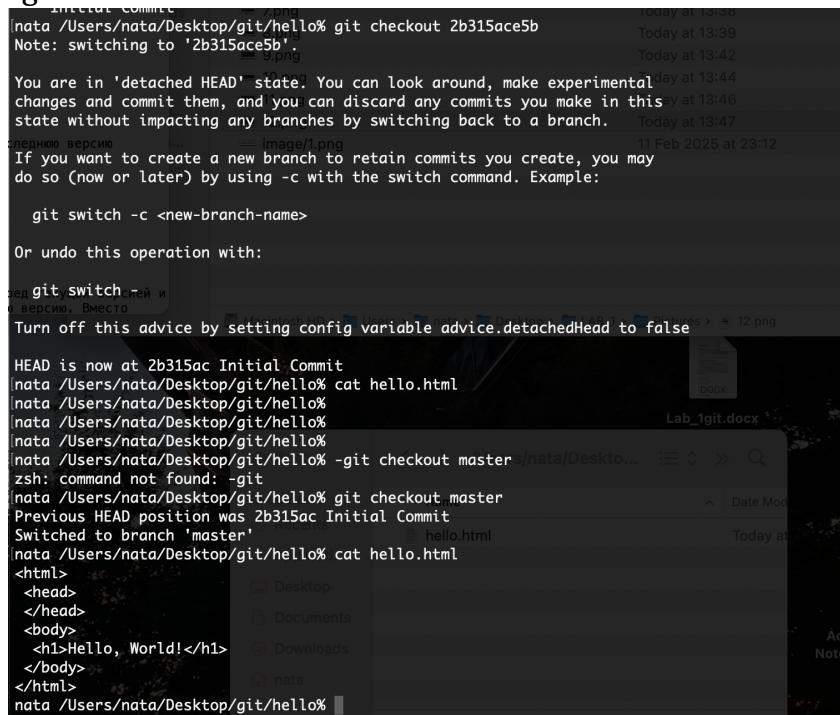
Возвращаться назад в историю очень просто. Команда `checkout` скопирует любой снимок из репозитория в рабочий каталог. Получите хэши предыдущих версий

git log Изучите данные лога и найдите хэш для первого коммита. Он должен быть в последней строке данных. Используйте этот хэш-код (достаточно первых 7 знаков) в команде ниже. Затем проверьте содержимое файла `hello.html`.

```
-git checkout -cat hello.html
```

Вернитесь к последней версии в ветке `master`

```
-git checkout master -cat hello.html
```



The screenshot shows a terminal window with the following content:

```
initial commit
nata /Users/nata/Desktop/git/hello% git log
Note: switching to '2b315ace5b'.
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 -m <old-branch-name>
Turn off this advice by setting config variable advice.detachedHead to false
HEAD is now at 2b315ac Initial Commit
nata /Users/nata/Desktop/git/hello% cat hello.html
nata /Users/nata/Desktop/git/hello%
nata /Users/nata/Desktop/git/hello%
nata /Users/nata/Desktop/git/hello%
nata /Users/nata/Desktop/git/hello% -git checkout master
zsh: command not found: -git
nata /Users/nata/Desktop/git/hello% git checkout master
Previous HEAD position was 2b315ac Initial Commit
Switched to branch 'master'
nata /Users/nata/Desktop/git/hello% cat hello.html
<html>
<head>
</head>
<body>
<h1>Hello, World!</h1>
</body>
</html>
nata /Users/nata/Desktop/git/hello%
```

`master` — имя ветки по

умолчанию. Переключая имена веток, вы попадаете на последнюю версию вы-

бранной ветки.

11 1.4.5 Создание тегов версий

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

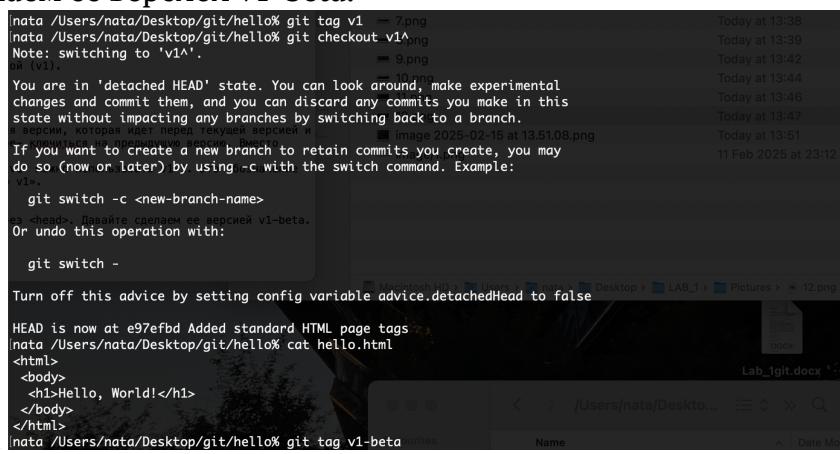
git tag v1 Теперь текущая версия страницы называется v1. Теги для предыдущих версий Давайте создадим тег для версии, которая идет перед текущей версией и назовем его v1-beta. В первую очередь нам надо переключиться на предыдущую версию. Вместо поиска до хэш, мы будем использовать ^, обозначающее «родитель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

`git checkout v1^ cat hello.html` Это версия с тегами

и

, но еще пока без

. Давайте сделаем ее версией v1-beta.



The screenshot shows a terminal window with the following command and output:

```
[nata /Users/nata/Desktop/git/hello% git tag v1 - 7.png
[nata /Users/nata/Desktop/git/hello% git checkout v1^
Note: switching to 'v1^'.
[nata /Users/nata/Desktop/git/hello% cat hello.html
<html>
<body>
<h1>Hello, World!</h1>
</body>
</html>
[nata /Users/nata/Desktop/git/hello% git tag v1-beta
```

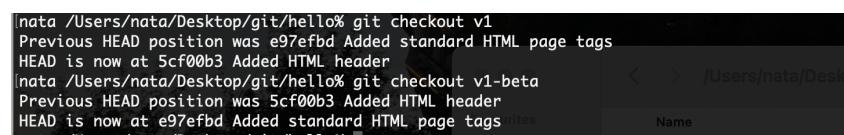
Below the terminal, there is a file browser window showing a folder structure with files like 'image 2025-02-15 at 13:51:08.png' and 'Lab_1git.docx'.

`git tag v1-beta`

12 1.4.6 Переключение по имени тега

Теперь попробуйте попереключаться между двумя отмеченными версиями.

git checkout v1 git checkout v1-beta



```
|nata /Users/nata/Desktop/git/hello% git checkout v1
Previous HEAD position was e97efbd Added standard HTML page tags
HEAD is now at 5cf00b3 Added HTML header
|nata /Users/nata/Desktop/git/hello% git checkout v1-beta
Previous HEAD position was 5cf00b3 Added HTML header
HEAD is now at e97efbd Added standard HTML page tags
```

Рис. 12.1: Переключение по имени тега

13 1.4.7 Просмотр тегов с помощью команды tag

Вы можете увидеть, какие теги доступны, используя команду git tag. git tag

Вы также можете посмотреть теги в логе.

git log master –all Вы можете видеть теги (v1 и v1-beta) в логе вместе с именем ветки (master). Кроме того HEAD показывает коммит, на который вы переключились (на данный момент это v1-beta).

```
nata /Users/nata/Desktop/git/hello% git tag
v1
v1-beta
nata /Users/nata/Desktop/git/hello% git log master --all
commit 5cf00b38bb86db48a2cd7055928163f3830900e7 (tag: v1, master)
Author: nnnefedova <1132226537@pfur.ru>
Date:   Sat Feb 15 13:46:22 2025 +0300

    Added HTML header

commit e97efbd25755a87d910b7ddc4b58746cc84a51ba (HEAD, tag: v1-beta)
Author: nnnefedova <1132226537@pfur.ru>
Date:   Sat Feb 15 13:45:23 2025 +0300

    Added standard HTML page tags

commit 2b315ace5b35329076bf2832536b4ef9ddf4f663
Author: nnnefedova <1132226537@pfur.ru>
Date:   Sat Feb 15 13:36:11 2025 +0300

    Initial Commit
nata /Users/nata/Desktop/git/hello%
```

Рис. 13.1: git log master –all

14 1.5 Отмена локальных изменений (до индексации)

1.5.1 Переключитесь на ветку master Убедитесь, что вы находитесь на последнем коммите ветки master, прежде чем продолжить работу.

```
git checkout master
```

1.5.2 Измените hello.html Иногда случается, что вы изменили файл в рабочем каталоге, и хотите отменить последние коммиты. С этим справится команда git checkout. Внесите изменение в файл hello.html в виде нежелательного комментария.

```
Hello, World!
```

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

2.png
3.png
4.png
5.png
6.png
7.png
8.png
9.png
10.png
11.png

Рис. 14.1: Изменим hello.html

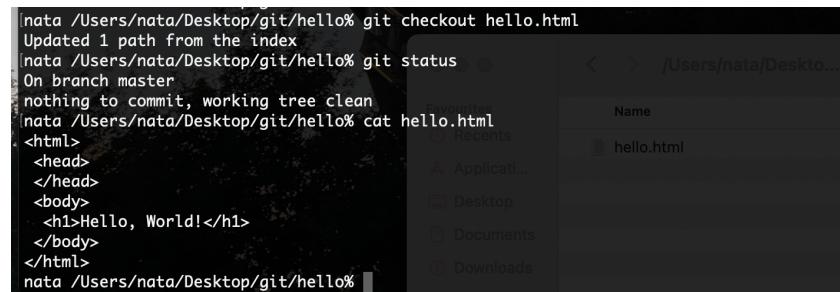
1.5.3 Проверьте состояние Сначала проверьте состояние рабочего каталога.

```
git status
```

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

1.5.4 Отмена изменений в рабочем каталоге Используйте команду git checkout для переключения версии файла hello.html в репозитории.

```
git checkout hello.html git status cat hello.html
```



The screenshot shows a terminal window and a file browser side-by-side. The terminal window displays the following command sequence:

```
[nata /Users/nata/Desktop/git/hello% git checkout hello.html
Updated 1 path from the index
[nata /Users/nata/Desktop/git/hello% git status
On branch master
nothing to commit, working tree clean
[nata /Users/nata/Desktop/git/hello% cat hello.html
<html>
  <head>
  </head>
  <body>
    <h1>Hello, World!</h1>
  </body>
</html>
nata /Users/nata/Desktop/git/hello%
```

To the right of the terminal is a file browser window titled "/Users/nata/Desktop..". It lists a single file named "hello.html".

