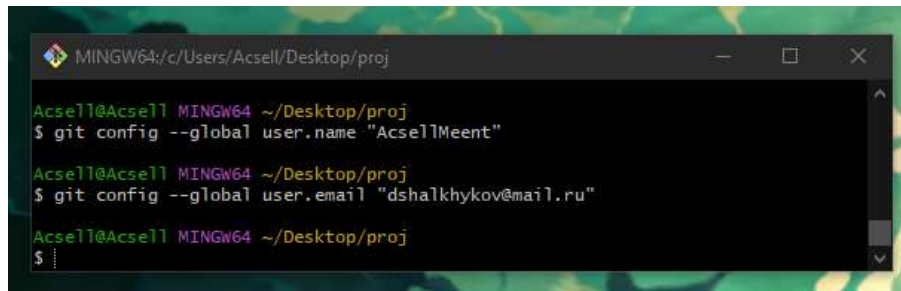


УП_1

Работа с Git

Подготовка

Если git никогда ранее не использовался пользователем, то необходимо установить имя и электронную почту. Для этого требуется выполнить команды, показанные на рисунке 1.



```
MINGW64/c/Users/Acsell/Desktop/proj
Acse11@Acse11 MINGW64 ~/Desktop/proj
$ git config --global user.name "Acse11Meent"
Acse11@Acse11 MINGW64 ~/Desktop/proj
$ git config --global user.email "dshalkhykov@mail.ru"
Acse11@Acse11 MINGW64 ~/Desktop/proj
$
```

Рисунок 1 - Установка имени и электронной почты

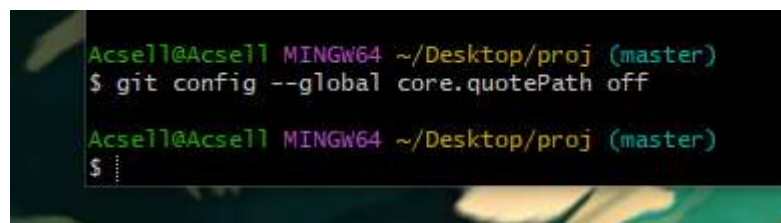
Затем нужно указать параметры установки окончаний строк (рисунок 2).



```
Acse11@Acse11 MINGW64 ~/Desktop/proj (master)
$ git config --global core.autocrlf true
Acse11@Acse11 MINGW64 ~/Desktop/proj (master)
$ git config --global core.safecrlf true
Acse11@Acse11 MINGW64 ~/Desktop/proj (master)
$
```

Рисунок 2 - Параметры окончаний строк

И последним пунктом идет установка отображения Unicode, показанная на рисунке 3.



```
Acse11@Acse11 MINGW64 ~/Desktop/proj (master)
$ git config --global core.quotePath off
Acse11@Acse11 MINGW64 ~/Desktop/proj (master)
$
```

Рисунок 3 - Установка отображения unicode

Создание проекта

Сначала нужно создать пустой каталог и внутри него файл hello.html (рисунок 4).



```
Acse11@Acse11 MINGW64 ~/Desktop
$ mkdir hello

Acse11@Acse11 MINGW64 ~/Desktop
$ cd hello/

Acse11@Acse11 MINGW64 ~/Desktop/hello
$ touch hello.html

Acse11@Acse11 MINGW64 ~/Desktop/hello
$
```

Рисунок 4 - Создание каталога и файла

После этого в файл необходимо ввести данные, например, «Hello, world», как показано на рисунке 5.

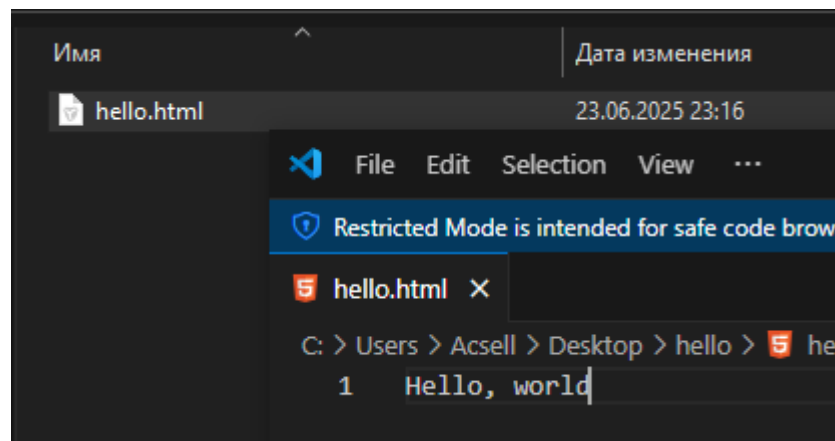
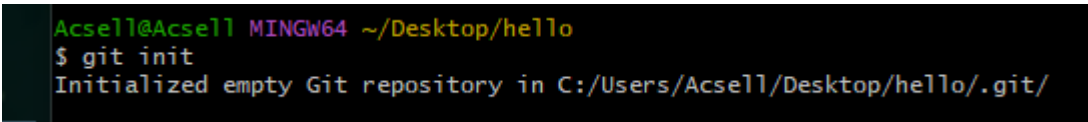


Рисунок 5 - Содержание файла


Для создания репозитория используется команда git init (рисунок 6).



```
Acse11@Acse11 MINGW64 ~/Desktop/hello
$ git init
Initialized empty Git repository in C:/Users/Acse11/Desktop/hello/.git/
```

Рисунок 6 - Создание репозитория

Для добавления страницы в репозиторий необходима команда git add (рисунок 7).



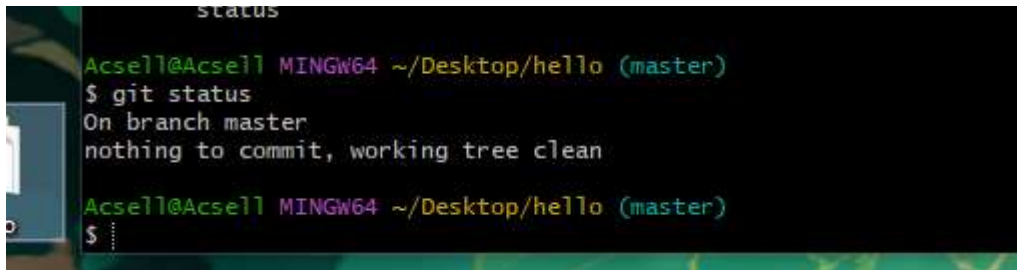
```
Acse11@Acse11 MINGW64 ~/Desktop/hello (master)
$ git add hello.html

Acse11@Acse11 MINGW64 ~/Desktop/hello (master)
$ git commit -m "First Commit"
[master (root-commit) 369b6f6] First Commit
1 file changed, 1 insertion(+)
create mode 100644 hello.html
```

Рисунок 7 - Добавление в репозиторий

Проверка состояния

Проверка состояния репозитория осуществляется с помощью команды `git status`. Если в репозитории хранится текущее состояние рабочего каталога и нет изменений, ожидающих записи, будет показано сообщение, как на рисунке 8.



```
status
Acse11@Acse11 MINGW64 ~/Desktop/hello (master)
$ git status
On branch master
nothing to commit, working tree clean
Acse11@Acse11 MINGW64 ~/Desktop/hello (master)
$
```

Рисунок 8 - Проверка состояния репозитория

Внесение изменений

Сначала необходимо внести изменения в файл (рисунок 9).

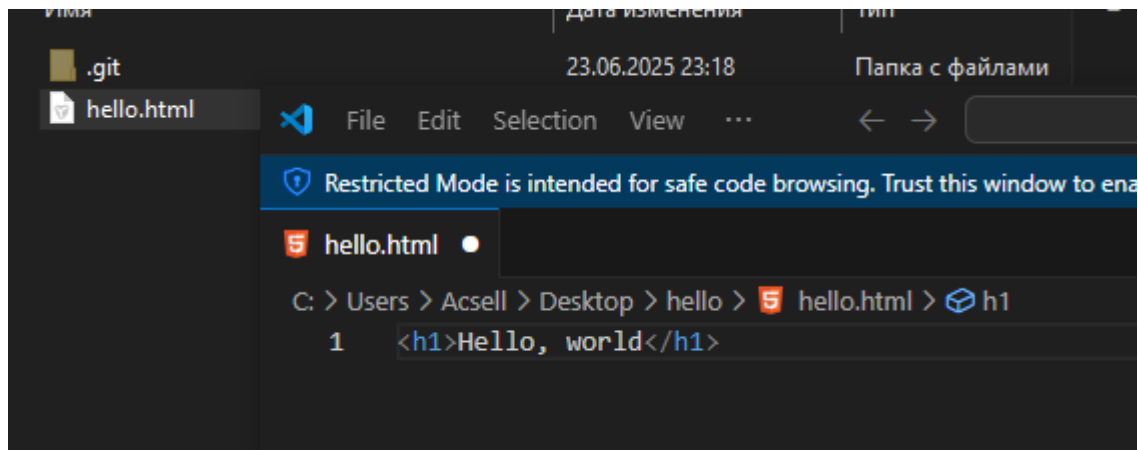
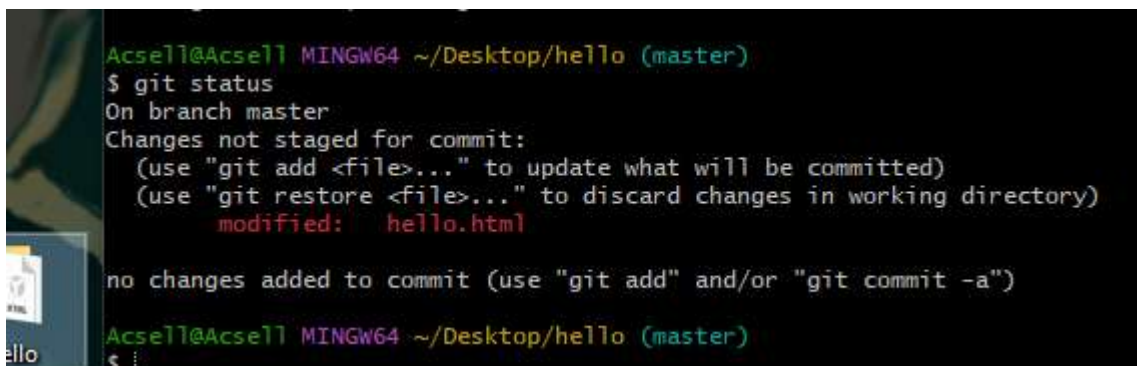


Рисунок 9 - Внесение изменений в файл

Если после предыдущего пункта осуществить проверку состояния репозитория, то будет показано данное сообщение (рисунок 10).



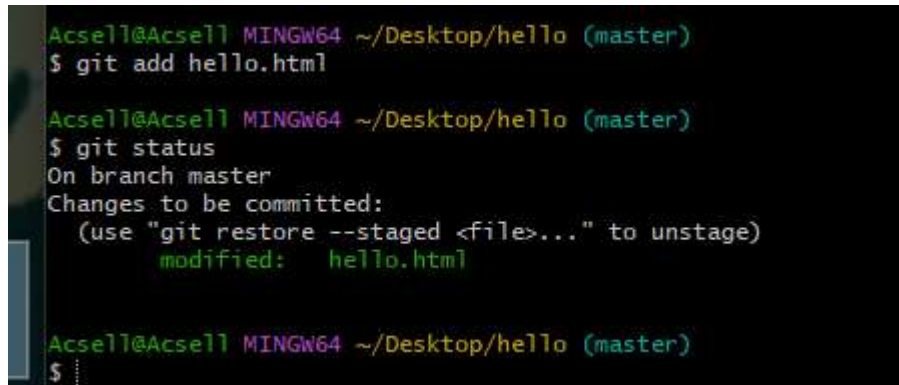
```
Acse11@Acse11 MINGW64 ~/Desktop/hello (master)
$ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   hello.html

no changes added to commit (use "git add" and/or "git commit -a")
Acse11@Acse11 MINGW64 ~/Desktop/hello (master)
$
```

Рисунок 10 - Сообщение о незафиксированных изменениях

Индексация изменений

Для того, чтобы проиндексировать изменения, нужно осуществить действия, показанные на рисунке 11.

A screenshot of a terminal window with a dark background and light-colored text. The prompt is 'Acse11@Acse11 MINGW64 ~/Desktop/hello (master)'. The first command is '\$ git add hello.html'. The second command is '\$ git status', which outputs 'On branch master' and 'Changes to be committed: (use "git restore --staged <file>..." to unstage) modified: hello.html'. The third command is '\$', which is partially visible.

```
Acse11@Acse11 MINGW64 ~/Desktop/hello (master)
$ git add hello.html

Acse11@Acse11 MINGW64 ~/Desktop/hello (master)
$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
       modified:   hello.html

Acse11@Acse11 MINGW64 ~/Desktop/hello (master)
$
```

Рисунок 11 - Команды для индексации изменений

После этого изменения файлы были проиндексированы. Это значит, что пока изменения не записаны в репозиторий. Если изменения позже не нужно будет фиксировать, то индексацию можно снять командой `git reset`.

Индексация и коммит

Можно зафиксировать изменения отдельными коммитами. Как это сделать, показано на рисунках 12-14.

.git	23.06.2025 23:20	Папка с файлами
a.html	23.06.2025 23:21	Yandex Browser H...
b.html	23.06.2025 23:21	Yandex Browser H...
hello.html	23.06.2025 23:19	Yandex Browser H...
v.html	23.06.2025 23:21	Yandex Browser H...

Рисунок 12 - Создано 3 файла

```
Acse11@Acse11 MINGW64 ~/Desktop/hello (master)
$ git add a.html

Acse11@Acse11 MINGW64 ~/Desktop/hello (master)
$ git add b.html

Acse11@Acse11 MINGW64 ~/Desktop/hello (master)
$ git commit -m "Changes for a and b"
[master 77e640d] Changes for a and b
3 files changed, 1 insertion(+), 1 deletion(-)
create mode 100644 a.html
create mode 100644 b.html

Acse11@Acse11 MINGW64 ~/Desktop/hello (master)
$ .....
```

Рисунок 13 - Индексация и коммит для 2 файлов

```
Acse11@Acse11 MINGW64 ~/Desktop/hello (master)
$ git add v.html ^C

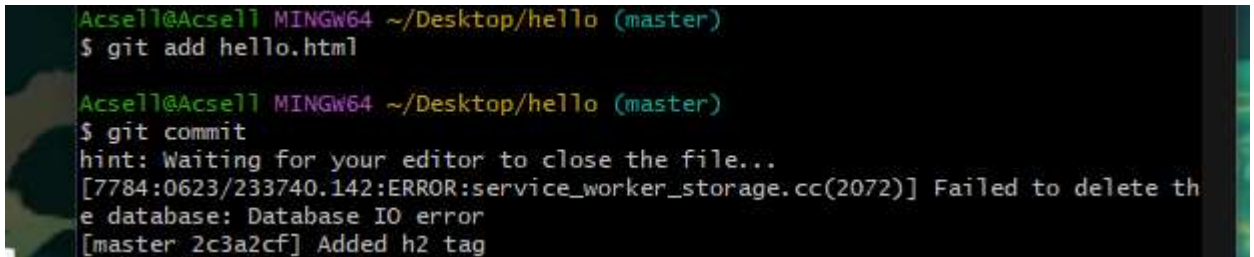
Acse11@Acse11 MINGW64 ~/Desktop/hello (master)
$ git commit -m "Unrelated change for v"
[master fcff9dc] Unrelated change for v
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 v.html

Acse11@Acse11 MINGW64 ~/Desktop/hello (master)
$ .....
```

Рисунок 14 - Индексация и коммит для третьего файла

Коммит изменений

Для того, чтобы редактировать комментарий коммита, нужно использовать команду `git commit` без метки `-m`.

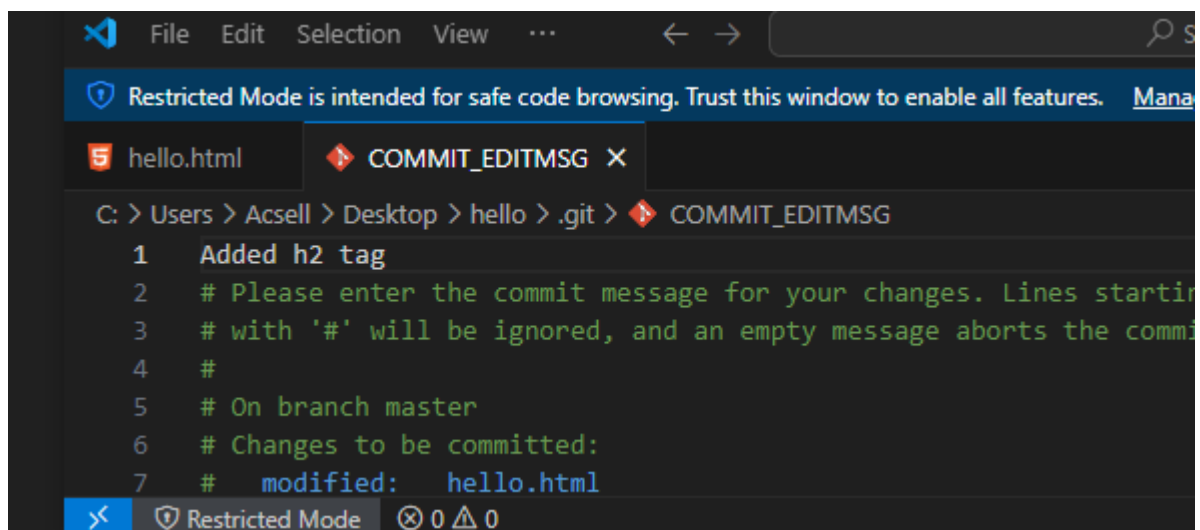


```
Acse11@Acse11 MINGW64 ~/Desktop/hello (master)
$ git add hello.html

Acse11@Acse11 MINGW64 ~/Desktop/hello (master)
$ git commit
hint: Waiting for your editor to close the file...
[7784:0623/233740.142:ERROR:service_worker_storage.cc(2072)] Failed to delete the database: Database IO error
[master 2c3a2cf] Added h2 tag
```

Рисунок 15 - Коммит изменений

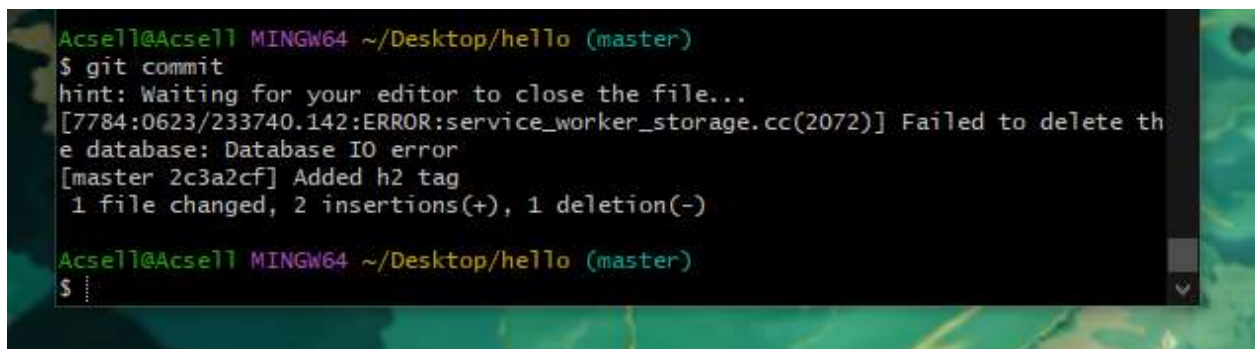
После этого будет открыт редактор, в котором необходимо на 1 строке записать комментарий коммита (рисунок 16).



```
File Edit Selection View ...
Restricted Mode is intended for safe code browsing. Trust this window to enable all features.
hello.html COMMIT_EDITMSG X
C:\> Users > Acse11 > Desktop > hello > .git > COMMIT_EDITMSG
1 Added h2 tag
2 # Please enter the commit message for your changes. Lines starting
3 # with '#' will be ignored, and an empty message aborts the commit.
4 #
5 # On branch master
6 # Changes to be committed:
7 #   modified:   hello.html
Restricted Mode 0 0
```

Рисунок 16 - Ввод комментария

После выхода из текстового редактора будет указано следующее сообщение (рисунок 17).

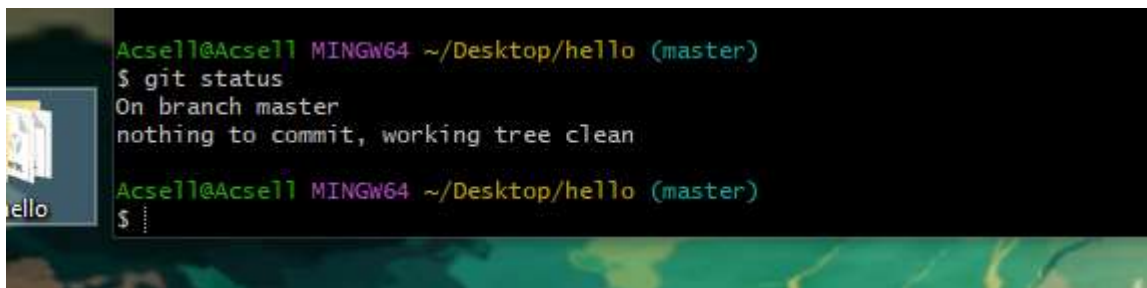


```
Acse11@Acse11 MINGW64 ~/Desktop/hello (master)
$ git commit
hint: Waiting for your editor to close the file...
[7784:0623/233740.142:ERROR:service_worker_storage.cc(2072)] Failed to delete the database: Database IO error
[master 2c3a2cf] Added h2 tag
1 file changed, 2 insertions(+), 1 deletion(-)

Acse11@Acse11 MINGW64 ~/Desktop/hello (master)
$
```

Рисунок 17 - Коммит-сообщения

После этого еще раз нужно проверить состояние репозитория (рисунок 18).

A terminal window with a dark background. The prompt is 'Acse11@Acse11 MINGW64 ~/Desktop/hello (master)'. The user enters '\$ git status'. The output is 'On branch master' and 'nothing to commit, working tree clean'. The prompt is repeated below the output.

```
Acse11@Acse11 MINGW64 ~/Desktop/hello (master)
$ git status
On branch master
nothing to commit, working tree clean

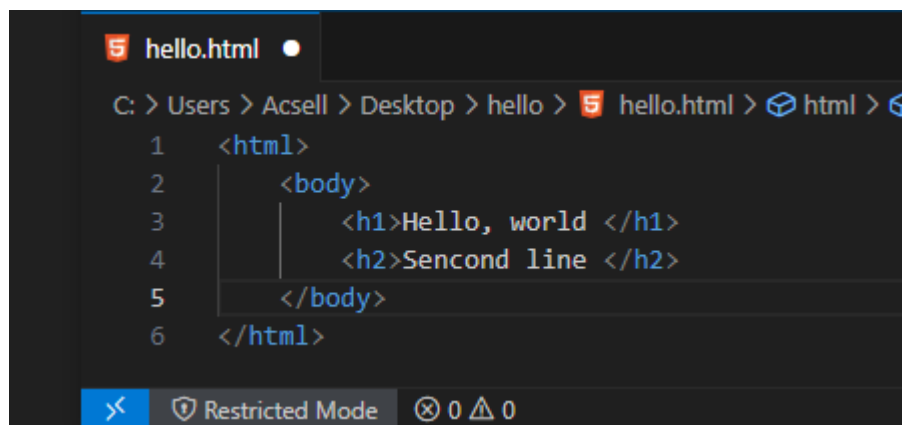
Acse11@Acse11 MINGW64 ~/Desktop/hello (master)
$
```

Рисунок 18 - Проверка состояния

Изменения, а не файлы

Для того, чтобы понять, что git фокусируется на изменениях в файле, а не на самом файле, можно проделать следующие действия.

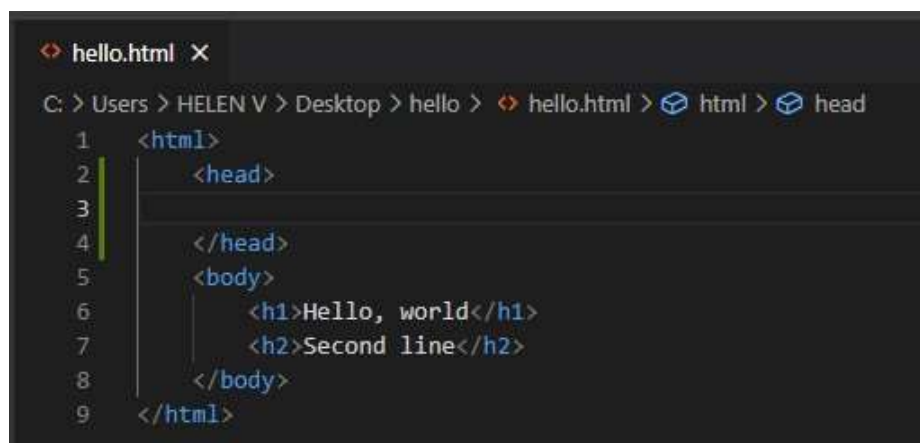
Сначала нужно в файл hello.html добавить теги html и body (рисунок 19), а затем проиндексировать изменения.

A code editor window titled 'hello.html'. The path bar shows 'C: > Users > Acse11 > Desktop > hello > hello.html > html'. The code contains an HTML structure with a body containing two headings.

```
1 <html>
2   <body>
3     <h1>Hello, world </h1>
4     <h2>Sencond line </h2>
5   </body>
6 </html>
```

Рисунок 19 - Добавление тегов html и body

Затем еще раз нужно добавить изменения в файл (добавить тег head), но изменения не индексировать (рисунок 20).

A code editor window titled 'hello.html'. The path bar shows 'C: > Users > HELEN V > Desktop > hello > hello.html > html > head'. The code shows the HTML structure with a new head tag added at the beginning of the body.

```
1 <html>
2   <head>
3
4   </head>
5   <body>
6     <h1>Hello, world</h1>
7     <h2>Second line</h2>
8   </body>
9 </html>
```

Рисунок 20 - Добавление тега head

Далее нужно проверить статус. На рисунке 21 видно, что файл hello.html указан дважды: первое изменение проиндексировано и готово к коммиту, а второе – нет.

```
Acse11@Acse11 MINGW64 ~/Desktop/hello (master)
$ 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

Acse11@Acse11 MINGW64 ~/Desktop/hello (master)
$
```

Рисунок 21 - Проверка состояния

Далее надо произвести коммит проиндексированного изменения и затем еще раз проверить состояние (рисунок 22).

```
Acse11@Acse11 MINGW64 ~/Desktop/hello (master)
$ git commit -m "Add standart HTML tags"
[master 666f580] Add standart HTML tags
1 file changed, 6 insertions(+), 2 deletions(-)

Acse11@Acse11 MINGW64 ~/Desktop/hello (master)
$ 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")

Acse11@Acse11 MINGW64 ~/Desktop/hello (master)
$
```

Рисунок 22 - Коммит 1 изменения и проверка состояния

Нужно добавить второе изменение в индекс и затем проверить состояние (рисунок 23).

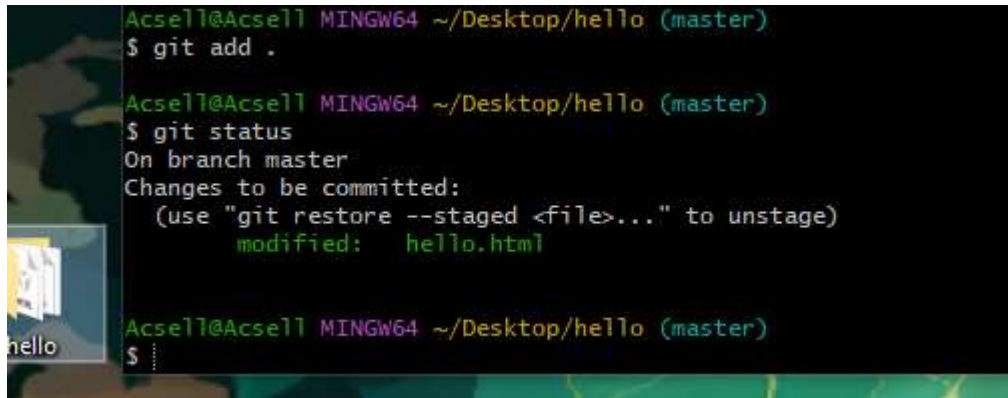
A screenshot of a terminal window with a dark background. The prompt is 'Acse11@Acse11 MINGW64 ~/Desktop/hello (master)'. The user enters '\$ git add .' and then '\$ git status'. The status output shows 'On branch master' and 'Changes to be committed: (use "git restore --staged <file>..." to unstage) modified: hello.html'. The prompt returns to '\$ '.

Рисунок 23 - Индексация 2 изменения и проверка состояния

После этого нужно сделать коммит второго изменения (рисунок 24).


A screenshot of a terminal window. The prompt is 'Acse11@Acse11 MINGW64 ~/Desktop/hello (master)'. The user enters '\$ git commit -m "Add HTML header"'. The output shows '[master bb11936] Add HTML header' and '1 file changed, 3 insertions(+)'.

Рисунок 24 - Коммит 2 изменения

УП_2

Работа с Git

История

Для того, чтобы просмотреть список произведенных изменений в проекте, используется команда git log (рисунок 1).

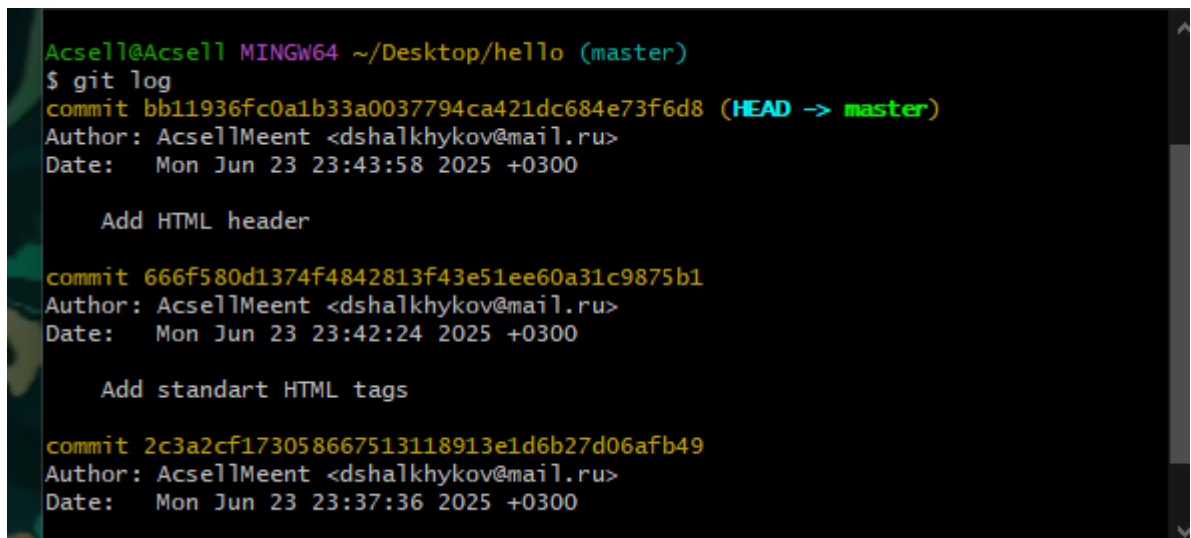
A screenshot of a terminal window showing the output of the 'git log' command. It lists three commits in reverse chronological order. The first commit is 'commit bb11936fc0a1b33a0037794ca421dc684e73f6d8 (HEAD -> master)' with the message 'Add HTML header'. The second is 'commit 666f580d1374f4842813f43e51ee60a31c9875b1' with the message 'Add standart HTML tags'. The third is 'commit 2c3a2cf173058667513118913e1d6b27d06afb49' with the message 'Add standart HTML tags'. Each entry includes the author 'Acse11Meent <dshalkhykov@mail.ru>' and the date 'Mon Jun 23 23:43:58 2025 +0300'.