Рис. 14.2: Отмена изменений в рабочем каталоге

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

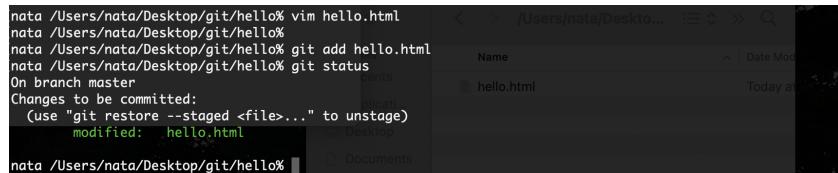
15 1.6 Отмена проиндексированных изменений (перед коммитом)

1.6.1 Измените файл и проиндексируйте изменения Внесите изменение в файл hello.html в виде нежелательного комментария

Hello, World!

Проиндексируйте это изменение. git add hello.html

1.6.2 Проверьте состояние Проверьте состояние нежелательного изменения. git status Состояние показывает, что изменение было проиндексировано и готово к коммиту.



```
nata /Users/nata/Desktop/git/hello% vim hello.html
nata /Users/nata/Desktop/git/hello%
nata /Users/nata/Desktop/git/hello% git add hello.html
nata /Users/nata/Desktop/git/hello% git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    modified:   hello.html
nata /Users/nata/Desktop/git/hello%
```

Рис. 15.1: Состояние

1.6.3 Выполните сброс буферной зоны 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.

1.6.4 Переключитесь на версию коммита git checkout hello.html git status Наш рабочий каталог опять чист.

```
nata ~/Users/nata/Desktop/git/hello% git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    modified:   hello.html

nata ~/Users/nata/Desktop/git/hello% git reset HEAD hello.html
Unstaged changes after reset:
M      hello.html
nata ~/Users/nata/Desktop/git/hello% git checkout hello.html hello.html
Updated 1 path from the index
nata ~/Users/nata/Desktop/git/hello% git status
On branch master
nothing to commit, working tree clean
nata ~/Users/nata/Desktop/git/hello%
```

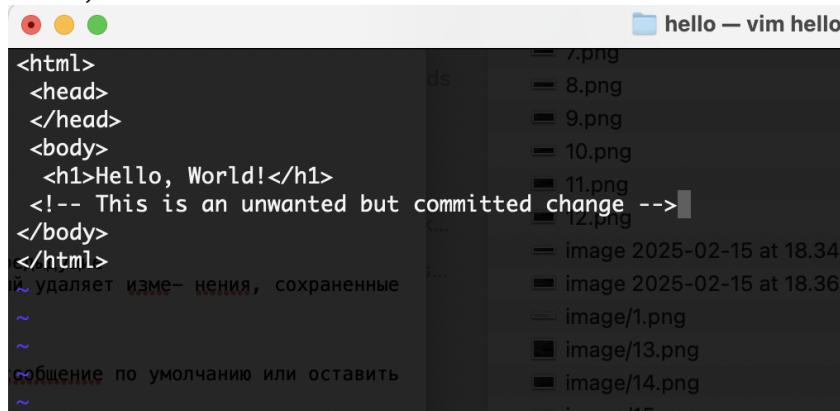
Рис. 15.2: Переключитесь на версию коммита

16 1.7 Отмена коммитов

1.7.1 Отмена коммитов Иногда вы понимаете, что новые коммиты являются неверными, и хотите их отменить. Есть несколько способов решения этого вопроса, здесь мы будем использовать самый безопасный. Мы отменим коммит путем создания нового коммита, отменяющего нежелательные изменения.

1.7.2 Измените файл и сделайте коммит Измените файл hello.html на следующий.

Hello, World!



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

удаляет изменения, сохраненные

~

~

~ сообщение по умолчанию или оставить

~

hello — vim hello

- 7.png
- 8.png
- 9.png
- 10.png
- 11.png
- 12.png
- image 2025-02-15 at 18.34
- image 2025-02-15 at 18.36
- image/1.png
- image/13.png
- image/14.png

Выполните:

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

```
[nata /Users/nata/Desktop/git/hello% git add hello.html
[nata /Users/nata/Desktop/git/hello% git commit -m "Oops, we didn't want this commit"
[master ab9728f] Oops, we didn't want this commit
 1 file changed, 2 insertions(+), 1 deletion(-)
[nata /Users/nata/Desktop/git/hello%
```

Рис. 16.1: Измените файл и сделайте коммит

1.7.3 Сделайте коммит с новыми изменениями, отменяющими предыдущие Чтобы отменить коммит, нам необходимо сделать коммит, который удаляет

изменения, сохраненные нежелательным коммитом.

git revert HEAD Перейдите в редактор, где вы можете отредактировать коммит-сообщение по умолчанию или оставить все как есть. Сохраните и закройте файл.

```
Revert "Oops, we didn't want this commit" 8.png
This reverts commit ab9728f353fc0b3dcaeb7f8c89b76cd1fc90baee.
It removes changes, saved by
# Please enter the commit message for your changes. Lines starting
# with '#' will be ignored, and an empty message aborts the commit.
# биение по умолчанию или оставить
# On branch master
# Changes to be committed:
#       modified:   hello.html
#
# Untracked files:
#       .DS_Store
#
~
~
~
~
```

1.7.4 Проверьте лог Проверка лога показывает нежелательные и отмененные коммиты в наш репозиторий.

git log

```
nata /Users/nata/Desktop/git/hello% git log
commit 557c3e91f1bb22d4f7d0505ac864a3ee6c120a18 (HEAD -> master)
Author: nnnefedova <1132226537@pfur.ru>
Date:   Sat Feb 15 18:40:09 2025 +0300
    в наш репозиторий.
        Revert "Oops, we didn't want this commit"
        This reverts commit ab9728f353fc0b3dcaeb7f8c89b76cd1fc90baee.
        commit ab9728f353fc0b3dcaeb7f8c89b76cd1fc90baee
        Author: nnnefedova <1132226537@pfur.ru>
        Date:   Sat Feb 15 18:38:37 2025 +0300
            было бы неплохо иметь команду
            для отмены коммита в истории git log.
            Oops, we didn't want this commit
            commit 5cf00b38bb86db48d2cd7055928163f3830900e7 (tag: v1)
            Author: nnnefedova <1132226537@pfur.ru>
            Date:   Sat Feb 15 13:46:22 2025 +0300
                Added HTML header
            commit e97efbd25755a87d910b7ddc4b58746cc84a51ba (tag: v1-beta)
            Author: nnnefedova <1132226537@pfur.ru>
            Date:   Sat Feb 15 13:45:23 2025 +0300
                Added standard HTML page tags
            commit 2b315aceb5b35329076bf2832536b4ef9ddf4f663
            Author: nnnefedova <1132226537@pfur.ru>
            Date:   Sat Feb 15 13:36:11 2025 +0300
                Initial Commit
nata /Users/nata/Desktop/git/hello%
```

Рис. 16.2: Сделайте коммит с новыми изменениями

#1.8 Удаление коммитов из ветки git revert является мощной командой, которая позволяет отменить любые коммиты в репозиторий. Однако, и оригинальный и

«отмененный» 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`.

1.8.1 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.

1.8.2 Проверьте нашу историю Давайте сделаем быструю проверку нашей истории коммитов. Выполните:

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

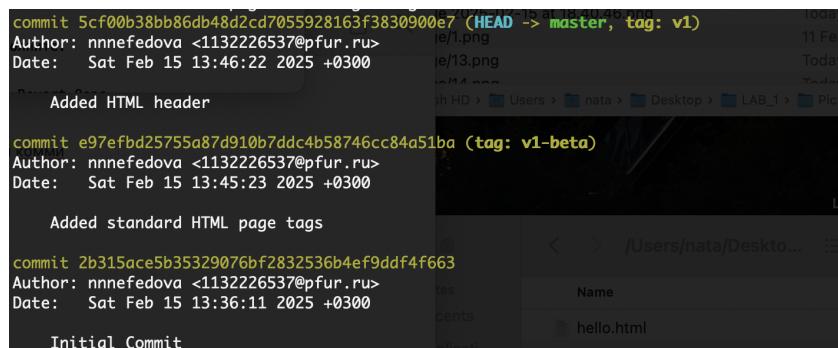
17 1.8.3 Для начала отметьте эту ветку

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

```
git tag oops
```

1.8.4 Сброс коммитов к предшествующим коммиту 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
```



The terminal window shows the following log output:

```
commit 5cf00b38bb86db48d2cd7055928163f3830900e7 (HEAD -> master, tag: v1)
Author: nnnefedova <1132226537@pfur.ru>
Date:   Sat Feb 15 13:46:22 2025 +0300

    Added HTML header

commit e97efbd25755a87d910b7ddc4b58746cc84a51ba (tag: v1-beta)
Author: nnnefedova <1132226537@pfur.ru>
Date:   Sat Feb 15 13:45:23 2025 +0300

    Added standard HTML page tags

commit 2b315ace5b35329076bf2832536b4ef9ddf4f663
Author: nnnefedova <1132226537@pfur.ru>
Date:   Sat Feb 15 13:36:11 2025 +0300

    Initial Commit
```

Below the terminal, a file browser window is open, showing a file named "hello.html".

Рис. 17.1: Сброс коммитов к предшествующим коммиту Oops

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.

1.8.5 Ничего никогда не теряется Что же случается с ошибочными коммитами? Оказывается, что коммиты все еще находятся в репозитории. На самом деле, мы

все еще можем на них ссылаться. Помните, в начале этого урока мы создали для отмененного коммита тег «oops». Давайте посмотрим на все коммиты.

git log --all Мы видим, что ошибочные коммиты не исчезли. Они все еще 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 имен. Коммиты, на которые нет ссылок, остаются в репозитории до тех пор, пока не будет запущен сборщик мусора.