Рисунок 25 - Просмотр истории изменений

На рисунке 1 была выведена полная история. Для того, чтобы увидеть однострочный формат используется команда `git log --pretty=oneline` (рисунок 2).

```
Acse11@Acse11 MINGW64 ~/Desktop/hello (master)
$ git log --pretty=oneline
bb11936fc0a1b33a0037794ca421dc684e73f6d8 (HEAD -> master) Add HTML header
666f580d1374f4842813f43e51ee60a31c9875b1 Add standart HTML tags
2c3a2cf173058667513118913e1d6b27d06afb49 Added h2 tag
fcff9dc042740210ea39700380ac107c98c0128a Unrelated change for v
77e640d94af8102252fc6b99179cc519a4dfa40a Changes for a and b
369b6f6343640d581447962f2dca24600563cbee First Commit

Acse11@Acse11 MINGW64 ~/Desktop/hello (master)
$
```

Рисунок 26 - Однострочный формат вывода

Далее на рисунках 3-8 показано несколько вариантов вывода истории изменений.

```
Acse11@Acse11 MINGW64 ~/Desktop/hello (master)
$ git log --pretty=oneline --max-count=2
bb11936fc0a1b33a0037794ca421dc684e73f6d8 (HEAD -> master) Add HTML header
666f580d1374f4842813f43e51ee60a31c9875b1 Add standart HTML tags
```

Рисунок 27 - Вывод последних 2 изменений

```
Acse11@Acse11 MINGW64 ~/Desktop/hello (master)
$ git log --pretty=oneline --since='10 days ago'
bb11936fc0a1b33a0037794ca421dc684e73f6d8 (HEAD -> master) Add HTML header
666f580d1374f4842813f43e51ee60a31c9875b1 Add standart HTML tags
2c3a2cf173058667513118913e1d6b27d06afb49 Added h2 tag
fcff9dc042740210ea39700380ac107c98c0128a Unrelated change for v
77e640d94af8102252fc6b99179cc519a4dfa40a Changes for a and b
369b6f6343640d581447962f2dca24600563cbee First Commit

Acse11@Acse11 MINGW64 ~/Desktop/hello (master)
$
```

Рисунок 28 - Вывод изменений начиная с определенного времени

```
Acse11@Acse11 MINGW64 ~/Desktop/hello (master)
$ git log --pretty=oneline --since='5 minuts ago'
bb11936fc0a1b33a0037794ca421dc684e73f6d8 (HEAD -> master) Add HTML header
666f580d1374f4842813f43e51ee60a31c9875b1 Add standart HTML tags
2c3a2cf173058667513118913e1d6b27d06afb49 Added h2 tag
fcff9dc042740210ea39700380ac107c98c0128a Unrelated change for v
77e640d94af8102252fc6b99179cc519a4dfa40a Changes for a and b
369b6f6343640d581447962f2dca24600563cbee First Commit
```

Рисунок 29 - Вывод изменений до определенного времени

```
Acse11@Acse11 MINGW64 ~/Desktop/hello (master)
$ git log --pretty=oneline --author='Acse11Meent'
bb11936fc0a1b33a0037794ca421dc684e73f6d8 (HEAD -> master) Add HTML header
666f580d1374f4842813f43e51ee60a31c9875b1 Add standart HTML tags
2c3a2cf173058667513118913e1d6b27d06afb49 Added h2 tag
fcff9dc042740210ea39700380ac107c98c0128a Unrelated change for v
77e640d94af8102252fc6b99179cc519a4dfa40a Changes for a and b
369b6f6343640d581447962f2dca24600563cbee First Commit
```

Рисунок 30 - Вывод изменений, внесенных определенным автором

```
Acse11@Acse11 MINGW64 ~/Desktop/hello (master)
$ git log --pretty=oneline --all
bb11936fc0a1b33a0037794ca421dc684e73f6d8 (HEAD -> master) Add HTML header
666f580d1374f4842813f43e51ee60a31c9875b1 Add standart HTML tags
2c3a2cf173058667513118913e1d6b27d06afb49 Added h2 tag
fcff9dc042740210ea39700380ac107c98c0128a Unrelated change for v
77e640d94af8102252fc6b99179cc519a4dfa40a Changes for a and b
369b6f6343640d581447962f2dca24600563cbee First Commit
```

Рисунок 31 - Вывод всех изменений

```
Acse11@Acse11 MINGW64 ~/Desktop/hello (master)
$ git log --pretty=format:"%h %cd %s (%an)" --since='12 days ago'
bb11936 Mon Jun 23 23:43:58 2025 +0300 Add HTML header (Acse11Meent)
666f580 Mon Jun 23 23:42:24 2025 +0300 Add standart HTML tags (Acse11Meent)
2c3a2cf Mon Jun 23 23:37:36 2025 +0300 Added h2 tag (Acse11Meent)
fcff9dc Mon Jun 23 23:23:39 2025 +0300 Unrelated change for v (Acse11Meent)
77e640d Mon Jun 23 23:22:47 2025 +0300 Changes for a and b (Acse11Meent)
369b6f6 Mon Jun 23 23:17:53 2025 +0300 First Commit (Acse11Meent)
```

Рисунок 32 - Использование нескольких параметров

Алиасы

Для настройки алиасов используется команда, показанная на рисунке 9.

```
Acse11@Acse11 MINGW64 ~/Desktop/hello (master)
$ git config --global alias.co checkout

Acse11@Acse11 MINGW64 ~/Desktop/hello (master)
$ git config --global alias.ci commit

Acse11@Acse11 MINGW64 ~/Desktop/hello (master)
$ git config --global alias.st status

Acse11@Acse11 MINGW64 ~/Desktop/hello (master)
$ git config --global alias.br branch

Acse11@Acse11 MINGW64 ~/Desktop/hello (master)
$ git config --global alias.hits "log --pretty=format:'%h %ad | %s%d [%an]' --graph --date=short"

Acse11@Acse11 MINGW64 ~/Desktop/hello (master)
$ git config --global alias.type 'cat-file -t'

Acse11@Acse11 MINGW64 ~/Desktop/hello (master)
$ git config --global alias.dump 'cat-file -p'

Acse11@Acse11 MINGW64 ~/Desktop/hello (master)
$
```

Рисунок 33 - Настройка алиасов для некоторых команд

При выполнении алиаса будет выполнена определенная команда и выведены нужные данные (рисунок 10).


```
Acse11@Acse11 MINGW64 ~/Desktop/hello (master)
$ git config --global alias.hist "log --pretty=format:'%h %ad | %s%d [%an]' --graph --date=short"

Acse11@Acse11 MINGW64 ~/Desktop/hello (master)
$ git hist
* bb11936 2025-06-23 | Add HTML header (HEAD -> master) [Acse11Meent]
* 666f580 2025-06-23 | Add standart HTML tags [Acse11Meent]
* 2c3a2cf 2025-06-23 | Added h2 tag [Acse11Meent]
* fcff9dc 2025-06-23 | Unrelated change for v [Acse11Meent]
* 77e640d 2025-06-23 | Changes for a and b [Acse11Meent]
* 369b6f6 2025-06-23 | First Commit [Acse11Meent]

Acse11@Acse11 MINGW64 ~/Desktop/hello (master)
$
```

Рисунок 34 - Выполнение алиаса hist

```
Acse11@Acse11 MINGW64 ~/Desktop/hello (master)
$ alias gs='git status'

Acse11@Acse11 MINGW64 ~/Desktop/hello (master)
$ gs
On branch master
nothing to commit, working tree clean

Acse11@Acse11 MINGW64 ~/Desktop/hello (master)
$
```

Рисунок 35 - Установка и выполнение алиаса gs

Получение старых версий

Для того, чтобы вернуть рабочий каталог к предыдущему состоянию, можно использовать следующий способ: для начала нужно узнать хэши предыдущих версий, что можно сделать с помощью ранее заданного алиаса hist (рисунок 12).

```
Acse11@Acse11 MINGW64 ~/Desktop/hello (master)
$ git hist
* bb11936 2025-06-23 | Add HTML header (HEAD -> master) [Acse11Meent]
* 666f580 2025-06-23 | Add standart HTML tags [Acse11Meent]
* 2c3a2cf 2025-06-23 | Added h2 tag [Acse11Meent]
* fcff9dc 2025-06-23 | Unrelated change for v [Acse11Meent]
* 77e640d 2025-06-23 | Changes for a and b [Acse11Meent]
* 369b6f6 2025-06-23 | First Commit [Acse11Meent]
```

Рисунок 36 - Просмотр хэшей предыдущих версий

Далее нужно выполнить команду git checkout с номером нужного хэша (достаточно первых 7 знаков). После этого можно просмотреть содержимое файла с помощью команды cat (рисунок 13).

```
Acse11@Acse11 MINGW64 ~/Desktop/hello (master)
$ git checkout 369b6f6
Note: switching to '369b6f6'.

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 369b6f6 First Commit
Acse11@Acse11 MINGW64 ~/Desktop/hello ((369b6f6...))
$ cat hello.html
Hello, world
Acse11@Acse11 MINGW64 ~/Desktop/hello ((369b6f6...))
$
```

Рисунок 37 - Возвращение к нужной версии и просмотр содержимого файла

Возвращение к последней версии в ветке master

Для возвращения к последней версии в ветке master (имя ветки по умолчанию) надо ввести команду `git checkout master`, что показано на рисунке 14.

```
Acse11@Acse11 MINGW64 ~/Desktop/hello ((369b6f6...))
$ git checkout master
Previous HEAD position was 369b6f6 First Commit
Switched to branch 'master'

Acse11@Acse11 MINGW64 ~/Desktop/hello (master)
$ cat hello.html
<html>
  <head>

  </head>
  <body>
    <h1>Hello, world </h1>
    <h2>Sencond line </h2>
  </body>
</html>
Acse11@Acse11 MINGW64 ~/Desktop/hello (master)
$
```

Рисунок 38 - Возвращение к последней версии в ветке master

Создание тегов версий

Для создания тега используется команда `git tag`. На рисунке 15 показано, тегом `ver1` была названа текущая версия страницы.

```
Acse11@Acse11 MINGW64 ~/Desktop/hello (master)
$ git tag ver1

Acse11@Acse11 MINGW64 ~/Desktop/hello (master)
$
```

Рисунок 39 - Задание тега

Чтобы перейти к предыдущей версии, можно использовать символ «^», который означает «родитель».

```
Acse11@Acse11 MINGW64 ~/Desktop/hello (master)
$ git checkout ver1^
Note: switching to 'ver1^'.

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 666f580 Add standart HTML tags

Acse11@Acse11 MINGW64 ~/Desktop/hello ((666f580...))
$ cat hello.html
<html>
  <body>
    <h1>Hello, world </h1>
    <h2>Sencond line </h2>
  </body>
</html>
Acse11@Acse11 MINGW64 ~/Desktop/hello ((666f580...))
$
```

Рисунок 40 – Переход к предыдущей версии с помощью тега

```
Acse11@Acse11 MINGW64 ~/Desktop/hello ((666f580...))
$ git tag ver1-beta

Acse11@Acse11 MINGW64 ~/Desktop/hello ((ver1-beta))
$
```

Рисунок 41 - Задание тега предыдущей версии

Теперь с помощью тегов можно переключаться между версиями (рисунок 18).


```
Acse11@Acse11 MINGW64 ~/Desktop/hello ((ver1-beta))
$ git checkout ver1
Previous HEAD position was 666f580 Add standart HTML tags
HEAD is now at bb11936 Add HTML header

Acse11@Acse11 MINGW64 ~/Desktop/hello ((ver1))
$ git checkout ver1-beta
Previous HEAD position was bb11936 Add HTML header
HEAD is now at 666f580 Add standart HTML tags

Acse11@Acse11 MINGW64 ~/Desktop/hello ((ver1-beta))
$
```

Рисунок 42 - Переключение между версиями с помощью тегов

Для просмотра всех тегов используется команда git tag (рисунок 19).

```
Acse11@Acse11 MINGW64 ~/Desktop/hello ((ver1-beta))
$ git tag
ver1
ver1-beta

Acse11@Acse11 MINGW64 ~/Desktop/hello ((ver1-beta))
$
```

Рисунок 43 - Просмотр тегов

Также можно посмотреть теги в логе, как показано на рисунке 20.

```
Acse11@Acse11 MINGW64 ~/Desktop/hello ((ver1-beta))
$ git hist master --all
* bb11936 2025-06-23 | Add HTML header (tag: ver1, master) [Acse11Meent]
* 666f580 2025-06-23 | Add standart HTML tags (HEAD, tag: ver1-beta) [Acse11Meent]
* 2c3a2cf 2025-06-23 | Added h2 tag [Acse11Meent]
* fcff9dc 2025-06-23 | Unrelated change for v [Acse11Meent]
* 77e640d 2025-06-23 | Changes for a and b [Acse11Meent]
* 369b6f6 2025-06-23 | First Commit [Acse11Meent]

Acse11@Acse11 MINGW64 ~/Desktop/hello ((ver1-beta))
$
```

Рисунок 44 - Просмотр тегов в логе

Отмена локальных изменений (до индексации)

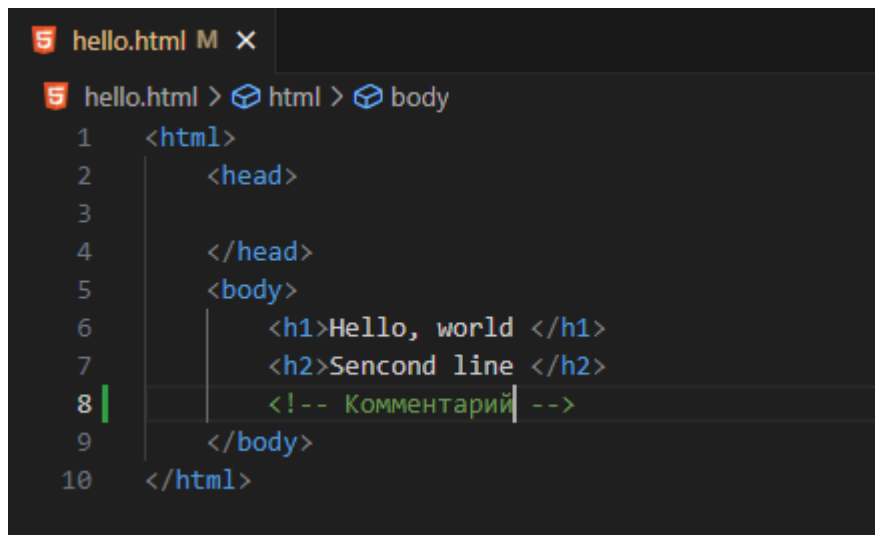
Сначала нужно переключиться на последний коммит master (рисунок 21).

```
Acse11@Acse11 MINGW64 ~/Desktop/hello ((ver1-beta))
$ git checkout master
Previous HEAD position was 666f580 Add standart HTML tags
Switched to branch 'master'

Acse11@Acse11 MINGW64 ~/Desktop/hello (master)
$
```

Рисунок 45 - Переключение на последний коммит

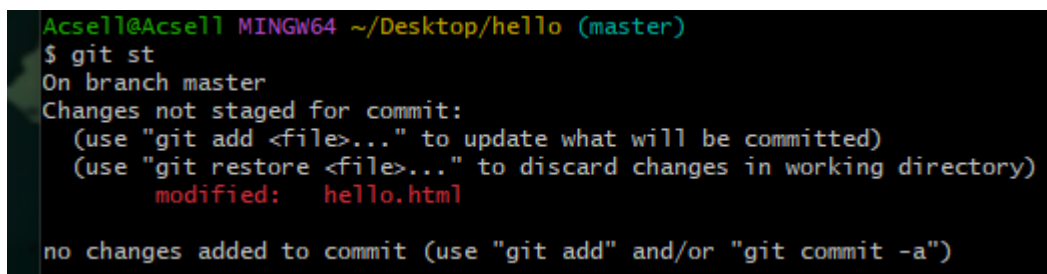
Далее для работы нужно внести изменение в файл (рисунок 22).



```
hello.html M X
hello.html > html > body
1 <html>
2   <head>
3
4   </head>
5   <body>
6     <h1>Hello, world </h1>
7     <h2>Sencond line </h2>
8     <!-- Комментарий -->
9   </body>
10 </html>
```

Рисунок 46 - Внесение изменения в файл

После выполнения команды `git status` будет показано, что есть не проиндексированное изменение (рисунок 23).

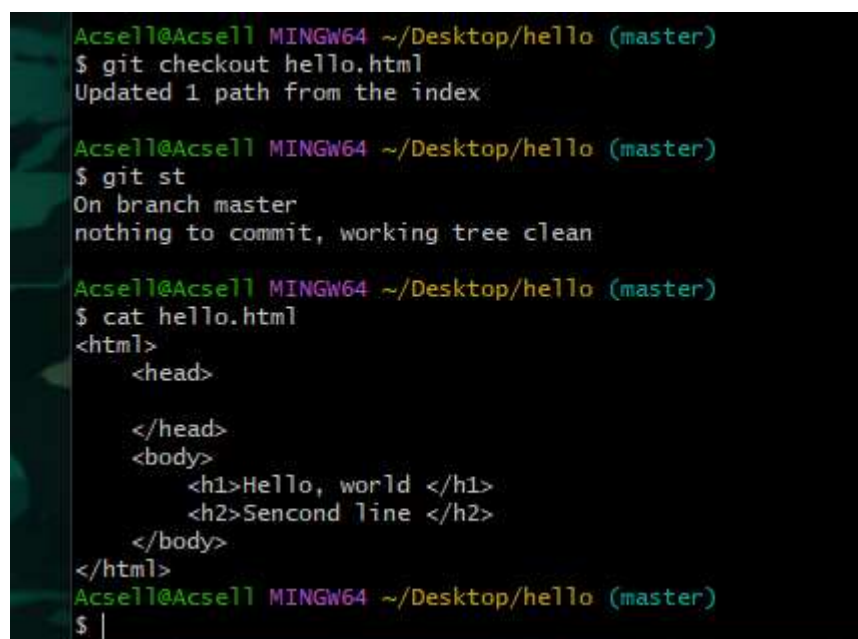


```
Acse11@Acse11 MINGW64 ~/Desktop/hello (master)
$ git st
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   hello.html

no changes added to commit (use "git add" and/or "git commit -a")
```

Рисунок 47 - Не проиндексированное изменение

Для переключения в версию файла без изменений используется команда `git checkout hello.html` (рисунок 24). Команда `git status` покажет, что не было произведено изменений, не зафиксированных в каталоге.



```
Acse11@Acse11 MINGW64 ~/Desktop/hello (master)
$ git checkout hello.html
Updated 1 path from the index

Acse11@Acse11 MINGW64 ~/Desktop/hello (master)
$ git st
On branch master
nothing to commit, working tree clean

Acse11@Acse11 MINGW64 ~/Desktop/hello (master)
$ cat hello.html
<html>
  <head>

  </head>
  <body>
    <h1>Hello, world </h1>
    <h2>Sencond line </h2>
  </body>
</html>

Acse11@Acse11 MINGW64 ~/Desktop/hello (master)
$ |
```

Рисунок 48 - Возвращение к версии

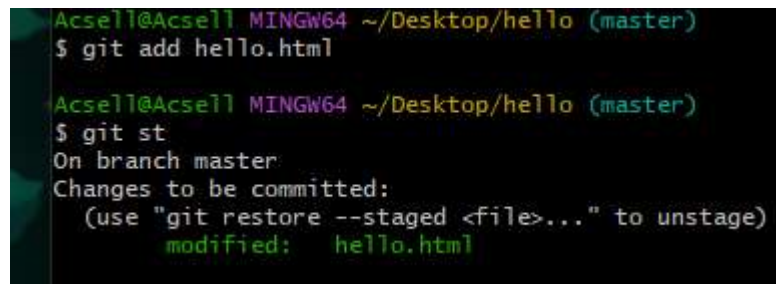
Отмена проиндексированных изменений (перед коммитом)