```
nata /Users/nata/Desktop/git/hello% git log --all
commit 557c3e91f1bb22d4f7d0505ac864a3ee6c120a18 (tag: oops)
Author: nnnefedova <1132226537@pfur.ru> 9.png
Date:   Sat Feb 15 18:40:09 2025 +0300 10.png
    любой «аварии» как правило, можно
    исправить, «Revert "Oops, we didn't want this commit»
    ветки.
    This reverts commit ab9728f353fc0b3dcaebe7f8c89b76cd1fc90baee.
    image 2025-02-15 at 18.36.42.png
commit ab9728f353fc0b3dcaebe7f8c89b76cd1fc90baee 025-02-15 at 18.38.02.png
Author: nnnefedova <1132226537@pfur.ru> image 2025-02-15 at 18.38.49.png
Date:   Sat Feb 15 18:38:37 2025 +0300 image 2025-02-15 at 18.40.18.png
    Oops, we didn't want this commit image 2025-02-15 at 18.40.46.png
    image 2025-02-15 at 18.43.55.png
commit 5cf00b38bb86db48d2cd7055928163f3830900e7 (HEAD -> master, tag: v1)
Author: nnnefedova <1132226537@pfur.ru>
Date:   Sat Feb 15 13:46:22 2025 +0300
    Added HTML header
    КОММІТЫ
commit e97efbd25755a87d910b7ddc4b58746cc84a51ba (tag: v1-beta)
Author: nnnefedova <1132226537@pfur.ru>
Date:   Sat Feb 15 13:45:23 2025 +0300
    Added standard HTML page tags
commit 2b315ace5b35329076bf2832536b4ef9ddf4f663
Author: nnnefedova <1132226537@pfur.ru>
Date:   Sat Feb 15 13:36:11 2025 +0300
    Initial Commit
```

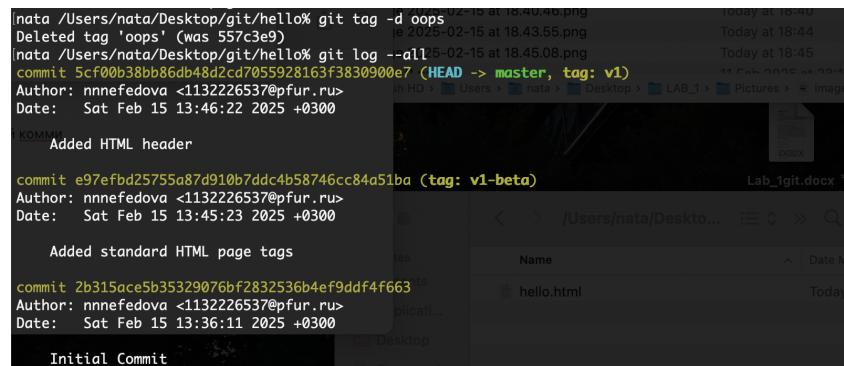
Рис. 17.2: ошибочные коммиты не исчезли

1.8.6 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 удаленных репозиториях, сброс может сбить с толку других пользователей ветки.

18 1.9 Удаление тега oops

1.9.1 Удаление тега oops 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.



```
[nata /Users/nata/Desktop/git/hello% git tag -d oops
Deleted tag 'oops' (was 557c3e9)
[nata /Users/nata/Desktop/git/hello% git log --all
commit 5cf00b38bb86db48d2cd7055928163f3830900e7 (HEAD -> master, tag: v1)
Author: nnnefedova <1132226537@pfur.ru>
Date:   Sat Feb 15 13:46:22 2025 +0300
    Added HTML header
commit e97efbd25755a87d910b7dc4b58746cc84a51ba (tag: v1-beta)
Author: nnnefedova <1132226537@pfur.ru>
Date:   Sat Feb 15 13:45:23 2025 +0300
    Added standard HTML page tags
commit 2b315ace5b35329076bf2832536b4ef9ddf4f663
Author: nnnefedova <1132226537@pfur.ru>
Date:   Sat Feb 15 13:36:11 2025 +0300
    Initial Commit
```

Рис. 18.1: Удаление тега oops

19 1.10 Внесение изменений в коммиты

1.10.1 Измените страницу, а затем сделайте коммит Добавьте в страницу комментарий автора (вставьте свою фамилию).

Hello, World!

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

~  

~  

~  

~  

~  

~  

~  

~  

~  

~
```

Рис. 19.1: Внесение изменений в коммиты

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

```
nata /Users/nata/Desktop/git/hello% git add hello.html
nata /Users/nata/Desktop/git/hello% git commit -m "Add an author comment"
[master ac63d68] Add an author comment
 1 file changed, 1 insertion(+)
nata /Users/nata/Desktop/git/hello%
```

Рис. 19.2: Добавим кормит с комментарием

1.10.2 Необходим email После совершения коммита вы понимаете, что любой хороший комментарий должен включать электронную почту автора. Обновите страницу hello, включив в нее email.

Hello, World!

```

<!-- Author: Nefedova Natalia N. <1132226537@pfur.ru> -->
<html>
  <head>
  </head>
  <body>
    <h1>Hello, World!</h1>
  </body>
</html>
//это электронной почты, давайте почты.
~
~
~

```

The terminal shows the vim editor with the file 'hello.html'. The code includes an author comment with an email address, an HTML structure with an h1 tag, and a note in Russian about adding an email. The right pane shows a directory tree with image files from 'image/17.png' to 'image/26.png'.

Рис. 19.3: Добавим email

1.10.3 Измените предыдущий коммит Мы действительно не хотим создавать отдельный коммит только ради электронной почты. Давайте изменим предыдущий коммит, включив в него адрес электронной почты. Выполните:

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

1.10.4 Просмотр истории Выполните: `git log` Мы можем увидеть, что оригинальный коммит «автор» заменен коммитом «ав- топ/email». Этого же эффекта можно достичь путем сброса последнего коммита в ветке, и повторного коммита новых изменений.

```

[nata /Users/nata/Desktop/git/hello% git log
commit e23b474eeff11412e1c51e6b7ca74bd653d20d2 (HEAD -> master)
Author: nnnefedova <1132226537@pfur.ru>
Date:   Sat Feb 15 18:47:19 2025 +0300
        image/25.png
        Add an author/email comment image/26.png
commit 5cf00b38bb86db48d2cd7055928163f3830900e7 (tag: v1)
Author: nnnefedova <1132226537@pfur.ru>
Date:   Sat Feb 15 13:46:22 2025 +0300
        Added HTML header
commit e97efbd25755a87d910b7ddc4b58746cc84a51ba (tag: v1-beta)
Author: nnnefedova <1132226537@pfur.ru>
Date:   Sat Feb 15 13:45:23 2025 +0300
        Added standard HTML page tags
commit 2b315ace5b35329076bf2832536b4ef9ddf4f663
Author: nnnefedova <1132226537@pfur.ru>
Date:   Sat Feb 15 13:36:11 2025 +0300
        Initial Commit

```

The terminal shows the git log command output. It lists three commits: 'Initial Commit', 'Added standard HTML page tags', and 'Added HTML header'. The first commit has been amended to include the author and email information. The commit messages and dates are also visible.

Рис. 19.4: git log

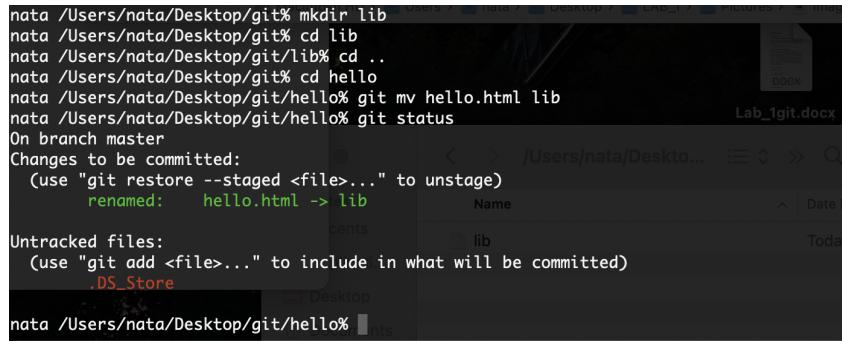
20 1.11 Перемещение файлов

1.11.1 Переместите файл hello.html в каталог lib 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 git status сообщает, что файл был перемещен.



```
nata /Users/nata/Desktop/git% mkdir lib
nata /Users/nata/Desktop/git% cd lib
nata /Users/nata/Desktop/git/lib% cd ..
nata /Users/nata/Desktop/git% cd hello
nata /Users/nata/Desktop/git/hello% git mv hello.html lib
nata /Users/nata/Desktop/git/hello% git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    renamed:   hello.html -> lib
Untracked files:
  (use "git add <file>..." to include in what will be committed)
    .DS_Store
nata /Users/nata/Desktop/git/hello%
```

Рис. 20.1: Перемещение файла файл hello.html в каталог lib

21 1.12 Второй способ перемещения файлов

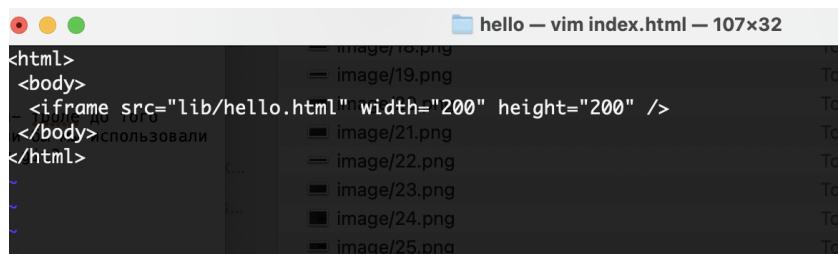
Положительной чертой git является то, что вы можете забыть о версионном контроле до того момента, когда вы готовы приступить к коммиту кода. Что бы случилось, если бы мы использовали командную строку операционной системы для перемещения файлов вместо команды git? Следующий набор команд идентичен нашим последним действиям. Работы здесь побольше, но результат тот же. Мы могли бы выполнить: `mkdir lib mv hello.html lib git add lib/hello.html git rm hello.html`

1.12.1 Коммит в новый каталог Давайте сделаем коммит этого перемещения:

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

1.13 Подробнее о структуре 1.13.1 Добавление index.html Добавим файл index.html в наш репозиторий

Добавьте файл и сделайте коммит.



The screenshot shows a terminal window titled 'hello — vim index.html — 107x32'. On the left, there is a code editor displaying an HTML file with the following content:

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

On the right, there is a file browser showing a directory structure:

- image/0.png
- image/19.png
- image/21.png
- image/22.png
- image/23.png
- image/24.png
- image/25.png

Рис. 21.1: Добавление index.html

`git add index.html git commit -m "Added index.html."` Теперь при открытии index.html, вы должны увидеть кусок страницы hello в маленьком окошке.

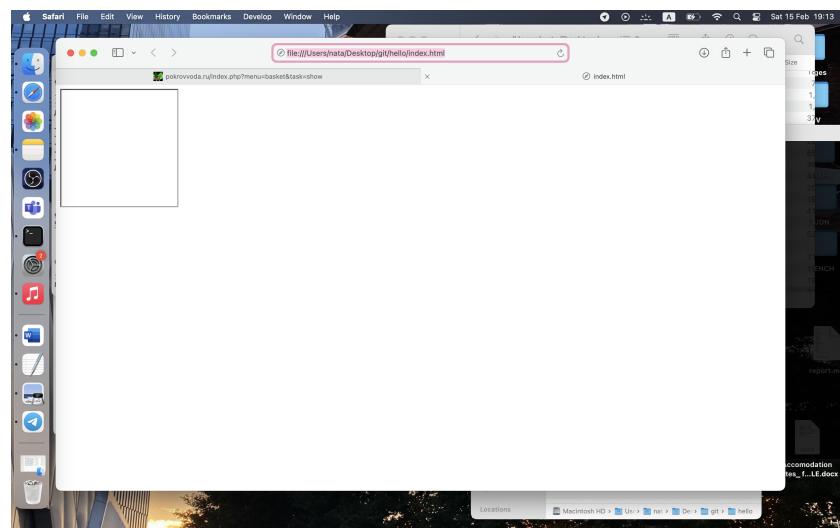


Рис. 21.2: страница hello в маленьком окошке

22 1.14 Git внутри: Каталог .git

1.14.3 Углубляемся в базу данных объектов Выполните:

```
nata /Users/nata/Desktop/git/hello% ls -C .git
COMMIT_EDITMSG  ORIG_HEAD  description  lib
HEAD          config     hooks        index
nata /Users/nata/Desktop/git/hello%
```

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

Смотрим в один из каталогов с именем из 2 букв. Вы увидите 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.

```
nata /Users/nata/Desktop/git/hello% ls -C .git/objects/ml
02      2b      4d      61      72      ab      d9      e9      f8      pack
10      31      55      62      a4      ac      e2      f2      f9
27      47      5c      66      a9      cc      e6      f7      info
nata /Users/nata/Desktop/git/hello%
```

Рис. 22.1: Один из каталогов с именем из 2 букв.

1.14.4 Config File Выполните:

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.

1.14.5 Ветки и теги Выполните:

```
-ls .git/refs -ls .git/refs/heads -ls .git/refs/tags -cat .git/refs/tags/v1
```

```

nata /Users/nata/Desktop/git/hello% ls .git/refs
heads tags
nata /Users/nata/Desktop/git/hello% -ls .git/refs/heads
zsh: command not found: -ls
nata /Users/nata/Desktop/git/hello% ls .git/refs/heads
master
nata /Users/nata/Desktop/git/hello% ls .git/refs/tags
v1 v1-beta
nata /Users/nata/Desktop/git/hello% cat .git/refs/tags/v1
5cf00b38bb86db48d2cd7055928163f3830900e7
nata /Users/nata/Desktop/git/hello%

```

Рис. 22.2: Ветки и теги

Вы должны узнавать файлы в подкаталоге тегов. 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 все, что вы увидите в этом каталоге — это ветка `master`.

Выполните:

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

```

nata /Users/nata/Desktop/git/hello% cat .git/HEAD
ref: refs/heads/master
nata /Users/nata/Desktop/git/hello%

```

Рис. 22.3: `cat .git/HEAD`

#1.15 Работа непосредственно с объектами git 1.15.1 Поиск последнего коммита

Выполните:

`git log --max-count=1` Эта команда должна показать последний коммит в репозиторий. SHA1 хэш в вашей системе, вероятно, отличается от моего, но вы увидите что-то наподобие этого.

```

nata /Users/nata/Desktop/git/hello% git log --max-count=1
commit d992254c2a5a732576cc6fb7566b855052a7194e (HEAD -> master)
Author: nnnefedova <1132226537@pfur.ru>
Date:   Sat Feb 15 19:12:55 2025 +0300

    Added index.html.
nata /Users/nata/Desktop/git/hello%

```

Рис. 22.4: `git log --max-count=1`

1.15.2 Вывод последнего коммита с помощью SHA1 хэша Выполните:

```
git cat-file -t git cat-file -p
```

```
nata /Users/nata/Desktop/git/hello% git cat-file -t d992254c2a5a7
commit
tree cc4c999575ca7ec59d55221ca40f314ba6a4567d
parent 66b1dd9f2807d4c1cf9996215d24065074013d98
author nnnefedova <1132226537@pfur.ru> 1739635975 +0300
committer nnnefedova <1132226537@pfur.ru> 1739635975 +0300
Added index.html.
nata /Users/nata/Desktop/git/hello%
```

Рис. 22.5: Вывод последнего коммита с помощью SHA1

1.15.3 Поиск дерева 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. Выполните:

```
git cat-file -p
```

```
nata /Users/nata/Desktop/git/hello% git cat-file -p d992254c2a5
tree cc4c999575ca7ec59d55221ca40f314ba6a4567d
parent 66b1dd9f2807d4c1cf9996215d24065074013d98
author nnnefedova <1132226537@pfur.ru> 1739635975 +0300
committer nnnefedova <1132226537@pfur.ru> 1739635975 +0300
Added index.html.
Favourites
Name
nata /Users/nata/Desktop/git/hello%
```

Рис. 22.6: Поиск дерева

23 1.16 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.

1.16.1 Создайте ветку Давайте назовем нашу новую ветку «style». Выполните:

```
git checkout -b style git status git checkout -b
```

```
nata /Users/nata/Desktop/git/hello% git checkout -b style
Switched to a new branch 'style'
nata /Users/nata/Desktop/git/hello% git status
On branch style
Untracked files:
  (use "git add <file>..." to include in what will be committed)
    .DS_Store
nothing added to commit but untracked files present (use "git add" to track)
```

Рис. 23.1: Создайте ветку

Обратите внимание, что команда git status сообщает о том, что вы находитесь в ветке «style».

1.16.2 Добавьте файл стилей style.css Выполните:

```
touch lib/style.css
```

Файл lib/style.css: h1 { color: red; } Выполните:

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

```
nata /Users/nata/Desktop/git/lib% touch lib/style.css
touch: lib/style.css: No such file or directory
nata /Users/nata/Desktop/git/lib% touch style.css
nata /Users/nata/Desktop/git/lib% vim style.css
nata /Users/nata/Desktop/git/lib% Macintosh HD > Users > nata > Desktop > LAB_1 > Pictures
nata /Users/nata/Desktop/git/lib%
nata /Users/nata/Desktop/git/lib% git add lib/style.css
warning: could not open directory 'lib/lib/': No such file or directory
fatal: pathspec 'lib/style.css' did not match any files
nata /Users/nata/Desktop/git/lib% git add style.css
nata /Users/nata/Desktop/git/lib% git commit -m "Added css stylesheet"
[master 36b7bef] Added css stylesheet
  1 file changed, 3 insertions(+)
  create mode 100644 lib/style.css
nata /Users/nata/Desktop/git/lib%
```

Рис. 23.2: Добавьте файл стилей style.css

1.16.3 Измените основную страницу Обновите файл hello.html, чтобы использовать стили style.css.

Hello, World!

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

```
nata /Users/nata/Desktop/git/lib% vim hello.html
nata /Users/nata/Desktop/git/lib% git add lib/hello.html
warning: could not open directory 'lib/lib/': No such file or directory
fatal: pathspec 'lib/hello.html' did not match any files
nata /Users/nata/Desktop/git/lib% git commit -m "Hello uses style.css"
[master 2571c3d] Hello uses style.css
 1 file changed, 9 insertions(+)
 create mode 100644 lib/hello.html
nata /Users/nata/Desktop/git/lib%
```

Рис. 23.3: Обновим hello.html

1.16.4 Измените index.html

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

Выполните: git add index.html git commit -m "Updated index.html"

```
nata /Users/nata/Desktop/git% cd hello
nata /Users/nata/Desktop/git/hello% vim index.html
nata /Users/nata/Desktop/git/hello% git add index.html
nata /Users/nata/Desktop/git/hello% git commit -m "Updated index.html"
[style ff4f132] Updated index.html
 1 file changed, 3 insertions(+)
nata /Users/nata/Desktop/git/hello%
```

Рис. 23.4: Сделаем комет изменений

#1.17 Навигация по веткам Теперь в вашем проекте есть две ветки: Выполните:

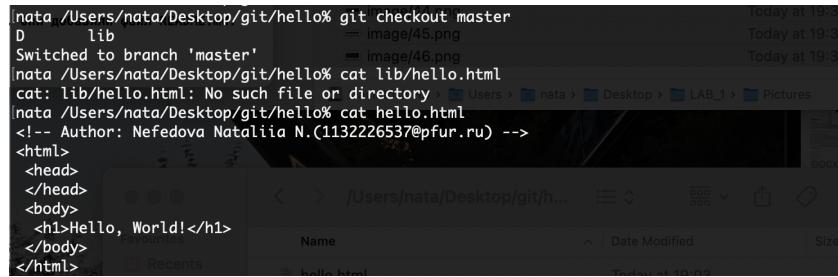
git log -all

```
nata /Users/nata/Desktop/git/hello% git log --all
commit ff4f13206c4f88530f04ea8678f2624e7354385c (HEAD -> style)
Author: nnnefedova <1132226537@pfur.ru>
Date:   Sat Feb 15 19:34:55 2025 +0300
        Updated index.html
        └── image/23.png
        └── image/24.png
        └── image/25.png
        └── image/26.png
commit d992254c2a5a732576c68fb7566b855052a7194e (master)
Author: nnnefedova <1132226537@pfur.ru>
Date:   Sat Feb 15 19:12:55 2025 +0300
        Added index.html.
        └── image/27.png
        └── image/28.png
commit 66b1dd9f2807d4c1cf9996215d24065074013d98
Author: nnnefedova <1132226537@pfur.ru>
Date:   Sat Feb 15 19:10:35 2025 +0300
        └── image/29.png
        └── image/30.png
        └── image/31.png
        └── image/32.png
        └── image/33.png
        └── image/34.png
        └── image/35.png
        └── image/36.png
```