Для того, чтобы научиться отменять проиндексированные изменения, сначала нужно внести ненужное изменение в файл (рисунок 25). После этого производится индексация (рисунок 26).



```
hello.html > html > head
1 <html>
2   <head>
3     <!-- K -->
4   </head>
5   <body>
6     <h1>Hello, world </h1>
7     <h2>Sencond line </h2>
8   </body>
9 </html>
```

Рисунок 49 - Внесение ненужного изменения

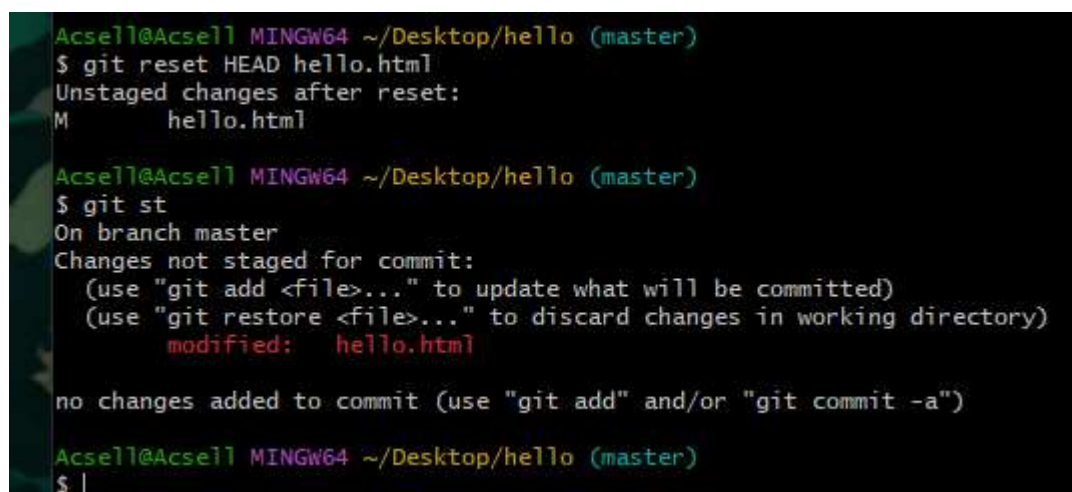


```
Acse11@Acse11 MINGW64 ~/Desktop/hello (master)
$ git add hello.html

Acse11@Acse11 MINGW64 ~/Desktop/hello (master)
$ git st
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   hello.html
```

Рисунок 50 - Индексация изменения

Для отмены индексация изменения используется команда `git reset HEAD hello.html` (рисунок 27). Команда `reset` сбрасывает буферную зону к HEAD и очищает ее от проиндексированных изменений. Но для удаления ненужного по-прежнему используется команда `git checkout` (рисунок 28).



```
Acse11@Acse11 MINGW64 ~/Desktop/hello (master)
$ git reset HEAD hello.html
Unstaged changes after reset:
M       hello.html

Acse11@Acse11 MINGW64 ~/Desktop/hello (master)
$ git st
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")

Acse11@Acse11 MINGW64 ~/Desktop/hello (master)
$
```

Рисунок 51 - Очистка буферной зоны

```
Acse11@Acse11 MINGW64 ~/Desktop/hello (master)
$ git checkout hello.html
Updated 1 path from the index

Acse11@Acse11 MINGW64 ~/Desktop/hello (master)
$ git st
On branch master
nothing to commit, working tree clean

Acse11@Acse11 MINGW64 ~/Desktop/hello (master)
$ |
```

Рисунок 52 - Удаление ненужных изменений

Отмена коммитов

Для отмены коммита можно использовать способ создания нового коммита, отменяющего изменения.

Для начала надо внести изменение, проиндексировать его и записать коммит (рисунки 29-30).

```
5 hello.html > html > body
1 <html>
2   <head>
3
4   </head>
5   <body>
6     <h1>Hello, world </h1>
7     <h2>Sencond line </h2>
8 |   <!-- -->
9   </body>
10 </html>
```

Рисунок 53 - Внесение изменения в файл

```
Acse11@Acse11 MINGW64 ~/Desktop/hello (master)
$ git add hello.html

Acse11@Acse11 MINGW64 ~/Desktop/hello (master)
$ git commit -m "bad commit"
[master 53869fc] bad commit
1 file changed, 1 insertion(+)

Acse11@Acse11 MINGW64 ~/Desktop/hello (master)
$ |
```

Рисунок 54 - Индексация и коммит

Для создания коммита, который удалит ненужные изменения, используется команда `git revert HEAD` (рисунок 31). После этого будет открыт редактор, в котором можно отредактировать коммит сообщение (рисунок 32), затем надо сохранить файл и закрыть редактор (рисунок 33).

```
Acse11@Acse11 MINGW64 ~/Desktop/hello (master)
$ git revert HEAD
hint: Waiting for your editor to close the file...
[14568:0624/211748.488:ERROR:service_worker_storage.cc(2072)] Failed to delete t
he database: Database IO error
[master f6c1abf] Revert "bad commit"
1 file changed, 1 deletion(-)

Acse11@Acse11 MINGW64 ~/Desktop/hello (master)
$
```

Рисунок 55 - Выполнение команды git revert

```
1  Revert "bad commit"
2
3  This reverts commit 53869fcfa91887aed0aaa8c3dac83162f83c1e44.
4
5  # Please enter the commit message for your changes. Lines starting
6  # with '#' will be ignored, and an empty message aborts the commit.
7  #
8  # On branch master
9  # Changes to be committed:
10 #   modified:   hello.html
11 #
12
```

Рисунок 56 - Коммит сообщение в редакторе

```
he database: Database IO error
[master f6c1abf] Revert "bad commit"
1 file changed, 1 deletion(-)

Acse11@Acse11 MINGW64 ~/Desktop/hello (master)
$
```

Рисунок 57 - Редактор закрыт

При проверке лога будут показаны все коммиты, в том числе и отмененные (рисунок 34).

```
Acse11@Acse11 MINGW64 ~/Desktop/hello (master)
$ git hist
* f6c1abf 2025-06-24 | Revert "bad commit" (HEAD -> master) [Acse11Meent]
* 53869fc 2025-06-24 | bad commit [Acse11Meent]
* bb11936 2025-06-23 | Add HTML header (tag: ver1) [Acse11Meent]
* 666f580 2025-06-23 | Add standart HTML tags (tag: ver1-beta) [Acse11Meent]
* 2c3a2cf 2025-06-23 | Added h2 tag [Acse11Meent]
* fcff9dc 2025-06-23 | Unrelated change for v [Acse11Meent]
* 77e640d 2025-06-23 | Changes for a and b [Acse11Meent]
* 369b6f6 2025-06-23 | First Commit [Acse11Meent]

Acse11@Acse11 MINGW64 ~/Desktop/hello (master)
$
```

Рисунок 58 - Все коммиты при просмотре лога

Перед удалением коммита последний из них нужно отметить тегом, чтобы не потерять его (рисунок 35).


```
Acse11@Acse11 MINGW64 ~/Desktop/hello (master)
$ git tag oooops

Acse11@Acse11 MINGW64 ~/Desktop/hello (master)
$ git hist
* f6c1abf 2025-06-24 | Revert "bad commit" (HEAD -> master, tag: oooops) [Acse11Meent]
* 53869fc 2025-06-24 | bad commit [Acse11Meent]
* bb11936 2025-06-23 | Add HTML header (tag: ver1) [Acse11Meent]
* 666f580 2025-06-23 | Add standart HTML tags (tag: ver1-beta) [Acse11Meent]
* 2c3a2cf 2025-06-23 | Added h2 tag [Acse11Meent]
* fcff9dc 2025-06-23 | Unrelated change for v [Acse11Meent]
* 77e640d 2025-06-23 | Changes for a and b [Acse11Meent]
* 369b6f6 2025-06-23 | First Commit [Acse11Meent]

Acse11@Acse11 MINGW64 ~/Desktop/hello (master)
$ |
```

Рисунок 59 - Отметка тегом коммита

Для сброса коммитов используется команда `git reset --hard ver1` (рисунок 36). Она сбрасывает ветку до версии с тегом `ver1`.

```
Acse11@Acse11 MINGW64 ~/Desktop/hello (master)
$ git reset --hard ver1
HEAD is now at bb11936 Add HTML header

Acse11@Acse11 MINGW64 ~/Desktop/hello (master)
$ git hist
* bb11936 2025-06-23 | Add HTML header (HEAD -> master, tag: ver1) [Acse11Meent]
* 666f580 2025-06-23 | Add standart HTML tags (tag: ver1-beta) [Acse11Meent]
* 2c3a2cf 2025-06-23 | Added h2 tag [Acse11Meent]
* fcff9dc 2025-06-23 | Unrelated change for v [Acse11Meent]
* 77e640d 2025-06-23 | Changes for a and b [Acse11Meent]
* 369b6f6 2025-06-23 | First Commit [Acse11Meent]

Acse11@Acse11 MINGW64 ~/Desktop/hello (master)
$ |
```

Рисунок 60 - Сброс коммита

Но при просмотре лога с помощью команды `git hist --all` отмененные коммиты по-прежнему будут показываться, так как они всё еще находятся в репозитории (рисунок 37).

```
Acse11@Acse11 MINGW64 ~/Desktop/hello (master)
$ git hist --all
* f6c1abf 2025-06-24 | Revert "bad commit" (tag: oooops) [Acse11Meent]
* 53869fc 2025-06-24 | bad commit [Acse11Meent]
* bb11936 2025-06-23 | Add HTML header (HEAD -> master, tag: ver1) [Acse11Meent]
* 666f580 2025-06-23 | Add standart HTML tags (tag: ver1-beta) [Acse11Meent]
* 2c3a2cf 2025-06-23 | Added h2 tag [Acse11Meent]
* fcff9dc 2025-06-23 | Unrelated change for v [Acse11Meent]
* 77e640d 2025-06-23 | Changes for a and b [Acse11Meent]
* 369b6f6 2025-06-23 | First Commit [Acse11Meent]

Acse11@Acse11 MINGW64 ~/Desktop/hello (master)
$ |
```

Рисунок 61 - Сброшенные коммиты находятся по-прежнему в репозитории

Удаление тега

Так как тег «oooops» больше не нужен, его и коммиты, на которые он указывает, можно удалить с помощью команды `git tag -d` (рисунок 38).


```
Acse11@Acse11 MINGW64 ~/Desktop/hello (master)
$ git tag -d oooops
Deleted tag 'oooops' (was f6c1abf)

Acse11@Acse11 MINGW64 ~/Desktop/hello (master)
$ git hist --all
* bb11936 2025-06-23 | Add HTML header (HEAD -> master, tag: ver1) [Acse11Meent]
* 666f580 2025-06-23 | Add standart HTML tags (tag: ver1-beta) [Acse11Meent]
* 2c3a2cf 2025-06-23 | Added h2 tag [Acse11Meent]
* fcff9dc 2025-06-23 | Unrelated change for v [Acse11Meent]
* 77e640d 2025-06-23 | Changes for a and b [Acse11Meent]
* 369b6f6 2025-06-23 | First Commit [Acse11Meent]

Acse11@Acse11 MINGW64 ~/Desktop/hello (master)
$ |
```

Рисунок 62 - Удаление тега

_УП_3

Работа с Git

Внесение изменений в коммиты

Для начала будет создан коммит, в который позже будут внесены изменения. На рисунках 1 и 2 происходит добавление комментария в файл hello.html и его индексация и коммит.

```
hello.html > ...
1 | <!-- Author: Danir -->
2 | <html>
3 |   <head>
4 |
5 |   </head>
6 |   <body>
7 |     <h1>Hello, world </h1>
8 |     <h2>Sencond line </h2>
9 |   </body>
10| </html>
```

Рисунок 63 - Добавление комментария в файл

```
MINGW64:/c/Users/Acsell/Desktop/hello

Acsell@Acsell MINGW64 ~/Desktop/hello (master)
$ git add hello.html

Acsell@Acsell MINGW64 ~/Desktop/hello (master)
$ git commit -m "Add an author comment"
[master 3b43afe] Add an author comment
1 file changed, 1 insertion(+)

Acsell@Acsell MINGW64 ~/Desktop/hello (master)
$ |
```

Рисунок 64 - Индексация и коммит

Далее необходимо добавить электронную почту в комментарий (рисунок 3).

```
hello.html > ...
1  <!-- Author: Danir (danir@mail.ru) -->
2  <html>
3    <head>
4
5    </head>
6    <body>
7      <h1>Hello, world </h1>
8      <h2>Sencond line </h2>
9    </body>
10 </html>
```

Рисунок 65 - Добавление электронной почты

Но для того, чтобы не создавать отдельный коммит ради электронной почты, можно изменить предыдущий так, как показано на рисунке 4.

```
Acsell@Acsell MINGW64 ~/Desktop/hello (master)
$ git add hello.html

Acsell@Acsell MINGW64 ~/Desktop/hello (master)
$ git commit --amend -m "Add an author comment"
[master f05a344] Add an author comment
Date: Tue Jun 24 21:24:15 2025 +0300
1 file changed, 1 insertion(+)

Acsell@Acsell MINGW64 ~/Desktop/hello (master)
$ |
```

Рисунок 66 - Индексация и изменение коммита

При просмотре истории можно будет заметить, что последний коммит был изменен.

```
Acse11@Acse11 MINGW64 ~/Desktop/hello (master)
$ git hist
* f05a344 2025-06-24 | Add an author comment (HEAD -> master) [Acse11Meent]
* bb11936 2025-06-23 | Add HTML header (tag: ver1) [Acse11Meent]
* 666f580 2025-06-23 | Add standart HTML tags (tag: ver1-beta) [Acse11Meent]
* 2c3a2cf 2025-06-23 | Added h2 tag [Acse11Meent]
* fcff9dc 2025-06-23 | Unrelated change for v [Acse11Meent]
* 77e640d 2025-06-23 | Changes for a and b [Acse11Meent]
* 369b6f6 2025-06-23 | First Commit [Acse11Meent]

Acse11@Acse11 MINGW64 ~/Desktop/hello (master)
$ .....
```

Рисунок 67 - Последний коммит изменен

Перемещение файлов

Для перемещения файлов в пределах репозитория используются команды, показанные на рисунке 6. После выполнения данных команды git индексирует эти изменения (удаление файла hello.html и создание файла lib/hello.html).

```
Acse11@Acse11 MINGW64 ~/Desktop/hello (master)
$ mkdir lib

Acse11@Acse11 MINGW64 ~/Desktop/hello (master)
$ git mv hello.html lib

Acse11@Acse11 MINGW64 ~/Desktop/hello (master)
$ git st
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        renamed:    hello.html -> lib/hello.html

Acse11@Acse11 MINGW64 ~/Desktop/hello (master)
$ |
```

Рисунок 68 - Перемещение файла

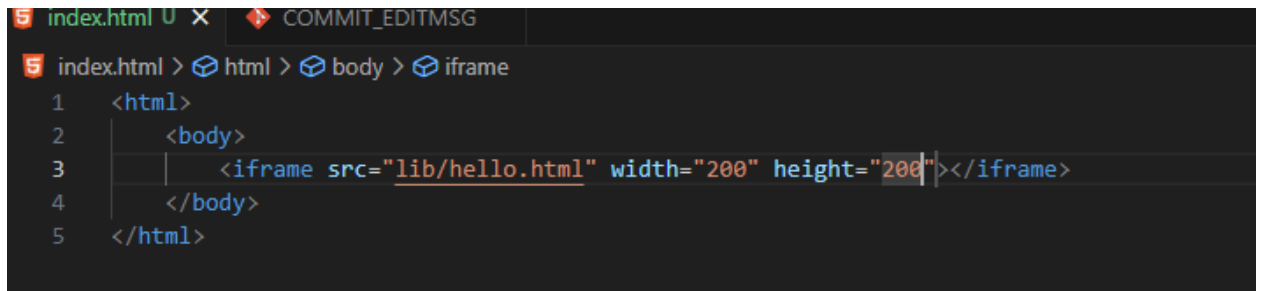
Далее надо осуществить коммит данного перемещения.

```
Acse11@Acse11 MINGW64 ~/Desktop/hello (master)
$ git commit -m "Moved hello.html to lib"
[master 7d5e215] Moved hello.html to lib
1 file changed, 0 insertions(+), 0 deletions(-)
rename hello.html => lib/hello.html (100%)
```

Рисунок 69 – Коммит перемещения

Подробнее о структуре

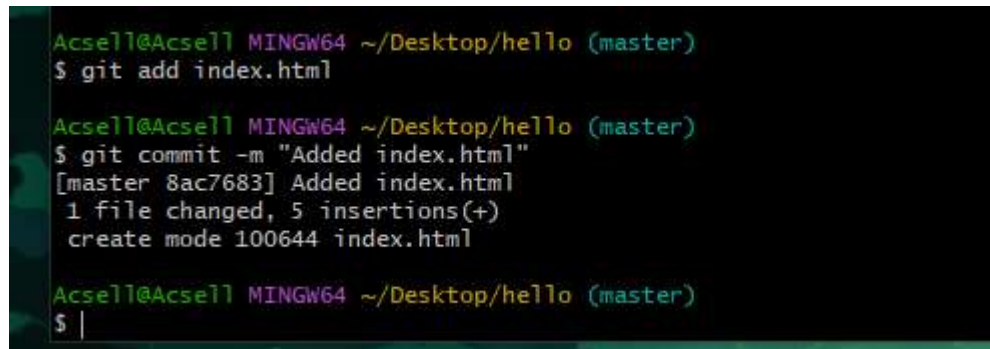
Необходимо добавить еще один файл в репозиторий. Это будет файл index.html с кодом, показанным на рисунке 8.



```
index.html > html > body > iframe
1 <html>
2   <body>
3     <iframe src="lib/hello.html" width="200" height="200"></iframe>
4   </body>
5 </html>
```

Рисунок 70 - Содержимое файла index.html

Далее нужно проиндексировать и закоммитить файл (рисунок 9).



```
Acse11@Acse11 MINGW64 ~/Desktop/hello (master)
$ git add index.html

Acse11@Acse11 MINGW64 ~/Desktop/hello (master)
$ git commit -m "Added index.html"
[master 8ac7683] Added index.html
1 file changed, 5 insertions(+)
create mode 100644 index.html

Acse11@Acse11 MINGW64 ~/Desktop/hello (master)
$
```

Рисунок 71 - Индексация и коммит

При открытии файла index.html будет виден кусок страницы hello.html (рисунок 10).