Рис. 23.5: Навигация по веткам

1.17.1 Переключение на ветку master

Используйте команду git checkout Для переключения между ветками
git checkout master cat lib/hello.html

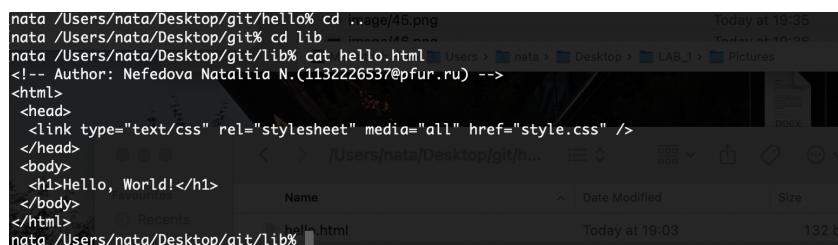


```
nata /Users/nata/Desktop/git/hello% git checkout master
D lib
Switched to branch 'master'
nata /Users/nata/Desktop/git/hello% cat lib/hello.html
cat: lib/hello.html: No such file or directory
nata /Users/nata/Desktop/git/hello% cat hello.html
<!-- Author: Nefedova Natalia N.(1132226537@pfur.ru) -->
<html>
<head>
</head>
<body>
<h1>Hello, World!</h1>
</body>
</html>
```

Рис. 23.6: git checkout - на ветку master

1.17.2 Вернемся к ветке style Выполните: git checkout style cat lib/hello.html

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



```
nata /Users/nata/Desktop/git/hello% cd .. /ge/45.png
nata /Users/nata/Desktop/git% cd lib
nata /Users/nata/Desktop/git/lib% cat hello.html
<!-- Author: Nefedova Natalia N.(1132226537@pfur.ru) -->
<html>
<head>
<link type="text/css" rel="stylesheet" media="all" href="style.css" />
</head>
<body>
<h1>Hello, World!</h1>
</body>
</html>
```

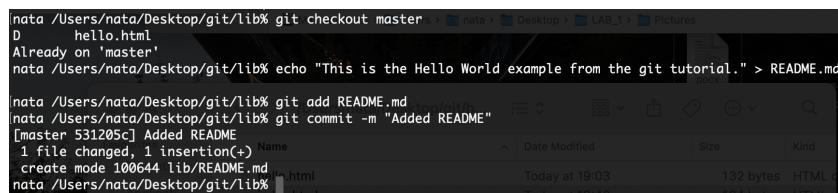
Рис. 23.7: Вернемся к ветке style

#1.18 Изменения в ветке master Пока вы меняли ветку style, кто-то решил обновить ветку master. Они добавили файл README.md.

1.18.1 Создайте файл README в ветке master

Выполните: git checkout master Создайте файл README.md echo "This is the Hello World example from the git tutorial." > README.md

#1.19 Сделайте коммит изменений README.md в ветку master.



```
nata /Users/nata/Desktop/git/lib% git checkout master
D hello.html
Already on 'master'
nata /Users/nata/Desktop/git/lib% echo "This is the Hello World example from the git tutorial." > README.md
nata /Users/nata/Desktop/git/lib% git add README.md
nata /Users/nata/Desktop/git/lib% git commit -m "Added README"
[master 531205c] Added README
1 file changed, 1 insertion(+)
create mode 100644 lib/README.md
```

Рис. 23.8: Сделаем коммит изменений README.md

1.19.1 Просмотр отличающихся веток 1.19.2 Просмотрите текущие ветки Теперь у нас в репозитории есть две отличающиеся ветки. Используйте следующую лог-команду для просмотра веток и их отличий. Выполните: `git log --graph --all`

```
nata /Users/nata/Desktop/git/lib% git log --graph --all
* commit 531205cb1a218cb541ba299002149b892c9a0da (HEAD -> master)
| Author: nnnefedova <1132226537@pfur.ru>
| Date:   Sat Feb 15 19:42:30 2025 +0300
| 
|     Added README
| 
| * commit 2571c3d3424c393ca92f2518db0cf1117377113d
| Author: nnnefedova <1132226537@pfur.ru>
| Date:   Sat Feb 15 19:33:07 2025 +0300
| 
|     Аддитивные коммиты с помощью простых ASCII
| master |     Hello uses style.css
| html». | 
| УМОЛЧАНИЕ показывается ТОЛЬКО
* commit 36b7befc34d7e519226db176eb9d16dfa3a0fe
| Author: nnnefedova <1132226537@pfur.ru>
| Date:   Sat Feb 15 19:28:36 2025 +0300
| 
|     Added css stylesheet
| 
* commit 0eca96fce814efde04766883bf5adf1856da5f85
| Author: nnnefedova <1132226537@pfur.ru>
| Date:   Sat Feb 15 13:30:17 2025 +0300
| 
|     Initial Commit
```

Рис. 23.9: Просмотр отличающихся веток

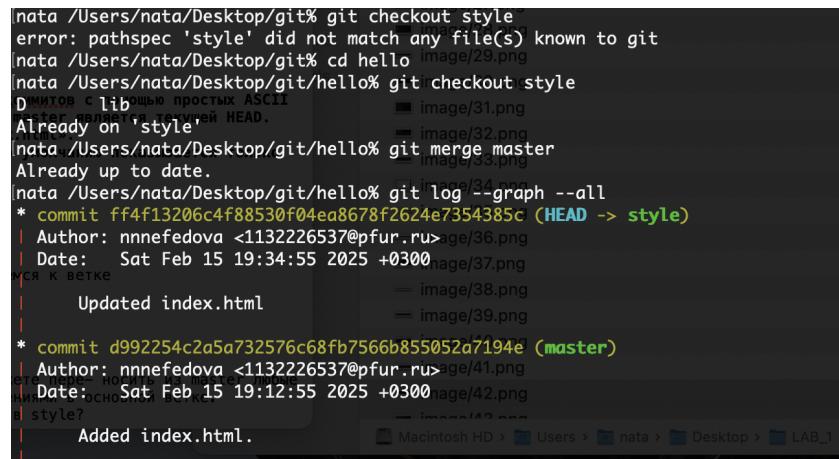
Добавление опции `-graph` в `git log` вызывает построение дерева коммитов с помощью простых ASCII символов. Мы видим обе ветки (`style` и `master`), и то, что ветка `master` является текущей `HEAD`. Общим предшественником обеих веток является коммит «`Added index.html`». Опция `-all` гарантированно означает, что мы видим все ветки. По умолчанию показывается только текущая ветка.

24 1.20 Слияние

1.20.1 Слияние веток Слияние переносит изменения из двух веток в одну.

Давайте вернемся к ветке style и сольем master с style. Выполните:

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



```
nata /Users/nata/Desktop/git% git checkout style
error: pathspec 'style' did not match any file(s) known to git
nata /Users/nata/Desktop/git% cd hello
nata /Users/nata/Desktop/git/hello% git checkout style
ДИПЛОМЫ С lib
Already on 'style'.
nata /Users/nata/Desktop/git/hello% git merge master
Already up to date.
nata /Users/nata/Desktop/git/hello% git log --graph --all
* commit ff4f13206c4f88530f04ea8678f2624e7354385c (HEAD -> style)
| Author: nnnefedova <1132226537@pfur.ru>
| Date:   Sat Feb 15 19:34:55 2025 +0300
|       Updated index.html
|
* commit d992254c2a5a732576c68fb7566b855052a7194e (master)
| Author: nnnefedova <1132226537@pfur.ru>
| Date:   Sat Feb 15 19:12:55 2025 +0300
|       Added index.html.
|
Macintosh HD > Users > nata > Desktop > LAB_1
```

Рис. 24.1: Слияние веток

Путем периодического слияния ветки master с веткой style вы можете переносить из master любые изменения и поддерживать совместимость изменений style с изменениями в основной ветке. Но что если изменения в ветке master конфликтуют с изменениями в style?

25 1.21 Создание конфликта

1.21.1 Вернитесь в master и создайте конфликт
Вернитесь в ветку master и внесите следующие изменения:

```
git checkout master
```

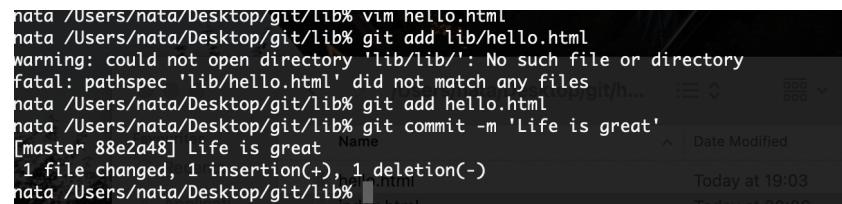
Файл lib/hello.html

Hello, World! Life is great!

Выполните:

```
git add lib/hello.html
```

```
git commit -m 'Life is great'
```



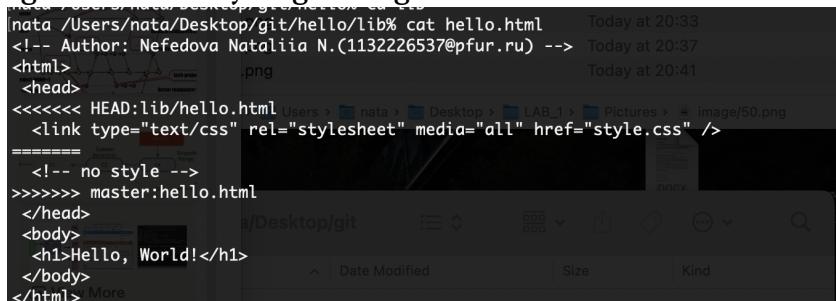
```
nata /Users/nata/Desktop/git/lib% vim hello.html
nata /Users/nata/Desktop/git/lib% git add lib/hello.html
warning: could not open directory 'lib/lib/': No such file or directory
fatal: pathspec 'lib/hello.html' did not match any files
nata /Users/nata/Desktop/git/lib% git add hello.html
nata /Users/nata/Desktop/git/lib% git commit -m 'Life is great'
[master 88e2a48] Life is great
 1 file changed, 1 insertion(+), 1 deletion(-)
nata /Users/nata/Desktop/git/lib%
```

Рис. 25.1: Создание конфликта

26 1.22 Разрешение конфликтов

1.22.1 Слияние master с веткой style Теперь вернемся к ветке style и попытаемся объединить ее с новой веткой master. Выполните:

```
-git checkout style -git merge master
```



```
nata /Users/nata/Desktop/git% cat hello.html      Today at 20:33
<!-- Author: Nefedova Natalia N.(1132226537@pfur.ru) --> Today at 20:37
<html>
  <head>
    .png
    <<<<< HEAD:lib/hello.html Users > nata > Desktop > LAB_1 > Pictures > image/50.png
      <link type="text/css" rel="stylesheet" media="all" href="style.css" />
    =====
      <!-- no style -->
    >>>>> master:hello.html
    </head>
    <body>
      <h1>Hello, World!</h1>
    </body>
  </html>
```

1.22.2 Решение конфликта



```
nata /Users/nata/Desktop/git% git commit -m "Merged master fixed conflict."
[master a799bde] Merged master fixed conflict.
 1 file changed, 1 insertion(+), 1 deletion(-)
```

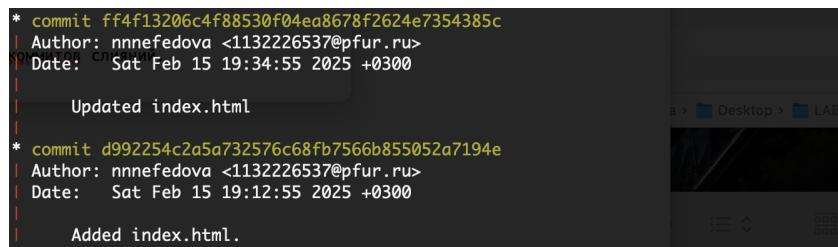
Рис. 26.1: Решение конфликта

1.22.4 Перебазирование как альтернатива слиянию

27 1.23 Сброс ветки style

1.23.1 Сброс ветки style Вернемся на ветку style к точке перед тем, как мы слили ее 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. Выполните:

```
git checkout style git log --graph
```



```
* commit ff4f13206c4f88530f04ea8678f2624e7354385c
| Author: nnnefedova <1132226537@pfur.ru>
| Date:   Sat Feb 15 19:34:55 2025 +0300
|
|     Updated index.html
|
* commit d992254c2a5a732576c68fb7566b855052a7194e
| Author: nnnefedova <1132226537@pfur.ru>
| Date:   Sat Feb 15 19:12:55 2025 +0300
|
|     Added index.html.
```

Рис. 27.1: Сброс ветки style

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

```
git reset --hard
```

1.23.2 Проверьте ветку. Поиските лог ветки style. У нас в истории больше нет коммитов слияний. Выполните:

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

28 1.24 Сброс ветки master

1.24.1 Сброс ветки master Добавив интерактивный режим в ветку master, мы внесли изменения, конфлик- тующие с изменениями в ветке style. Давайте вернемся в ветку master в точку перед внесением конфликтующих изменений. Это позволяет нам продемонстри- ровать работу команды git rebase, не беспокоясь о конфликтах. Выполните:

git checkout master
git log –graph
Коммит «Added README» идет непосредствен-но перед коммитом конфлик- тующего интерактивного режима. Мы сбросим ветку master к коммиту «Added README». Выполните:

```
git reset –hard
```



The screenshot shows a terminal window with two panes. The left pane displays the command history and its output:

```
* commit 36b7befc3c4df7e519226db176eb9d16dfa3a0fe
| Author: nnnefedova <1132226537@pfur.ru>
| Date: Sat Feb 15 19:28:36 2025 +0300
|
|     Added css stylesheet
: Полнотью удал
* commit 0eca96fce814efde04766883bf5adf1856da5f85
nata /Users/nata/Desktop/git/lib% git reset --hard 531205cb1a21
HEAD is now at 531205c Added README
nata /Users/nata/Desktop/git/lib% hello.html
```

The right pane shows a file browser window titled 'Desktop' with a single file named 'hello.html'.

Рис. 28.1: Сброс ветки master

git log –graph –all Просмотрите лог. Он должен выглядеть, как будто репозито-рий был перемотан назад во времени к точке до какого-либо слияния.

29 1.25 Перебазирование

```
git checkout style git rebase master git log --graph
```

```
nata ~/Users/nata/Desktop/git% git rebase master
Current branch master is up to date.
nata ~/Users/nata/Desktop/git% git log --graph
* commit 531205cb1a218db541ba299002149b892c9a0da (HEAD -> master)
| Author: nnnefedova <1132226537@pfur.ru>
| Date:  Sat Feb 15 19:42:30 2025 +0300
| 
|   Added README
|   УМНИЧАТЕЛЬНЫЕ ИЗМЕНЕНИЯ
|
* commit 2571c3d3424c393ca92f2518db0cf1117377113d
| Author: nnnefedova <1132226537@pfur.ru>
| Date:  Sat Feb 15 19:33:07 2025 +0300
| 
|   Hello uses style.css
|
* commit 36b7befbe3c4df7e519226db176beb016dfa3a0fe
| Author: nnnefedova <1132226537@pfur.ru>
| Date:  Sat Feb 15 19:28:36 2025 +0300
| 
|   Cannot rebase: your Index contains uncommitted changes...
|   Было внесено изменение
|   Added css stylesheet
|
* commit 0eca96fce814fede04766883bf5adf1856da5f85
| Author: nnnefedova <1132226537@pfur.ru>
| Date:  Sat Feb 15 13:30:17 2025 +0300
|
СМОЖЕТЕ ВЫПОЛНИТЬ: git rebase
    Initial Commit
nata ~/Users/nata/Desktop/git%
```

Рис. 29.1: Перебазирование

```
nata /Users/nata/Desktop/git/hello% git checkout style
Switched to branch 'style'
nata /Users/Desktop/git/hello% git rebase master
Successfully rebased and updated refs/heads/style.
nata /Users/nata/Desktop/git/hello% git log --graph
* commit caaaf289fc8274803915265adf6fa9f2e9f4506d1 (HEAD -> style)
| Author: nnnefedova <1132226537@pfur.ru>
| Date:  Sat Feb 15 21:51:56 2025 +0300
|           image/56.png
| Updated index.html
|
* commit 6d54ca18e9315bb6ec141f18328f1bdf52175755 LAB_1_> Pictures > image/56.png
| Author: nnnefedova <1132226537@pfur.ru>
| Date:  Sat Feb 15 21:50:56 2025 +0300
|
| Hello uses style.css
|
* commit e78061dae0df86df7efa70c6ef5ba9f84e7c2d7c
| Author: nnnefedova <1132226537@pfur.ru>
| Date:  Sat Feb 15 21:49:07 2025 +0300
```

Рис. 29.2: Перебазирование 2

1.25.1 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.

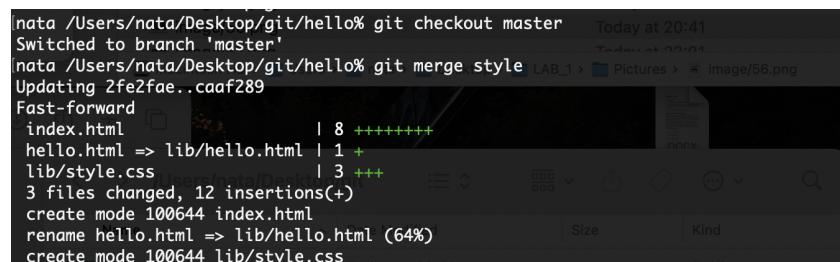
Учитывая приведенные выше рекомендации, рекомендуется использовать git rebase для кратковременных, локальных веток, а слияние для веток в публичном репозитории.

1.26 Слияние в ветку master

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

1.26.1 Слияние style в master Выполните:

```
git checkout master git merge style
```



```
nata /Users/nata/Desktop/git/hello% git checkout master          Today at 20:41
Switched to branch 'master'
nata /Users/nata/Desktop/git/hello% git merge style           Today at 20:41
Updating 2fe2fae..caaf289
Fast-forward
  index.html    | 8 ++++++++
  hello.html   | 1 +
  lib/style.css| 3 ===
3 files changed, 12 insertions(+)
create mode 100644 index.html
rename hello.html => lib/hello.html (64%)
create mode 100644 lib/style.css
```

Рис. 29.3: Слияние в ветку master

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

1.26.2 Просмотрите логи Выполните:

```
git log Теперь ветки style и master идентичны.
```

```
nata /Users/nata/Desktop/git/hello% git log
commit caaf289fc8274803915265adf6fa9f2e9f4506d1 (HEAD -> master, style)
Author: nnnefedova <1132226537@pfur.ru>
Date:   Sat Feb 15 21:51:56 2025 +0300
        └── image/44.png
    Updated index.html
        └── image/49.png
commit 6d54ca18e9315bb6ec141f1832&f1bdf52175755
Author: nnnefedova <1132226537@pfur.ru>
Date:   Sat Feb 15 21:50:56 2025 +0300
        └── image/49.png
Hello uses style.css
commit e78061dae0df86df7efa70c6ef5ba9f84e7c2d7c
Author: nnnefedova <1132226537@pfur.ru>
Date:   Sat Feb 15 21:49:07 2025 +0300
        └── image/54.png
Added css stylesheet
commit f51b8efa920b03237c20f399f81fa9a5fd0406b
Author: nnnefedova <1132226537@pfur.ru>
Date:   Sat Feb 15 21:41:50 2025 +0300
        Added index.html.

commit 856a879431ec74eb2b7d76a07ca89c3c59512e98
Author: nnnefedova <1132226537@pfur.ru>
Date:   Sat Feb 15 21:40:39 2025 +0300
```

Рис. 29.4: Просмотр логов

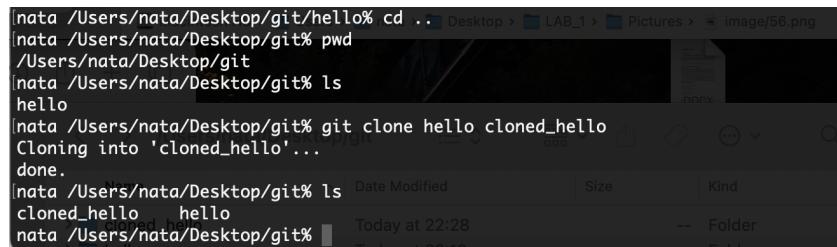
30 1.27 Клонирование репозиторием

1.27.1 Перейдите в рабочий каталог
Перейдите в рабочий каталог и сделайте клон вашего репозитория hello. Выполните:

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».