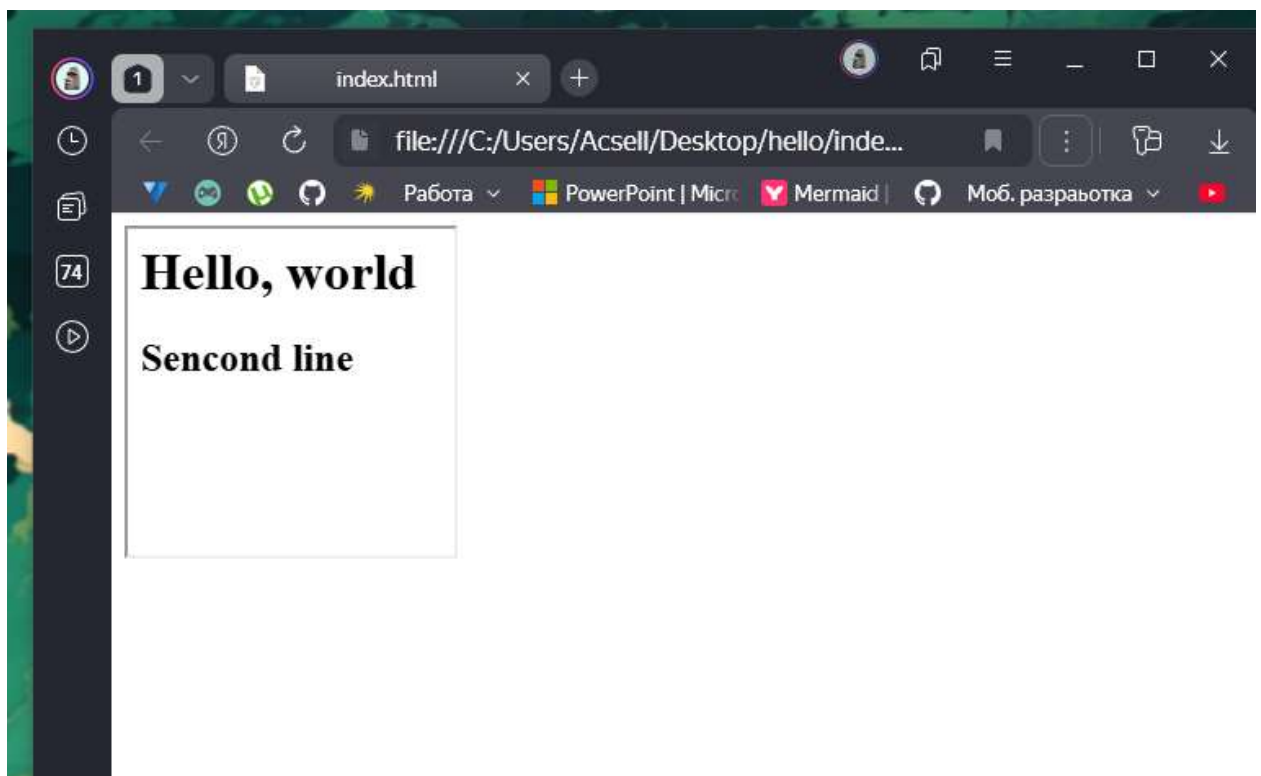


Рисунок 72 - Файл index.html, открытый в браузере

Каталог .git

Чтобы посмотреть структуру каталога .git необходимо выполнить команду, показанную на рисунке 11.

```
Acse11@Acse11 MINGW64 ~/Desktop/hello (master)
$ ls -C .git
COMMIT_EDITMSG  ORIG_HEAD  description  index  logs/  packed-refs
HEAD           config     hooks/       info/  objects/ refs/

Acse11@Acse11 MINGW64 ~/Desktop/hello (master)
$
```

Рисунок 73 - Содержание каталога .git

При аналогичном просмотре каталога objects можно будет увидеть множество каталогов с именами из 2 символов (рисунок 12). Имена каталогов являются первыми двумя буквами хэша.

```
Acse11@Acse11 MINGW64 ~/Desktop/hello (master)
$ ls -C .git/objects/
03/ 0e/ 36/ 3b/ 53/ 65/ 77/ 7f/ 8f/ a2/ bb/ d2/ d8/ e6/ ec/ f6/ fe/ pack/
05/ 2c/ 39/ 49/ 5c/ 66/ 7d/ 8a/ 98/ b3/ be/ d6/ db/ e7/ f0/ fc/ info/

Acse11@Acse11 MINGW64 ~/Desktop/hello (master)
$
```

Рисунок 74 - Содержание каталога objects

При просмотре содержимого любого из каталогов будут показаны файлы, названия которых состоят из 38 символов (рисунок 13).

```
Acse11@Acse11 MINGW64 ~/Desktop/hello (master)
$ ls -C .git/objects/8a
c7683bf1dad1220b881c2540db6c86e75d4cb1

Acse11@Acse11 MINGW64 ~/Desktop/hello (master)
$ |
```

Рисунок 75 - Просмотр каталога 8d

Далее требуется посмотреть файл конфигурации с помощью команды cat (рисунок 14).

```
Acse11@Acse11 MINGW64 ~/Desktop/hello (master)
$ cat .git/config
[core]
    repositoryformatversion = 0
    filemode = false
    bare = false
    logallrefupdates = true
    symlinks = false
    ignorecase = true

Acse11@Acse11 MINGW64 ~/Desktop/hello (master)
$
```

Рисунок 76 - Просмотр файла конфигурации

На рисунке 14 показан просмотр файлов в подкаталоге tags и веток в каталоге heads.

```
Acse11@Acse11 MINGW64 ~/Desktop/hello (master)
$ ls .git/refs
heads/  tags/

Acse11@Acse11 MINGW64 ~/Desktop/hello (master)
$ ls .git/refs/heads/
master

Acse11@Acse11 MINGW64 ~/Desktop/hello (master)
$ ls .git/refs/tags/
ver1  ver1-beta

Acse11@Acse11 MINGW64 ~/Desktop/hello (master)
$ cat .git/refs/tags/ver1
bb11936fc0a1b33a0037794ca421dc684e73f6d8

Acse11@Acse11 MINGW64 ~/Desktop/hello (master)
$ |
```

Рисунок 77 - Просмотр файлов и веток

Файл HEAD содержит ссылку на текущую ветку (рисунок 16).

```
Acse11@Acse11 MINGW64 ~/Desktop/hello (master)
$ cat .git/HEAD
ref: refs/heads/master

Acse11@Acse11 MINGW64 ~/Desktop/hello (master)
$ |
```

Рисунок 78 - Содержимое файла HEAD

Работа с объектами git

Для начала необходимо просмотреть последний коммит (рисунок 17).

```
Acse11@Acse11 MINGW64 ~/Desktop/hello (master)
$ git hist --max-count=1
* 8ac7683 2025-06-24 | Added index.html (HEAD -> master) [Acse11Meent]

Acse11@Acse11 MINGW64 ~/Desktop/hello (master)
$ |
```

Рисунок 79 - Последний коммит

Далее надо использовать хэш последнего коммита, используя команды `cat-file -p` и `cat-file -t` (рисунок 18) для просмотра объекта коммита. Также вместо длинных команд можно использовать сокращенные `type` и `dump`, если данные команды были заданы как алиасы.


```

Acsell@Acsell MINGW64 ~/Desktop/hello (master)
$ git cat-file -t 8ac7683
commit

Acsell@Acsell MINGW64 ~/Desktop/hello (master)
$ git cat-file -p 8ac7683
tree e6b873773fd74eb1e45f4234aedcfacfe4ed0e5f
parent 7d5e215088f78bacfd325358e12d84094cbb3b1d
author AcsellMeent <dshalkhykov@mail.ru> 1750789737 +0300
committer AcsellMeent <dshalkhykov@mail.ru> 1750789737 +0300

Added index.html

Acsell@Acsell MINGW64 ~/Desktop/hello (master)

```

Рисунок 80 - Просмотр объекта коммита

```

Acsell@Acsell MINGW64 ~/Desktop/hello (master)
$ git type 8ac7683
commit

Acsell@Acsell MINGW64 ~/Desktop/hello (master)
$ git dump 8ac7683
tree e6b873773fd74eb1e45f4234aedcfacfe4ed0e5f
parent 7d5e215088f78bacfd325358e12d84094cbb3b1d
author AcsellMeent <dshalkhykov@mail.ru> 1750789737 +0300
committer AcsellMeent <dshalkhykov@mail.ru> 1750789737 +0300

Added index.html

Acsell@Acsell MINGW64 ~/Desktop/hello (master)

```

Рисунок 81 - Использование алиасов

Для просмотра дерева каталогов необходимо использовать его хэш (рисунок 20).

```

Acsell@Acsell MINGW64 ~/Desktop/hello (master)
$ git cat-file -p e6b8737
100644 blob e69de29bb2d1d6434b8b29ae775ad8c2e48c5391    a.html
100644 blob e69de29bb2d1d6434b8b29ae775ad8c2e48c5391    b.html
100644 blob 03ee26631ef36c173c9c1a9f7e51dd419f72482d    index.html
040000 tree ec096c360d7cb8bed36f708421acf333a470f849    lib
100644 blob e69de29bb2d1d6434b8b29ae775ad8c2e48c5391    v.html

Acsell@Acsell MINGW64 ~/Desktop/hello (master)
$

```

Рисунок 82 - Просмотр дерева каталогов

Затем нужно просмотреть каталог lib (рисунок 21).

```

Acsell@Acsell MINGW64 ~/Desktop/hello (master)
$ git cat-file -p ec096c
100644 blob 3bc43652c2aa064f7a3b0b6276df38961e07d900    hello.html

Acsell@Acsell MINGW64 ~/Desktop/hello (master)
$

```

Рисунок 83 - Просмотр каталога lib

И затем требуется вывести содержимое файла hello.html (рисунок 22).

```
Acse11@Acse11 MINGW64 ~/Desktop/hello (master)
$ git cat-file -p 3bc436
<!-- Author: Danir (danir@mail.ru) -->
<html>
  <head>

  </head>
  <body>
    <h1>Hello, world </h1>
    <h2>Sencond line </h2>
  </body>
</html>
Acse11@Acse11 MINGW64 ~/Desktop/hello (master)
$
```

Рисунок 84 - Вывод содержимого файла hello.html

Аналогичным образом можно просмотреть содержимое файла, каким оно было в самом первом коммите, как показано на рисунке 23. Для этого требуется использовать лишь нужные хэши.

```
Acsell@Acsell MINGW64 ~/Desktop/hello (master)
$ git hist
* 8ac7683 2025-06-24 | Added index.html (HEAD -> master) [AcsellMeent]
* 7d5e215 2025-06-24 | Moved hello.html to lib [AcsellMeent]
* f05a344 2025-06-24 | Add an author comment [AcsellMeent]
* bb11936 2025-06-23 | Add HTML header (tag: ver1) [AcsellMeent]
* 666f580 2025-06-23 | Add standart HTML tags (tag: ver1-beta) [AcsellMeent]
* 2c3a2cf 2025-06-23 | Added h2 tag [AcsellMeent]
* fcff9dc 2025-06-23 | Unrelated change for v [AcsellMeent]
* 77e640d 2025-06-23 | Changes for a and b [AcsellMeent]
* 369b6f6 2025-06-23 | First Commit [AcsellMeent]

Acsell@Acsell MINGW64 ~/Desktop/hello (master)
$ git cat-file -p 369b6f6
tree d8a800db068656c3aafc962f7a438c987b49dd60
author AcsellMeent <dshalkhykov@mail.ru> 1750709873 +0300
committer AcsellMeent <dshalkhykov@mail.ru> 1750709873 +0300

First Commit

Acsell@Acsell MINGW64 ~/Desktop/hello (master)
$ git cat-file -p d8a800
100644 blob dbe9dba55ea8fd4d5be3868b015e044be0848ec5      hello.html

Acsell@Acsell MINGW64 ~/Desktop/hello (master)
$ git cat-file -p dbe9db
Hello, world
Acsell@Acsell MINGW64 ~/Desktop/hello (master)
$
```

Рисунок 85 - Просмотр содержимого файла при первом коммите

_УП_4

Работа с Git

Создание ветки

Для начала необходимо создать ветку style с помощью команды `git checkout -b style` (рисунок 1).

```

Acse11@Acse11 MINGW64 ~/Desktop/hello (master)
$ git checkout -b style
Switched to a new branch 'style'

Acse11@Acse11 MINGW64 ~/Desktop/hello (style)
$ git ststus
git: 'ststus' is not a git command. See 'git --help'.

The most similar command is
    status

Acse11@Acse11 MINGW64 ~/Desktop/hello (style)
$ git checkout -b style
fatal: a branch named 'style' already exists

Acse11@Acse11 MINGW64 ~/Desktop/hello (style)
$ git st
On branch style
nothing to commit, working tree clean

Acse11@Acse11 MINGW64 ~/Desktop/hello (style)
$

```

Рисунок 86 - Создание новой ветки style

Затем нужно создать файл стилей (рисунок 2) и внести в него код, показанный на рисунке

3.

```

Acse11@Acse11 MINGW64 ~/Desktop/hello (style)
$ touch lib/style.css

Acse11@Acse11 MINGW64 ~/Desktop/hello (style)
$

```

Рисунок 87 - Создание файла стилей

```

lib > css style.css > h1
1  h1 {
2    color: red;
3  }

```

Рисунок 88 - Код style.css

После этого надо произвести индексацию и коммит (рисунок 4).

```

Acse11@Acse11 MINGW64 ~/Desktop/hello (style)
$ git add lib/style.css

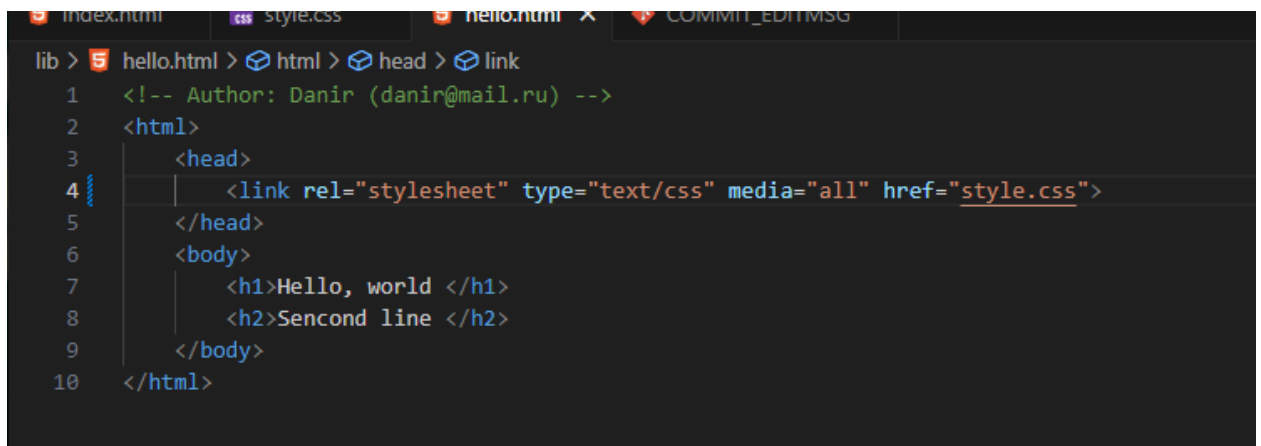
Acse11@Acse11 MINGW64 ~/Desktop/hello (style)
$ git commit -m "Added css stylesheet"
[style b8073b3] Added css stylesheet
1 file changed, 3 insertions(+)
create mode 100644 lib/style.css

Acse11@Acse11 MINGW64 ~/Desktop/hello (style)
$

```

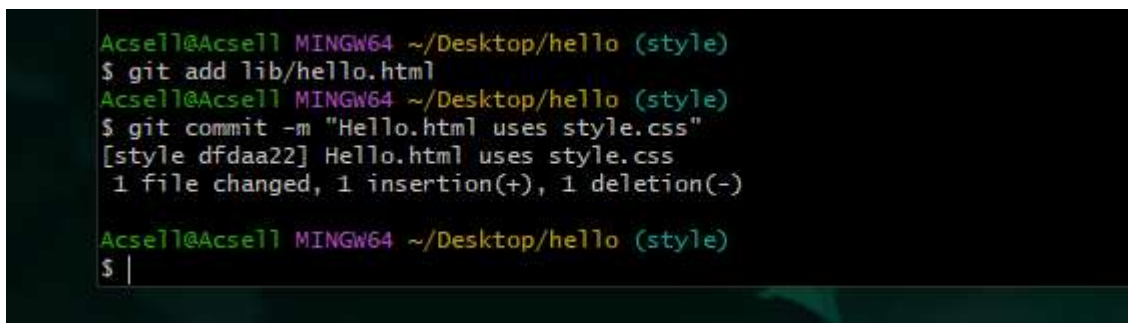
Рисунок 89 - Индексация и коммит нового файла

Далее требуется изменить основную страницу hello.html и закоммитить изменения (рисунки 5-6).



```
lib > hello.html > html > head > link
1  <!-- Author: Danir (danir@mail.ru) -->
2  <html>
3    <head>
4      <link rel="stylesheet" type="text/css" media="all" href="style.css">
5    </head>
6    <body>
7      <h1>Hello, world </h1>
8      <h2>Sencond line </h2>
9    </body>
10 </html>
```

Рисунок 90 - Изменения в файле hello.html



```
Acse11@Acse11 MINGW64 ~/Desktop/hello (style)
$ git add lib/hello.html
Acse11@Acse11 MINGW64 ~/Desktop/hello (style)
$ git commit -m "Hello.html uses style.css"
[style dfdaa22] Hello.html uses style.css
1 file changed, 1 insertion(+), 1 deletion(-)

Acse11@Acse11 MINGW64 ~/Desktop/hello (style)
$ |
```

Рисунок 91 - Индексация и коммит

Далее аналогичные действия нужно осуществить с файлом index.html, как это показано на рисунках 7-8.



```
index.html > html > head
1  <html>
2    <head>
3      <link rel="stylesheet" type="text/css" media="all" href="style.css">
4    </head>
5    <body>
6      <iframe src="lib/hello.html" width="200" height="200"></iframe>
7    </body>
8  </html>
```

Рисунок 92 - Изменения в файле index.html

```
Acse11@Acse11 MINGW64 ~/Desktop/hello (style)
$ git add index.html

Acse11@Acse11 MINGW64 ~/Desktop/hello (style)
$ git commit -m "Update index.html"
[style fec03e8] Update index.html
1 file changed, 3 insertions(+)

Acse11@Acse11 MINGW64 ~/Desktop/hello (style)
$ |
```

Рисунок 93 - Индексация и коммит

После выполнения предыдущих действий была создана новая ветка style с 3 коммитами.