1.27.2 Создайте клон репозитория hello Создадим клон репозитория. Выполните:

git clone hello cloned_hello ls В вашем рабочем каталоге теперь должно быть два репозитория: оригинальный репозиторий «hello» и клонированный репозиторий «cloned_hello»



The screenshot shows a terminal window on a Mac OS X desktop. The window title is 'Terminal'. The command history shows:

```
[nata /Users/nata/Desktop/git/hello% cd .. Desktop > LAB_1 > Pictures > image/06.png
[nata /Users/nata/Desktop/git% pwd
/Users/nata/Desktop/git
[nata /Users/nata/Desktop/git% ls
hello
[nata /Users/nata/Desktop/git% git clone hello cloned_hello
Cloning into 'cloned_hello'...
done.
[nata /Users/nata/Desktop/git% ls
```

Below the terminal window, a file browser window is visible, showing a folder named 'cloned_hello' containing a single item named 'hello'. The file browser has columns for Name, Date Modified, Size, and Kind.

Рис. 30.1: Клонирование репозитория

31 1.28 Просмотр колонизованного репозитория

1.28.1 Давайте взглянем на клонированный репозиторий. Выполните:

cd cloned_hello ls Вы увидите список всех файлов на верхнем уровне оригинального репозитория README.md, index.html и lib.

```
|nata /Users/nata/Desktop/git% cd cloned_hello
|nata /Users/nata/Desktop/git/cloned_hello% ls
README.md      index.html      lib
nata /Users/nata/Desktop/git/cloned_hello%
```

Рис. 31.1: Клонированный репозиторий

1.28.2 Просмотрите историю репозитория Выполните:

git log –all Вы увидите 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.

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

```
nata ~/Users/nata/Desktop/git/cloned_hello% git log --all   today at 19:24
commit caaf289fc8274803915265adf6fa9f2e9f4506d1 (HEAD -> master, origin/style, origin/master,
Author: nnnefedova <1132226537@pfur.ru>
Date:   Sat Feb 15 21:51:56 2025 +0300
        - image/40.png
        Updated index.html4.png
        - image/45.png
commit 6d54ca18e9315bb6ec141f18328f1bdf52175755
Author: nnnefedova <1132226537@pfur.ru>
Date:   Sat Feb 15 21:50:56 2025 +0300
        - image/40.png
        Hello uses style.css
        - image/30.png
commit e78061daef86df7efa70c6ef5ba9f84e7c2d7c
Author: nnnefedova <1132226537@pfur.ru>
Date:   Sat Feb 15 21:49:07 2025 +0300
        - image/30.png
        Added css stylesheet
commit f51b8efa920b03237c20f399f81fa9a5fdd0406b
Author: nnnefedova <1132226537@pfur.ru>
Date:   Sat Feb 15 21:41:50 2025 +0300
        - image/30.png
        Added index.html.

Macintosh-HD:~/Users/nata/Desktop/git/cloned_hello nata%
```

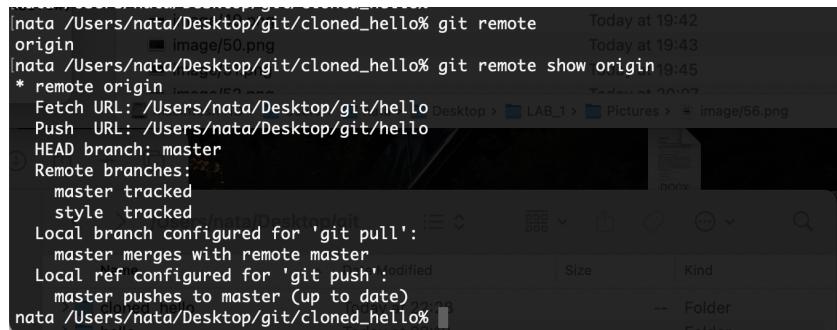
Рис. 31.2: Все логи

32 1.29 Что такое origin?

Выполните:

`git remote` Мы видим, что клонированный репозиторий знает об имени по умолчанию удаленного репозитория. Давайте посмотрим, можем ли мы получить более по-дробную информацию об имени по умолчанию: Выполните:

`git remote show origin` Удаленные репозитории обычно размещаются на отдельной машине, возможно, но, 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).



The screenshot shows a terminal window with the following content:

```
nata /Users/nata/Desktop/git/cloned_hello% git remote      Today at 19:42
origin          image/50.png
nata /Users/nata/Desktop/git/cloned_hello% git remote show origin 9:45
* remote origin
  Fetch URL: /Users/nata/Desktop/git/hello
  Push URL:  /Users/nata/Desktop/git/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)
nata /Users/nata/Desktop/git/cloned_hello%
```

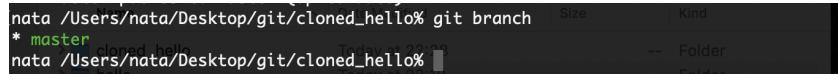
Рис. 32.1: Удаленные репозитории

33 1.30 Удаленные ветки

Давайте посмотрим на ветки, доступные в нашем клонированном репозитории.

Выполните:

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

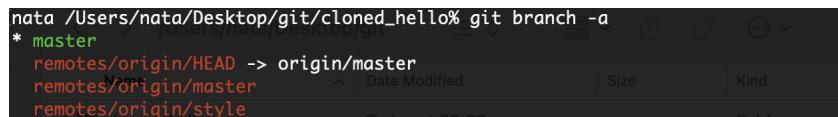


```
nata /Users/nata/Desktop/git/cloned_hello% git branch
* master
nata /Users/nata/Desktop/git/cloned_hello%
```

Рис. 33.1: `git branch` - просмотр локальных веток

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

`git branch -a` Git выводит все коммиты в оригинальный репозиторий, но ветки в удаленном репозитории не рассматриваются как локальные. Если мы хотим собственную ветку `style`, мы должны сами ее создать. Через минуту вы увидите, как это делается.



```
nata /Users/nata/Desktop/git/cloned_hello% git branch -a
* master
  remotes/origin/HEAD -> origin/master
  remotes/origin/master
  remotes/origin/style
nata /Users/nata/Desktop/git/cloned_hello%
```

Рис. 33.2: `git branch -a` просмотр всех веток

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

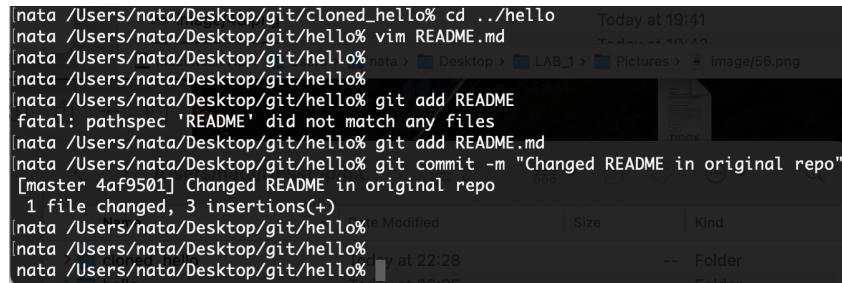
34 1.31 Внесите изменения в оригинальный репозиторий hello

Выполните:

cd ..//hello Примечание: Сейчас мы находимся в репозитории hello Внесите следующие изменения в файл README.md: Файл README.md

This is the Hello World example from the git tutorial. Теперь добавьте это изменение и сделайте коммит Выполните:

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.



The screenshot shows a terminal window with the following command history:

```
[nata /Users/nata/Desktop/git/cloned_hello% cd ..//hello          Today at 19:41
[nata /Users/nata/Desktop/git/hello% vim README.md
[nata /Users/nata/Desktop/git/hello%                                     Desktop > nata > Desktop > LAB_1 > Pictures > image/56.png
[nata /Users/nata/Desktop/git/hello%
[nata /Users/nata/Desktop/git/hello% git add README
fatal: pathspec 'README' did not match any files
[nata /Users/nata/Desktop/git/hello% git add README.md
[nata /Users/nata/Desktop/git/hello% git commit -m "Changed README in original repo"
[master 4af9501] Changed README in original repo
  1 file changed, 3 insertions(+)
[nata /Users/nata/Desktop/git/hello% git log --oneline
[nata /Users/nata/Desktop/git/hello%
[nata /Users/nata/Desktop/git/hello%                                     at 22:28
[nata /Users/nata/Desktop/git/hello% -- Folder
```

Рис. 34.1: Внесите изменения в оригиналный репозиторий

1.31.2 Извлечение изменений Научиться извлекать изменения из удаленного репозитория. Выполните:

cd ..//cloned_hello git fetch git log -all

```

nata /Users/nata/Desktop/git/hello% cd ..cloned_hello
nata /Users/nata/Desktop/git/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 (from 0)
Unpacking objects: 100% (3/3), 357 bytes | 178.00 KiB/s, done.
From /Users/nata/Desktop/git/hello
 * [new branch] master      -> origin/master
nata /Users/nata/Desktop/git/cloned_hello% git log --all
commit 4af95014880a2f2b499dd7e5a5b297d45d59cce (origin/master, origin/HEAD)
Author: nnnefedova <1132226537@pfur.ru>
Date:   Sat Feb 15 22:36:22 2025 +0300
        image/46.png
    Changed README in original repo
        image/46.png
commit caaf289fc8274803915265adf6fa9f2e9f4506d1 (HEAD -> master, origin/style)
Author: nnnefedova <1132226537@pfur.ru>
Date:   Sat Feb 15 21:51:56 2025 +0300
        Hello uses style.css
        Updated index.html

commit 6d54ca18e9315bb6ec141f18328f1bdf52175755
Author: nnnefedova <1132226537@pfur.ru>
Date:   Sat Feb 15 21:50:56 2025 +0300

```

The terminal shows the output of `git fetch` from the cloned repository. It lists three commits: one from the `origin/master` branch (the original repository) and two from the `origin/style` branch (local changes). The commit from `origin/style` is the most recent. The file browser shows the contents of the cloned repository, including files like `index.html`, `style.css`, and `README.md`.

Рис. 34.2: Извлечение изменений

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 найдите коммит «Changed README in original repo». Обратите внимание, что коммит включает в себя коммиты «origin/master» и «origin/HEAD». Теперь давайте посмотрим на коммит «Updated index.html». Вы увидите, 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 удаленного репозитория, но не будет сливать их с вашими наработками в локальных ветках.

Мы можем продемонстрировать, что клонированный файл `README.md` не изменился. Выполните: `cat README`

```

nata /Users/nata/Desktop/git/cloned_hello% cat README.md
This is the Hello World example from the git tutorial.
nata /Users/nata/Desktop/git/cloned_hello%

```

The terminal shows the command `cat README.md` being run, which outputs the content of the `README.md` file. The file contains the text: "This is the Hello World example from the git tutorial."

Рис. 34.3: README.md не изменился

35 1.32 Слияние извлеченных изменений

1.32.1 Слейте извлеченные изменения в локальную ветку master Выполните:

```
git merge origin/master
```

1.32.2 Еще раз проверьте файл README.md Сейчас мы должны увидеть изменения. Выполните: cat README.md

The screenshot shows a terminal window with the following text:
nata /Users/nata/Desktop/git/cloned_hello% git merge origin/master
Updating caaf289..4af9501
Fast-forward
 README.md | 3 +++
 1 file changed, 3 insertions(+)
nata /Users/nata/Desktop/git/cloned_hello%
nata /Users/nata/Desktop/git/cloned_hello%
nata /Users/nata/Desktop/git/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.
nata /Users/nata/Desktop/git/cloned_hello%

Рис. 35.1: Слияние извлеченных изменений

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

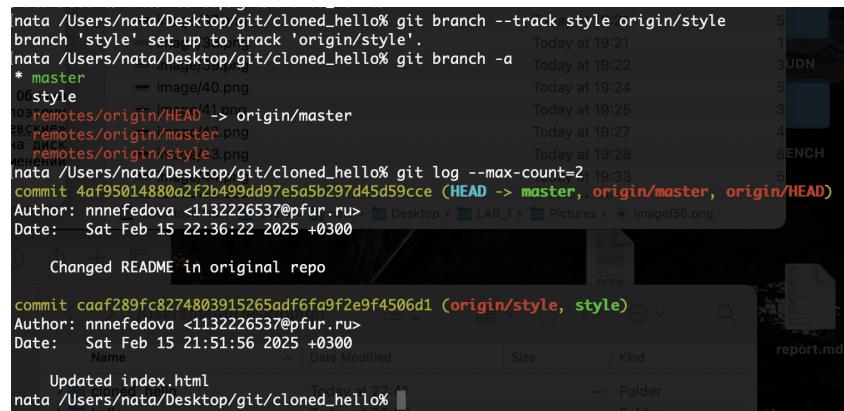
Теперь давайте рассмотрим объединение fetch и merge в одну команду. Выполнение: git pull эквивалентно двум следующим шагам: git fetch git merge origin/master

36 1.33 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 Выполните:

```
git branch -track style origin/style git branch -a git log --max-count=2
```



The screenshot shows a terminal window with the following command and its output:

```
nata /Users/nata/Desktop/git/cloned_hello% git branch --track style origin/style
branch 'style' set up to track 'origin/style'.
nata /Users/nata/Desktop/git/cloned_hello% git branch -a
* master
  style
  remotes/origin/HEAD -> origin/master
  remotes/origin/master
  remotes/origin/style
nata /Users/nata/Desktop/git/cloned_hello% git log --max-count=2
commit 4af95014880a2f2b499dd97e5a5b297d45d59cce (HEAD -> master, origin/master, origin/HEAD)
Author: nnnefedova <1132226537@pfur.ru>
Date:   Sat Feb 15 22:36:22 2025 +0300

    Changed README in original repo

commit caaf289fc8274803915265addf6fa9f2e9f4506d1 (origin/style, style)
Author: nnnefedova <1132226537@pfur.ru>
Date:   Sat Feb 15 21:51:56 2025 +0300

    Updated index.html
nata /Users/nata/Desktop/git/cloned_hello%
```

Рис. 36.1: Добавим локальную ветку

Теперь мы можем видеть ветку style в списке веток и логе

37 1.34 Чистые репозитории

Чистые репозитории (без рабочих каталогов) обычно используются для расширения. 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-репозиториевые» 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».

38 1.35 Создайте чистый репозиторий

cd .. git clone –bare hello hello.git Сейчас мы находимся в рабочем каталоге

```
nata /Users/nata/Desktop/git/cloned_hello% cd ..  
nata /Users/nata/Desktop/git% git clone --bare hello hello.git  
Cloning into bare repository 'hello.git'...  
done.  
nata /Users/nata/Desktop/git% ls hello.git  
HEAD      Name      description  ^ info  Modified    packed-refs  
config    hooks      objects      refs  
nata /Users/nata/Desktop/git%
```

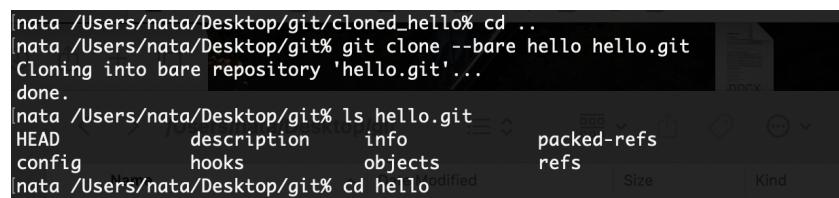
Рис. 38.1: Создайте чистый репозиторий

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

39 1.36 Добавление удаленного репозитория

Давайте добавим репозиторий hello.git к нашему оригинальному репозиторию.

```
cd hello git remote add shared ../hello.git
```



```
nata /Users/nata/Desktop/git/cloned_hello% cd ..  
nata /Users/nata/Desktop/git% git clone --bare hello hello.git  
Cloning into bare repository 'hello.git'...  
done.  
nata /Users/nata/Desktop/git% ls hello.git  
HEAD      description  info          packed-refs  
config    hooks        objects       refs  
nata /Users/nata/Desktop/git% cd hello
```

Рис. 39.1: Добавление удаленного репозитория

40 1.37 Отправка изменений

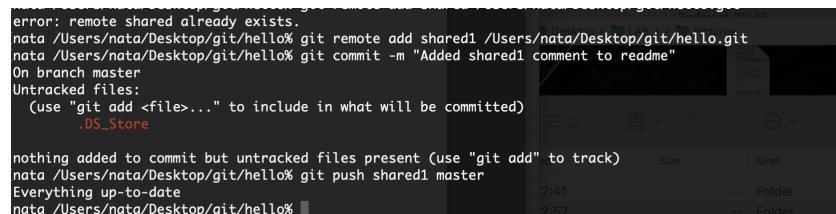
Так как чистые репозитории, как правило, расшариваются на каком-нибудь сетевом сервере, нам необходимо отправить наши изменения в другие репозитории. Начнем с создания изменения для отправки. Отредактируйте файл README.md и сделайте коммит Файл README.md:

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

Выполните:

git checkout master
git add README
git commit -m "Added shared comment to readme"
Теперь отправьте изменения в общий репозиторий. Выполните:

git push shared master
Общим называется репозиторий, получающий отправленные нами изменения.



The screenshot shows a terminal window with the following output:

```
error: remote shared already exists.
nata /Users/nata/Desktop/git/hello% git remote add shared1 /Users/nata/Desktop/git/hello.git
nata /Users/nata/Desktop/git/hello% git commit -m "Added shared1 comment to readme"
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)
    .DS_Store

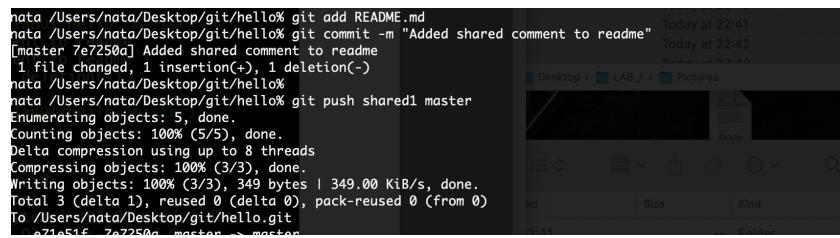
nothing added to commit but untracked files present (use "git add" to track)
nata /Users/nata/Desktop/git/hello% git push shared1 master
Everything up-to-date
nata /Users/nata/Desktop/git/hello%
```

Рис. 40.1: Отправка изменений в репозиторий

41 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. Выполните:

```
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
```



```
nata /Users/nata/Desktop/git/hello% git add README.md  
nata /Users/nata/Desktop/git/hello% git commit -m "Added shared comment to readme"  
[master 7e7250a] Added shared comment to readme  
1 file changed, 1 insertion(+), 1 deletion(-)  
nata /Users/nata/Desktop/git/hello% git push shared1 master  
Enumerating objects: 5, done.  
Counting objects: 100% (5/5), done.  
Delta compression using up to 8 threads  
Compressing objects: 100% (3/3), done.  
Writing objects: 100% (3/3), 349 bytes | 349.00 KiB/s, done.  
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)  
To /Users/nata/Desktop/git/hello.git  
    a71e51f..7e7250a master -> master
```

Рис. 41.1: Извлечение общих изменений

42 Выводы

По результатам выполнения работы нам удалось создать репозиторий и несколько веток в нем, объединить эти ветки, провести слияние, сделать коммиты и отменить коммиты, клонировать репозиторий, и поработать с его копией.

Список литературы

1. Newham C. LabNº1:git , markdown. RUDN, Dmitry S. Kulyabov, 2025. 39 c.