Навигация по веткам

При просмотре истории, как на рисунке 9, можно увидеть, что теперь в проекте 2 ветки.

```
Acse11@Acse11 MINGW64 ~/Desktop/hello (style)
$ git hist --all
* fec03e8 2025-06-24 | Update index.html (HEAD -> style) [Acse11Meent]
* dfdaa22 2025-06-24 | Hello.html uses style.css [Acse11Meent]
* b8073b3 2025-06-24 | Added css stylesheet [Acse11Meent]
* 8ac7683 2025-06-24 | Added index.html (master) [Acse11Meent]
* 7d5e215 2025-06-24 | Moved hello.html to lib [Acse11Meent]
* f05a344 2025-06-24 | Add an author comment [Acse11Meent]
* bb11936 2025-06-23 | Add HTML header (tag: ver1) [Acse11Meent]
* 666f580 2025-06-23 | Add standart HTML tags (tag: ver1-beta) [Acse11Meent]
* 2c3a2cf 2025-06-23 | Added h2 tag [Acse11Meent]
* fcff9dc 2025-06-23 | Unrelated change for v [Acse11Meent]
* 77e640d 2025-06-23 | Changes for a and b [Acse11Meent]
* 369b6f6 2025-06-23 | First Commit [Acse11Meent]

Acse11@Acse11 MINGW64 ~/Desktop/hello (style)
$
```

Рисунок 94 - Просмотр истории

Для переключения на ветку master используется команда `git checkout master` (рисунок 10). После переключения на нужную ветку при выводе файла `hello.html` можно увидеть, что изменения отсутствуют (по причине того, что они закоммичены в другой ветке).

```
Acse11@Acse11 MINGW64 ~/Desktop/hello (style)
$ git checkout master
Switched to branch 'master'

Acse11@Acse11 MINGW64 ~/Desktop/hello (master)
$ cat lib/hello.html
<!-- Author: Danir (danir@mail.ru) -->
<html>
  <head>

  </head>
  <body>
    <h1>Hello, world </h1>
    <h2>Sencond line </h2>
  </body>
</html>
Acse11@Acse11 MINGW64 ~/Desktop/hello (master)
$
```

Рисунок 95 - Переключение на ветку master

11). При переключении на ветку style файл hello.html будет иметь другое содержание (рисунок

```
Acse11@Acse11 MINGW64 ~/Desktop/hello (master)
$ git checkout style
Switched to branch 'style'

Acse11@Acse11 MINGW64 ~/Desktop/hello (style)
$ cat lib/hello.html
<!-- Author: Danir (danir@mail.ru) -->
<html>
  <head>
    <link rel="stylesheet" type="text/css" media="all" href="s
  </head>
  <body>
    <h1>Hello, world </h1>
    <h2>Sencond line </h2>
  </body>
</html>
Acse11@Acse11 MINGW64 ~/Desktop/hello (style)
$ |
```

Рисунок 96 - Переключение на ветку style

Изменения в ветке master

Необходимо переключиться на ветку master (рисунок 12) и добавить файл README (рисунки 13-14).

```
Acse11@Acse11 MINGW64 ~/Desktop/hello (style)
$ git checkout master
Switched to branch 'master'

Acse11@Acse11 MINGW64 ~/Desktop/hello (master)
```

Рисунок 97 - Переключение на ветку master

```
≡ README ×
C: > Users > elena > Documents > hello > ≡ README
1 This is the Hello World example from the Git tutorial|
```

Рисунок 98 - Содержимое файла README

```
Acse11@Acse11 MINGW64 ~/Desktop/hello (master)
$ git add README

Acse11@Acse11 MINGW64 ~/Desktop/hello (master)
$ git commit -m "Added README"
[master 757462c] Added README
1 file changed, 1 insertion(+)
create mode 100644 README

Acse11@Acse11 MINGW64 ~/Desktop/hello (master)
```

Рисунок 99 - Индексация и коммит

Просмотр отличающихся веток

На рисунке 15 можно увидеть дерево коммитов.

```
Acse11@Acse11 MINGW64 ~/Desktop/hello (master)
$ git hist --all
* 757462c 2025-06-24 | Added README (HEAD -> master) [Acse11Meent]
| * fec03e8 2025-06-24 | Update index.html (style) [Acse11Meent]
| * dfdaa22 2025-06-24 | Hello.html uses style.css [Acse11Meent]
| * b8073b3 2025-06-24 | Added css stylesheet [Acse11Meent]
|/
* 8ac7683 2025-06-24 | Added index.html [Acse11Meent]
* 7d5e215 2025-06-24 | Moved hello.html to lib [Acse11Meent]
* f05a344 2025-06-24 | Add an author comment [Acse11Meent]
* bb11936 2025-06-23 | Add HTML header (tag: ver1) [Acse11Meent]
* 666f580 2025-06-23 | Add standart HTML tags (tag: ver1-beta) [Acse11Meent]
* 2c3a2cf 2025-06-23 | Added h2 tag [Acse11Meent]
* fcff9dc 2025-06-23 | Unrelated change for v [Acse11Meent]
* 77e640d 2025-06-23 | Changes for a and b [Acse11Meent]
* 369b6f6 2025-06-23 | First Commit [Acse11Meent]
Acse11@Acse11 MINGW64 ~/Desktop/hello (master)
```

Рисунок 100 - Дерево коммитов

Слияние

Слияние переносит изменения из двух веток в одну. Для слияния нужно перейти на ветку style и с помощью команды git merge master совместить ветки (рисунки 16-18).

```
Acse11@Acse11 MINGW64 ~/Desktop/hello (master)
$ git checkout style
Switched to branch 'style'

Acse11@Acse11 MINGW64 ~/Desktop/hello (style)
$ git merge master
Merge made by the 'ort' strategy.
```

Рисунок 101 - Переключение на ветку style

```
Acse11@Acse11 MINGW64 ~/Desktop/hello (style)
$ git merge master
Merge made by the 'ort' strategy.
 README | 1 +
 1 file changed, 1 insertion(+)
 create mode 100644 README

Acse11@Acse11 MINGW64 ~/Desktop/hello (style)
$
```

Рисунок 102 - Слияние с веткой master

```

Acse11@Acse11 MINGW64 ~/Desktop/hello (style)
$ git hist --all
* aec9849 2025-06-24 | Merge branch 'master' into style (HEAD -> style) [Acse11Meent]
| \
| * 757462c 2025-06-24 | Added README (master) [Acse11Meent]
* | fec03e8 2025-06-24 | Update index.html [Acse11Meent]
* | dfdaa22 2025-06-24 | Hello.html uses style.css [Acse11Meent]
* | b8073b3 2025-06-24 | Added css stylesheet [Acse11Meent]
| /
* 8ac7683 2025-06-24 | Added index.html [Acse11Meent]
* 7d5e215 2025-06-24 | Moved hello.html to lib [Acse11Meent]
* f05a344 2025-06-24 | Add an author comment [Acse11Meent]
* bb11936 2025-06-23 | Add HTML header (tag: ver1) [Acse11Meent]
* 666f580 2025-06-23 | Add standart HTML tags (tag: ver1-beta) [Acse11Meent]
* 2c3a2cf 2025-06-23 | Added h2 tag [Acse11Meent]
* fcff9dc 2025-06-23 | Unrelated change for v [Acse11Meent]
* 77e640d 2025-06-23 | Changes for a and b [Acse11Meent]
* 369b6f6 2025-06-23 | First Commit [Acse11Meent]

Acse11@Acse11 MINGW64 ~/Desktop/hello (style)
$ |

```

Рисунок 103 - Просмотр истории

Создание конфликта

Для того, чтобы создать конфликт необходимо перейти в ветку master и внести изменения в файл hello.html (рисунки 19-21).

```

Acse11@Acse11 MINGW64 ~/Desktop/hello (style)
$ git checkout master
Switched to branch 'master'

Acse11@Acse11 MINGW64 ~/Desktop/hello (master)
$ |

```

Рисунок 104 - Переход в ветку master

```

lib > 5 hello.html > html > body > h1
1  <!-- Author: Danir (danir@mail.ru) -->
2  <html>
3    <head>
4      <!-- no style -->
5    </head>
6    <body>
7      <h1>Hello, world. Life is great </h1>
8      <h2>Sencond line </h2>
9    </body>
10 </html>

```

Рисунок 105 - Внесение изменений

```

Acse11@Acse11 MINGW64 ~/Desktop/hello (master)
$ git add lib/hello.html

Acse11@Acse11 MINGW64 ~/Desktop/hello (master)
$ git commit -m 'Life is great'
[master 254c5e0] Life is great
1 file changed, 2 insertions(+), 2 deletions(-)

```

Рисунок 106 - Индексация и коммит

После выполнения предыдущих действий при просмотре веток можно будет увидеть конфликт в виде, как на рисунке 22.

```

Acse11@Acse11 MINGW64 ~/Desktop/hello (master)
$ git hist --all
* 254c5e0 2025-06-24 | Life is great (HEAD -> master) [Acse11Meent]
| * aec9849 2025-06-24 | Merge branch 'master' into style (style) [Acse11Meent]
| | \
| | /
| | /
| | /
* | 757462c 2025-06-24 | Added README [Acse11Meent]
| * fec03e8 2025-06-24 | Update index.html [Acse11Meent]
| * dfdaa22 2025-06-24 | Hello.html uses style.css [Acse11Meent]
| * b8073b3 2025-06-24 | Added css stylesheet [Acse11Meent]
| /
* 8ac7683 2025-06-24 | Added index.html [Acse11Meent]
* 7d5e215 2025-06-24 | Moved hello.html to lib [Acse11Meent]
* f05a344 2025-06-24 | Add an author comment [Acse11Meent]
* bb11936 2025-06-23 | Add HTML header (tag: ver1) [Acse11Meent]
* 666f580 2025-06-23 | Add standart HTML tags (tag: ver1-beta) [Acse11Meent]
* 2c3a2cf 2025-06-23 | Added h2 tag [Acse11Meent]
* fcff9dc 2025-06-23 | Unrelated change for v [Acse11Meent]
* 77e640d 2025-06-23 | Changes for a and b [Acse11Meent]
* 369b6f6 2025-06-23 | First Commit [Acse11Meent]

```

Рисунок 107 - Конфликт изменений

Разрешение конфликтов

При попытке объединить ветку style с master будет показана ошибка из-за конфликта, как показано на рисунке 23.

```

Acse11@Acse11 MINGW64 ~/Desktop/hello (master)
$ git checkout style
Switched to branch 'style'

Acse11@Acse11 MINGW64 ~/Desktop/hello (style)
$ 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.

```

Рисунок 108 - Ошибка при слиянии

При открытии файла hello.html конфликт будет показан (рисунок 24).

```
lib > hello.html > html > head > ?
1 <!-- Author: Danir (danir@mail.ru) -->
2 <html>
3   <head>
4     <<<<<< HEAD (Current Change)
5     <link rel="stylesheet" type="text/css" media="all" href="style.css">
6     =====
7     <!-- no style -->
8     >>>>>> master (Incoming Change)
9   </head>
10  <body>
11    <h1>Hello, world. Life is great </h1>
12    <h2>Sencond line </h2>
13  </body>
14 </html>
```

Рисунок 109 - Просмотр файла hello.html при наличии конфликта

Чтобы решить конфликт, нужно внести изменения вручную (рисунок 25).

```
lib > hello.html > html > head
1 <!-- Author: Danir (danir@mail.ru) -->
2 <html>
3   <head>
4     <link rel="stylesheet" type="text/css" media="all" href="style.css">
5   </head>
6   <body>
7     <h1>Hello, world. Life is great </h1>
8     <h2>Sencond line </h2>
9   </body>
10 </html>
```

Рисунок 110 - Решение конфликта вручную

Затем следует произвести индексацию и коммит (рисунок 26).

```
Acse11@Acse11 MINGW64 ~/Desktop/hello (style|MERGING)
$ git add lib/hello.html

Acse11@Acse11 MINGW64 ~/Desktop/hello (style|MERGING)
$ git commit -m "Merged master fixed conflict."
[style f60d4d8] Merged master fixed conflict.
```

Рисунок 111 - Индексация и коммит

_УП_5

Работа с Git

Сброс ветки style

Для сброса ветки необходимо применить команду `reset --hard` до требуемой точки (рисунки 1-2).

```
Acse11@Acse11 MINGW64 ~/Desktop/hello (style)
$ git hist
* f60d4d8 2025-06-24 | Merged master fixed conflict. (HEAD -> style) [Acse11Meent]
|
| * 254c5e0 2025-06-24 | Life is great (master) [Acse11Meent]
| * | aec9849 2025-06-24 | Merge branch 'master' into style [Acse11Meent]
| |
| | * 757462c 2025-06-24 | Added README [Acse11Meent]
| * | fec03e8 2025-06-24 | Update index.html [Acse11Meent]
| * | dfdaa22 2025-06-24 | Hello.html uses style.css [Acse11Meent]
| * | b8073b3 2025-06-24 | Added css stylesheet [Acse11Meent]
| |
| * 8ac7683 2025-06-24 | Added index.html [Acse11Meent]
| * 7d5e215 2025-06-24 | Moved hello.html to lib [Acse11Meent]
| * f05a344 2025-06-24 | Add an author comment [Acse11Meent]
| * bb11936 2025-06-23 | Add HTML header (tag: ver1) [Acse11Meent]
| * 666f580 2025-06-23 | Add standart HTML tags (tag: ver1-beta) [Acse11Meent]
| * 2c3a2cf 2025-06-23 | Added h2 tag [Acse11Meent]
| * fcff9dc 2025-06-23 | Unrelated change for v [Acse11Meent]
| * 77e640d 2025-06-23 | Changes for a and b [Acse11Meent]
| * 369b6f6 2025-06-23 | First Commit [Acse11Meent]
|
Acse11@Acse11 MINGW64 ~/Desktop/hello (style)
$ |
```

Рисунок 112 - Просмотр истории

```
Acse11@Acse11 MINGW64 ~/Desktop/hello (style)
$ git reset --hard 8ac7683
HEAD is now at 8ac7683 Added index.html

Acse11@Acse11 MINGW64 ~/Desktop/hello (style)
$ git hist --all
* 254c5e0 2025-06-24 | Life is great (master) [Acse11Meent]
* 757462c 2025-06-24 | Added README [Acse11Meent]
* 8ac7683 2025-06-24 | Added index.html (HEAD -> style) [Acse11Meent]
* 7d5e215 2025-06-24 | Moved hello.html to lib [Acse11Meent]
* f05a344 2025-06-24 | Add an author comment [Acse11Meent]
* bb11936 2025-06-23 | Add HTML header (tag: ver1) [Acse11Meent]
* 666f580 2025-06-23 | Add standart HTML tags (tag: ver1-beta) [Acse11Meent]
* 2c3a2cf 2025-06-23 | Added h2 tag [Acse11Meent]
* fcff9dc 2025-06-23 | Unrelated change for v [Acse11Meent]
* 77e640d 2025-06-23 | Changes for a and b [Acse11Meent]
* 369b6f6 2025-06-23 | First Commit [Acse11Meent]
```

Рисунок 113 - Сброс ветки style

Сброс ветки master

Аналогичные действия нужно произвести и для ветки master (рисунки 3-4).

```
Acse11@Acse11 MINGW64 ~/Desktop/hello (style)
$ git checkout master
Switched to branch 'master'

Acse11@Acse11 MINGW64 ~/Desktop/hello (master)
$ git hist
* 254c5e0 2025-06-24 | Life is great (HEAD -> master) [Acse11Meent]
* 757462c 2025-06-24 | Added README [Acse11Meent]
* 8ac7683 2025-06-24 | Added index.html (style) [Acse11Meent]
* 7d5e215 2025-06-24 | Moved hello.html to lib [Acse11Meent]
* f05a344 2025-06-24 | Add an author comment [Acse11Meent]
* bb11936 2025-06-23 | Add HTML header (tag: ver1) [Acse11Meent]
* 666f580 2025-06-23 | Add standart HTML tags (tag: ver1-beta) [Acse11Meent]
* 2c3a2cf 2025-06-23 | Added h2 tag [Acse11Meent]
* fcff9dc 2025-06-23 | Unrelated change for v [Acse11Meent]
* 77e640d 2025-06-23 | Changes for a and b [Acse11Meent]
* 369b6f6 2025-06-23 | First Commit [Acse11Meent]
```

Рисунок 114 - Переключение на master и просмотр истории

```
Acse11@Acse11 MINGW64 ~/Desktop/hello (master)
$ git reset --hard 254c5e0
HEAD is now at 254c5e0 Life is great

Acse11@Acse11 MINGW64 ~/Desktop/hello (master)
$ git hist --all
* 254c5e0 2025-06-24 | Life is great (HEAD -> master) [Acse11Meent]
* 757462c 2025-06-24 | Added README [Acse11Meent]
* 8ac7683 2025-06-24 | Added index.html (style) [Acse11Meent]
* 7d5e215 2025-06-24 | Moved hello.html to lib [Acse11Meent]
* f05a344 2025-06-24 | Add an author comment [Acse11Meent]
* bb11936 2025-06-23 | Add HTML header (tag: ver1) [Acse11Meent]
* 666f580 2025-06-23 | Add standart HTML tags (tag: ver1-beta) [Acse11Meent]
* 2c3a2cf 2025-06-23 | Added h2 tag [Acse11Meent]
* fcff9dc 2025-06-23 | Unrelated change for v [Acse11Meent]
* 77e640d 2025-06-23 | Changes for a and b [Acse11Meent]
* 369b6f6 2025-06-23 | First Commit [Acse11Meent]
```

Рисунок 115 - Сброс ветки master

Перебазирование

Команду rebase можно использовать вместо команды merge (рисунок 5).

```
Acse11@Acse11 MINGW64 ~/Desktop/hello (master)
$ git checkout style
Switched to branch 'style'

Acse11@Acse11 MINGW64 ~/Desktop/hello (style)
$ git rebase master
Successfully rebased and updated refs/heads/style.

Acse11@Acse11 MINGW64 ~/Desktop/hello (style)
$ git hist
* 254c5e0 2025-06-24 | Life is great (HEAD -> style, master) [Acse11Meent]
* 757462c 2025-06-24 | Added README [Acse11Meent]
* 8ac7683 2025-06-24 | Added index.html [Acse11Meent]
* 7d5e215 2025-06-24 | Moved hello.html to lib [Acse11Meent]
* f05a344 2025-06-24 | Add an author comment [Acse11Meent]
* bb11936 2025-06-23 | Add HTML header (tag: ver1) [Acse11Meent]
* 666f580 2025-06-23 | Add standart HTML tags (tag: ver1-beta) [Acse11Meent]
* 2c3a2cf 2025-06-23 | Added h2 tag [Acse11Meent]
* fcff9dc 2025-06-23 | Unrelated change for v [Acse11Meent]
* 77e640d 2025-06-23 | Changes for a and b [Acse11Meent]
* 369b6f6 2025-06-23 | First Commit [Acse11Meent]
```

Рисунок 116 - Перебазирование веток

Слияние в ветку master

Далее требуется произвести слияние веток с помощью merge (рисунки 6-7).

```
Acse11@Acse11 MINGW64 ~/Desktop/hello (style)
$ git checkout master
Switched to branch 'master'

Acse11@Acse11 MINGW64 ~/Desktop/hello (master)
$ git merge style
Already up to date.

Acse11@Acse11 MINGW64 ~/Desktop/hello (master)
$ |
```

Рисунок 117 - Слияние веток

```
Acse11@Acse11 MINGW64 ~/Desktop/hello (master)
$ git hist
* 254c5e0 2025-06-24 | Life is great (HEAD -> master, style) [Acse11Meent]
* 757462c 2025-06-24 | Added README [Acse11Meent]
* 8ac7683 2025-06-24 | Added index.html [Acse11Meent]
* 7d5e215 2025-06-24 | Moved hello.html to lib [Acse11Meent]
* f05a344 2025-06-24 | Add an author comment [Acse11Meent]
* bb11936 2025-06-23 | Add HTML header (tag: ver1) [Acse11Meent]
* 666f580 2025-06-23 | Add standart HTML tags (tag: ver1-beta) [Acse11Meent]
* 2c3a2cf 2025-06-23 | Added h2 tag [Acse11Meent]
* fcff9dc 2025-06-23 | Unrelated change for v [Acse11Meent]
* 77e640d 2025-06-23 | Changes for a and b [Acse11Meent]
* 369b6f6 2025-06-23 | First Commit [Acse11Meent]

Acse11@Acse11 MINGW64 ~/Desktop/hello (master)
$
```

Рисунок 118 - Просмотр истории

Клонирование репозитория

Далее требуется научиться делать копии репозитория. Для этого необходимо перейти в рабочий каталог и затем использовать команду `git clone`. Все данные действия показаны на рисунке 8.

```
Acse11@Acse11 MINGW64 ~
$ cd

Acse11@Acse11 MINGW64 ~
$ pwd
/c/Users/Acse11

Acse11@Acse11 MINGW64 ~
$ cd Desktop/

Acse11@Acse11 MINGW64 ~/Desktop
$ git clone hello cloned_hello
Cloning into 'cloned_hello'...
done.

Acse11@Acse11 MINGW64 ~/Desktop
$ ls
Prac/                  cloned_hello/  hello/  'ДРД 30 июля.txt'
'...'                 desktop.ini    qwe.pptx 'Новый текстовый документ.txt'

Acse11@Acse11 MINGW64 ~/Desktop
$
```

Рисунок 119 - Переход в рабочий каталог и его клонирование

Просмотр клонированного репозитория

После этого можно посмотреть клонированный репозиторий (рисунки 9-10).

```
Acse11@Acse11 MINGW64 ~/Desktop
$ cd cloned_hello/

Acse11@Acse11 MINGW64 ~/Desktop/cloned_hello (master)
$ ls
README  a.html  b.html  index.html  lib/  v.html

Acse11@Acse11 MINGW64 ~/Desktop/cloned_hello (master)
$
```

Рисунок 120 - Просмотр содержимого клонированного репозитория

```

Acse11@Acse11 MINGW64 ~/Desktop/cloned_hello (master)
$ git hist --all
* 254c5e0 2025-06-24 | Life is great (HEAD -> master, origin/style, origin/master, origin/HEAD) [Acse11Meent]
* 757462c 2025-06-24 | Added README [Acse11Meent]
* 8ac7683 2025-06-24 | Added index.html [Acse11Meent]
* 7d5e215 2025-06-24 | Moved hello.html to lib [Acse11Meent]
* f05a344 2025-06-24 | Add an author comment [Acse11Meent]
* bb11936 2025-06-23 | Add HTML header (tag: ver1) [Acse11Meent]
* 666f580 2025-06-23 | Add standart HTML tags (tag: ver1-beta) [Acse11Meent]
* 2c3a2cf 2025-06-23 | Added h2 tag [Acse11Meent]
* fcff9dc 2025-06-23 | Unrelated change for v [Acse11Meent]
* 77e640d 2025-06-23 | Changes for a and b [Acse11Meent]
* 369b6f6 2025-06-23 | First Commit [Acse11Meent]

Acse11@Acse11 MINGW64 ~/Desktop/cloned_hello (master)
$

```

Рисунок 121 - Просмотр логов клонированного каталога

Origin

Origin – имя по умолчанию. Просмотр данных о нем возможен с помощью команд, показанных на рисунке 11.

```

Acse11@Acse11 MINGW64 ~/Desktop/cloned_hello (master)
$ git remote
origin

Acse11@Acse11 MINGW64 ~/Desktop/cloned_hello (master)
$ git remote show origin
* remote origin
  Fetch URL: C:/Users/Acse11/Desktop/hello
  Push URL: C:/Users/Acse11/Desktop/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)

Acse11@Acse11 MINGW64 ~/Desktop/cloned_hello (master)

```

Рисунок 122 - Просмотр данных об origin

Удаленные ветки

Для просмотра удаленных веток используется команда `git branch -a` (рисунок 12).


```
Acse11@Acse11 MINGW64 ~/Desktop/cloned_hello (master)
$ git branch
* master

Acse11@Acse11 MINGW64 ~/Desktop/cloned_hello (master)
$ git branch -a
* master
remotes/origin/HEAD -> origin/master
remotes/origin/master
remotes/origin/style

Acse11@Acse11 MINGW64 ~/Desktop/cloned_hello (master)
$ |
```

Рисунок 123 - Просмотр удаленных веток

_УП_6

Работа с Git

Сначала необходимо внести изменения в оригинальный репозиторий. Для этого нужно перейти в данный репозиторий (рисунок 1).

```
Acse11@Acse11 MINGW64 ~/Desktop/cloned_hello (master)
$ cd ../hello/

Acse11@Acse11 MINGW64 ~/Desktop/hello (master)
$
```

Рисунок 124 - Переход в оригинальный репозиторий

Далее надо внести изменения в файл README (рисунок 2) и затем произвести индексацию и коммит (рисунок 3).

```
README
1 This is the Hello World example from the Git tutorial
2 | (changed in original)
```

Рисунок 125 - Изменения в файле README

```
Acse11@Acse11 MINGW64 ~/Desktop/hello (master)
$ git add README

Acse11@Acse11 MINGW64 ~/Desktop/hello (master)
$ git commit -m "Changed README in original repo"
[master 45aa4d5] Changed README in original repo
1 file changed, 2 insertions(+), 1 deletion(-)

Acse11@Acse11 MINGW64 ~/Desktop/hello (master)
$
```

Рисунок 126 - Индексация и коммит новых изменений

Далее требуется перейти в клонированный репозиторий и извлечь изменения с помощью команды git fetch (рисунок 4) и просмотреть историю (рисунок 5).

```
Acse11@Acse11 MINGW64 ~/Desktop/hello (master)
$ cd ../cloned_hello/

Acse11@Acse11 MINGW64 ~/Desktop/cloned_hello (master)
$ 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 1), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), 346 bytes | 43.00 KiB/s, done.
From C:/Users/Acse11/Desktop/hello
 254c5e0..45aa4d5  master    -> origin/master

Acse11@Acse11 MINGW64 ~/Desktop/cloned_hello (master)
$ |
```

Рисунок 127 - Извлечение изменений

```
Acse11@Acse11 MINGW64 ~/Desktop/cloned_hello (master)
$ git hist --all
* 45aa4d5 2025-06-24 | Changed README in original repo (origin/master, origin/HEAD) [Acse11Meent]
* 254c5e0 2025-06-24 | Life is great (HEAD -> master, origin/style) [Acse11Meent]
* 757462c 2025-06-24 | Added README [Acse11Meent]
* 8ac7683 2025-06-24 | Added index.html [Acse11Meent]
* 7d5e215 2025-06-24 | Moved hello.html to lib [Acse11Meent]
* f05a344 2025-06-24 | Add an author comment [Acse11Meent]
* bb11936 2025-06-23 | Add HTML header (tag: ver1) [Acse11Meent]
* 666f580 2025-06-23 | Add standart HTML tags (tag: ver1-beta) [Acse11Meent]
* 2c3a2cf 2025-06-23 | Added h2 tag [Acse11Meent]
* fcff9dc 2025-06-23 | Unrelated change for v [Acse11Meent]
* 77e640d 2025-06-23 | Changes for a and b [Acse11Meent]
* 369b6f6 2025-06-23 | First Commit [Acse11Meent]

Acse11@Acse11 MINGW64 ~/Desktop/cloned_hello (master)
$ |
```

Рисунок 128 - Просмотр истории

При попытке вывести содержимое файла README можно увидеть, что изменения не были внесены (рисунок 6).

```
Acse11@Acse11 MINGW64 ~/Desktop/cloned_hello (master)
$ cat README
This is the Hello World example from the Git tutorial
Acse11@Acse11 MINGW64 ~/Desktop/cloned_hello (master)
$
```

Рисунок 129 - Вывод содержимого файла README

Далее нужно слить извлеченные изменения в ветку master (рисунок 7).

```

This is the Hello World example from the Git tutorial
Acse11@Acse11 MINGW64 ~/Desktop/cloned_hello (master)
$ git merge origin/master
Updating 254c5e0..45aa4d5
Fast-forward
 README | 3 ++-
 1 file changed, 2 insertions(+), 1 deletion(-)

Acse11@Acse11 MINGW64 ~/Desktop/cloned_hello (master)
$

```

Рисунок 130 - Слияние изменений

И после выполнения предыдущего действия при выводе README можно будет увидеть последние изменения (рисунок 8).

```

Acse11@Acse11 MINGW64 ~/Desktop/cloned_hello (master)
$ cat README
This is the Hello World example from the Git tutorial
(changed in original)
Acse11@Acse11 MINGW64 ~/Desktop/cloned_hello (master)
$

```

Рисунок 131 - Вывод содержимого файла README

Также существует команда, объединяющая функции git fetch и git merge, которая показана на рисунке 9.

```

Acse11@Acse11 MINGW64 ~/Desktop/cloned_hello (master)
$ git pull
Already up to date.

Acse11@Acse11 MINGW64 ~/Desktop/cloned_hello (master)
$ |

```

Рисунок 132 - Команда git pull

Далее требуется добавить локальную ветку, которая будет отслеживать удаленную ветку (рисунок 10).

```
Already up to date.

Acse11@Acse11 MINGW64 ~/Desktop/cloned_hello (master)
$ git branch --track style origin/style
branch 'style' set up to track 'origin/style'.

Acse11@Acse11 MINGW64 ~/Desktop/cloned_hello (master)
$ git branch -a
* master
  style
remotes/origin/HEAD -> origin/master
remotes/origin/master
remotes/origin/style

Acse11@Acse11 MINGW64 ~/Desktop/cloned_hello (master)
$ git hist --max-count=2
* 45aa4d5 2025-06-24 | Changed README in original repo (HEAD -> master, origin/master, /HEAD) [Acse11Meent]
* 254c5e0 2025-06-24 | Life is great (origin/style, style) [Acse11Meent]

Acse11@Acse11 MINGW64 ~/Desktop/cloned_hello (master)
$ |
```

Рисунок 133 - Добавление локальной ветки

Далее необходимо создать чистый репозиторий (рисунок 11).

```
Acse11@Acse11 MINGW64 ~/Desktop/cloned_hello (master)
$ cd ..

Acse11@Acse11 MINGW64 ~/Desktop
$ git clone --bare hello hello.git
Cloning into bare repository 'hello.git'...
done.

Acse11@Acse11 MINGW64 ~/Desktop
$ ls hello.git
HEAD  config  description  hooks/  info/  objects/  packed-refs  refs/

Acse11@Acse11 MINGW64 ~/Desktop
$
```

Рисунок 134 - Создание чистого репозитория

Для добавления удаленного репозитория используется команда, показанная на рисунке 12.

```
Acse11@Acse11 MINGW64 ~/Desktop
$ cd hello

Acse11@Acse11 MINGW64 ~/Desktop/hello (master)
$ git remote add shared ../hellot.git

Acse11@Acse11 MINGW64 ~/Desktop/hello (master)
$ |
```

Рисунок 135 - Добавление удаленного репозитория

Затем требуется научиться отправлять изменения в удаленный репозиторий. Для этого сначала надо внести изменения, проиндексировать и произвести коммит (рисунок 13-14).



Рисунок 136 - Внесение изменений в файл

```
Acse11@Acse11 MINGW64 ~/Desktop/hello (master)
$ git checkout master
Already on 'master'
M   README

Acse11@Acse11 MINGW64 ~/Desktop/hello (master)
$ git add README

Acse11@Acse11 MINGW64 ~/Desktop/hello (master)
$ git commit -m "Added shared comment to README"
[master 6712487] Added shared comment to README
1 file changed, 1 insertion(+), 1 deletion(-)

Acse11@Acse11 MINGW64 ~/Desktop/hello (master)
$
```

Рисунок 137 - Индексация и коммит

Далее надо отправить изменения в общий репозиторий, используя команду `git push shared master` (рисунок 15).

```
Acse11@Acse11 MINGW64 ~/Desktop/hello (master)
$ git push shared master
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 12 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 340 bytes | 340.00 KiB/s, done.
Total 3 (delta 2), reused 0 (delta 0), pack-reused 0
To ../hello.git
45aa4d5..6712487 master -> master
```

Рисунок 138 - Отправка изменений в общий репозиторий

Для извлечения общих изменений нужно перейти в клонированный каталог и выполнить перечень команд (рисунок 16).

```
Acse11@Acse11 MINGW64 ~/Desktop/hello (master)
$ cd ../cloned_hello/

Acse11@Acse11 MINGW64 ~/Desktop/cloned_hello (master)
$ |
```

Рисунок 139 - Переход в клонированный репозиторий


```

Acse11@Acse11 MINGW64 ~/Desktop/cloned_hello (master)
$ git remote add shared ../hello.git
error: remote shared already exists.

Acse11@Acse11 MINGW64 ~/Desktop/cloned_hello (master)
$ git branch --track shared master
fatal: a branch named 'shared' already exists

Acse11@Acse11 MINGW64 ~/Desktop/cloned_hello (master)
$ 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 2), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), 320 bytes | 45.00 KiB/s, done.
From ../hello
* branch          master      -> FETCH_HEAD
  45aa4d5..6712487 master      -> shared/master
Updating 45aa4d5..6712487
Fast-forward
 README | 2 +-
 1 file changed, 1 insertion(+), 1 deletion(-)

Acse11@Acse11 MINGW64 ~/Desktop/cloned_hello (master)
$ cat README
This is the Hello World example from the Git tutorial
(changed in original and pushed to shared)
Acse11@Acse11 MINGW64 ~/Desktop/cloned_hello (master)
$

```

Рисунок 140 - Команды, извлекающие общие изменения

Для настройки git сервера нужно выполнить команду, показанную на рисунке 18. Затем в другом окне можно проверить работу сервера, сделав копию проекта hello (рисунок 18).

```

Acse11@Acse11 MINGW64 ~/Desktop/cloned_hello (master)
$ git daemon --verbose --export-all --base-path=.
[8520] Ready to rumble
|

```

Рисунок 141 - Настройка сервера

```

Acse11@Acse11 MINGW64 ~/Desktop/hello (master)
$ git clone ~/Desktop/hello.git netwok_hello
Cloning into 'netwok_hello'...
done.

Acse11@Acse11 MINGW64 ~/Desktop/hello (master)
$ cd netwok_hello

Acse11@Acse11 MINGW64 ~/Desktop/hello/netwok_hello (master)
$ ls
README  a.html  b.html  index.html  lib/  v.html

Acse11@Acse11 MINGW64 ~/Desktop/hello/netwok_hello (master)
$

```

Рисунок 142 - Клонирование проекта

_УП_7

Работа с Git

Для того, чтобы просмотреть список настроенных удалённых репозиториях, необходимо запустить команду `git remote` (рисунок 1).

```
Acse11@Acse11 MINGW64 ~/Desktop
$ git clone https://github.com/schacon/ticgit
Cloning into 'ticgit'...
remote: Enumerating objects: 1857, done.
remote: Total 1857 (delta 0), reused 0 (delta 0), pack-reused 1857 (from 1)
Receiving objects: 100% (1857/1857), 334.06 KiB | 1.93 MiB/s, done.
Resolving deltas: 100% (837/837), done.

Acse11@Acse11 MINGW64 ~/Desktop
$ c dti
bash: c: command not found

Acse11@Acse11 MINGW64 ~/Desktop
$ cd ticgit/

Acse11@Acse11 MINGW64 ~/Desktop/ticgit (master)
$ git remote
origin

Acse11@Acse11 MINGW64 ~/Desktop/ticgit (master)
$ |
```

Рисунок 143 – Клонирование репозитория и просмотр удалённых репозиториях

Можно указать ключ `-v`, чтобы просмотреть адреса для чтения и записи, привязанные к репозиторию (рисунок 2).

```
Acse11@Acse11 MINGW64 ~/Desktop/ticgit (master)
$ git remote -v
origin https://github.com/schacon/ticgit (fetch)
origin https://github.com/schacon/ticgit (push)

Acse11@Acse11 MINGW64 ~/Desktop/ticgit (master)
$
```

Рисунок 144 - Просмотр удалённых репозиториях с ключом `-v`

Для добавления удалённого репозитория с новым именем используется команда `git remote add` (рисунок 3).

```
Acse11@Acse11 MINGW64 ~/Desktop/ticgit (master)
$ git remote add pb https://github.com/paulboone/ticgit

Acse11@Acse11 MINGW64 ~/Desktop/ticgit (master)
$ git remote -v
origin https://github.com/schacon/ticgit (fetch)
origin https://github.com/schacon/ticgit (push)
pb https://github.com/paulboone/ticgit (fetch)
pb https://github.com/paulboone/ticgit (push)
```

Рисунок 145 - Добавление удалённого репозитория

После задания имени репозиторию впоследствии его можно использовать вместо указания полного пути (рисунок 4).

```
Acse11@Acse11 MINGW64 ~/Desktop/ticgit (master)
$ git fetch pb
remote: Enumerating objects: 43, done.
remote: Counting objects: 100% (22/22), done.
remote: Total 43 (delta 22), reused 22 (delta 22), pack-reused 21 (from 1)
Unpacking objects: 100% (43/43), 5.99 KiB | 82.00 KiB/s, done.
From https://github.com/paulboone/ticgit
* [new branch]      master    -> pb/master
* [new branch]      ticgit    -> pb/ticgit

Acse11@Acse11 MINGW64 ~/Desktop/ticgit (master)
$ |
```

Рисунок 146 - Использование имени вместо пути

Для получения данных из удалённых проектов используется команда git fetch (рисунок 5).

```
Acse11@Acse11 MINGW64 ~/Desktop/ticgit (master)
$ git fetch ~/Desktop/cloned_hello/
remote: Enumerating objects: 40, done.
remote: Counting objects: 100% (40/40), done.
remote: Compressing objects: 100% (33/33), done.
remote: Total 40 (delta 15), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (40/40), 3.34 KiB | 44.00 KiB/s, done.
From C:/Users/Acse11/Desktop/cloned_hello
* branch            HEAD      -> FETCH_HEAD

Acse11@Acse11 MINGW64 ~/Desktop/ticgit (master)
$ |
```

Рисунок 147 - Получение данных из удаленных проектов

Для отправки изменений в удаленный репозиторий используется команда git push (рисунок 6).

```
Acse11@Acse11 MINGW64 ~/Desktop/ticgit (master)
$ git push origin master
remote: Permission to schacon/ticgit.git denied to Acse11Meent.
fatal: unable to access 'https://github.com/schacon/ticgit/': The requested URL
returned error: 403

Acse11@Acse11 MINGW64 ~/Desktop/ticgit (master)
```

Рисунок 148 - Отправка изменений в удаленный репозиторий

Для получения информации об одном из удалённых репозиториях, можно использовать команду git remote show (рисунок 7).

```
e1ena@LAPTOP-I3I3QM5I MINGW64 ~/Documents/ticgit (master)
$ git remote show origin
* remote origin
Fetch URL: https://github.com/schacon/ticgit
Push URL: https://github.com/schacon/ticgit
HEAD branch: master
Remote branches:
  master tracked
  ticgit 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)

e1ena@LAPTOP-I3I3QM5I MINGW64 ~/Documents/ticgit (master)
$ |
```

Рисунок 149 - Информация об удаленном репозитории

Нет доступа к репозиторию

Для переименования удаленных репозитория используется команда `git remote rename` (рисунок 8).

```
Acse11@Acse11 MINGW64 ~/Desktop/ticgit (master)
$ git remote rename pb paul
Renaming remote references: 100% (2/2), done.

Acse11@Acse11 MINGW64 ~/Desktop/ticgit (master)
$ git remote
origin
paul

Acse11@Acse11 MINGW64 ~/Desktop/ticgit (master)
$
```

Рисунок 150 - Переименование удаленного репозитория

Для удаления удаленного репозитория нужно выполнить команду `git remote remove` (рисунок 9).

```
Acse11@Acse11 MINGW64 ~/Desktop/ticgit (master)
$ git remote remove paul

Acse11@Acse11 MINGW64 ~/Desktop/ticgit (master)
$ git remote
origin

Acse11@Acse11 MINGW64 ~/Desktop/ticgit (master)
$ |
```

Рисунок 151 - Удаление удаленного репозитория

Просмотреть существующие теги можно с помощью команды `git tag` (рисунок 10).

```

Acse11@Acse11 MINGW64 ~/Desktop/hello (master)
$ git tag
ver1
ver1-beta

Acse11@Acse11 MINGW64 ~/Desktop/hello (master)
$ |

```

Рисунок 152 - Просмотр тегов

11. Для создания аннотированной метки нужно выполнить команду, показанную на рисунке

```

Acse11@Acse11 MINGW64 ~/Desktop/hello (master)
C$ git tag -a v2.1 -m "My version 2.1"

Acse11@Acse11 MINGW64 ~/Desktop/hello (master)
$ git tag
v2.1
ver1

```

Рисунок 153 - Создание аннотированной метки

Команда git show осуществляет просмотр данных тегов вместе с коммитом (рисунок 12).

```

Acse11@Acse11 MINGW64 ~/Desktop/hello (master)
$ git show v2.1
tag v2.1
Tagger: Acse11Meent <dshalkhykov@mail.ru>
Date: Tue Jun 24 22:30:39 2025 +0300

My version 2.1

commit 671248786143f20629401fb3cf4b569c3ad03aab (HEAD -> master, tag: v2.1, shar
ed/master)
Author: Acse11Meent <dshalkhykov@mail.ru>
Date: Tue Jun 24 22:12:43 2025 +0300

    Added shared comment to README

diff --git a/README b/README
index 9091eb5..0d63084 100644
--- a/README
+++ b/README
@@ -1,2 +1,2 @@
 This is the Hello World example from the Git tutorial
-(changed in original)
 \ No newline at end of file
+(changed in original and pushed to shared)
 \ No newline at end of file

Acse11@Acse11 MINGW64 ~/Desktop/hello (master)
$

```

Рисунок 154 - Просмотр данных тега

Для создания легковесной метки не нужно передавать опции -a, -s и -m, надо указать только название (рисунок 13). Просмотр данных такой метки осуществляется также с помощью git show (рисунок 14).


```
Acse11@Acse11 MINGW64 ~/Desktop/hello (master)
$ git tag v2.1-lw

Acse11@Acse11 MINGW64 ~/Desktop/hello (master)
$ git tag
v2.1
v2.1-lw
ver1
ver1-beta
```

Рисунок 155 - Создание легковесной метки

```
Acse11@Acse11 MINGW64 ~/Desktop/hello (master)
$ git show v2.1-lw
commit 671248786143f20629401fb3cf4b569c3ad03aab (HEAD -> master, tag: v2.1-lw, tag: v2.1, shared/master)
Author: Acse11Meent <dshalkhykov@mail.ru>
Date: Tue Jun 24 22:12:43 2025 +0300

    Added shared comment to README

diff --git a/README b/README
index 9091eb5..0d63084 100644
--- a/README
+++ b/README
@@ -1,2 +1,2 @@
 This is the Hello World example from the Git tutorial
-(changed in original)
 \ No newline at end of file
+(changed in original and pushed to shared)
 \ No newline at end of file

Acse11@Acse11 MINGW64 ~/Desktop/hello (master)
$ |
```

Рисунок 156 - Просмотр данных тега

Для отметки определенного коммита тегом надо указать его хэш (рисунки 15-17).

```
Acse11@Acse11 MINGW64 ~/Desktop/hello (master)
$ git log --pretty=oneline
671248786143f20629401fb3cf4b569c3ad03aab (HEAD -> master, tag: v2.1-lw, tag: v2.1, shared/master) Added
shared comment to README
45aa4d57c8510eb6ac15db9c3b7aafa3be7a552c Changed README in original repo
254c5e0a1d3f5a4ccac23d76f932ae35a4371390 (style) Life is great
757462c62953925a9a4e50753708cd4c16dad64f Added README
8ac7683bf1dad1220b881c2540db6c86e75d4cb1 Added index.html
7d5e215088f78bacfd325358e12d84094cbb3b1d Moved hello.html to lib
f05a34428efe574bab143391ae454b2b6aefdc79 Add an author comment
bb11936fc0a1b33a0037794ca421dc684e73f6d8 (tag: ver1) Add HTML header
666f580d1374f4842813f43e51ee60a31c9875b1 (tag: ver1-beta) Add standart HTML tags
2c3a2cf173058667513118913e1d6b27d06afb49 Added h2 tag
fcff9dc042740210ea39700380ac107c98c0128a Unrelated change for v
77e640d94af8102252fc6b99179cc519a4dfa0a Changes for a and b
369b6f6343640d581447962f2dca24600563cbee First Commit

Acse11@Acse11 MINGW64 ~/Desktop/hello (master)
$ |
```

Рисунок 157 - Просмотр истории

```
Acse11@Acse11 MINGW64 ~/Desktop/hello (master)
$ git tag -a ver3 45aa4d
hint: Waiting for your editor to close the file...
[3880x0634/222522:585x5800P:\services\worker_storage_ss(2072)
```

Рисунок 158 - Создание тега определенному коммиту

```

t > TAG_EDITMSG
1 Tag message1
2 #
3 # Write a message for tag:
4 # ver3
5 # Lines starting with '#' will be ignored.
6

```

Рисунок 159 - Ввод сообщения в текстовом редакторе

Данные этого тега можно просмотреть аналогичным образом (рисунок 18).

```

Acse11@Acse11 MINGW64 ~/Desktop/hello (master)
$ git show ver3
tag ver3
Tagger: Acse11Meent <dshalkhykov@mail.ru>
Date: Tue Jun 24 22:35:19 2025 +0300

Tag message1

commit 45aa4d57c8510eb6ac15db9c3b7aafa3be7a552c (tag: ver3)
Author: Acse11Meent <dshalkhykov@mail.ru>
Date: Tue Jun 24 22:07:18 2025 +0300

    Changed README in original repo

diff --git a/README b/README
index 3f16465..9091eb5 100644
--- a/README
+++ b/README
@@ -1,2 @@
-This is the Hello World example from the Git tutorial
\ No newline at end of file
+This is the Hello World example from the Git tutorial
+(changed in original)
\ No newline at end of file

```

Рисунок 160 - Просмотр данных тега

По умолчанию, команда git push не отправляет теги на удалённые сервера. Нужно выполнить команду git push shared (рисунок 19).

```

Acse11@Acse11 MINGW64 ~/Desktop/hello (master)
$ git push shared v2.1
Enumerating objects: 1, done.
Counting objects: 100% (1/1), done.
Writing objects: 100% (1/1), 167 bytes | 167.00 KiB/s, done.
Total 1 (delta 0), reused 0 (delta 0), pack-reused 0
To ../hello.git
* [new tag]          v2.1 -> v2.1

```

Рисунок 161 - Отправка тега на удаленный сервер

Можно использовать опцию --tags для команды git push. В таком случае все теги отправятся на удалённый сервер (рисунок 20).

```

Acse11@Acse11 MINGW64 ~/Desktop/hello (master)
$ git push shared --tags
Enumerating objects: 1, done.
Counting objects: 100% (1/1), done.
Writing objects: 100% (1/1), 166 bytes | 166.00 KiB/s, done.
Total 1 (delta 0), reused 0 (delta 0), pack-reused 0
To ../hello.git
* [new tag]          v2.1-lw -> v2.1-lw
* [new tag]          ver3 -> ver3

```

Рисунок 162 - Отправка всех тегов на сервер

Для того, чтобы удалить тег, надо использовать команду `git tag` с параметром `-d` (рисунок 21).

```

Acse11@Acse11 MINGW64 ~/Desktop/hello (master)
$ git tag -d v2.1-lw
Deleted tag 'v2.1-lw' (was 6712487)

Acse11@Acse11 MINGW64 ~/Desktop/hello (master)
$

```

Рисунок 163 - Удаление тега

Для удаления тега с сервера используется команда, показанная на рисунке 22.

```

Acse11@Acse11 MINGW64 ~/Desktop/hello (master)
$ git push shared :refs/tags/v2.1-lw
To ../hello.git
- [deleted]          v2.1-lw

Acse11@Acse11 MINGW64 ~/Desktop/hello (master)
$ |

```

Рисунок 164 - Удаление тегов с сервера

Для того, чтобы получить версии файлов, на которые указывает тег, можно выполнить `git checkout` для тега. Однако, это переведёт репозиторий в состояние «detached HEAD». Если в состоянии «detached HEAD» внести изменения и сделать коммит, то тег не изменится, при этом новый коммит не будет относиться ни к какой из веток, а доступ к нему можно будет получить только по его хэшу. Поэтому в таком случае следует создать новую ветку (рисунки 23-24).

```
Acse11@Acse11 MINGW64 ~/Desktop/hello (master)
$ git checkout v2.1
Note: switching to 'v2.1'.

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 6712487 Added shared comment to README
Acse11@Acse11 MINGW64 ~/Desktop/hello ((v2.1))
$
```

Рисунок 165 - Переключение на метку

```
Acse11@Acse11 MINGW64 ~/Desktop/hello ((v2.1))
$ git checkout -b version2 v2.1
Switched to a new branch 'version2'

Acse11@Acse11 MINGW64 ~/Desktop/hello (version2)
$
```

Рисунок 166 - Создание новой ветки

Можно создать псевдонимы (алиасы) для команд. Создание алиасов и примеры их использования показаны на рисунках 25-30.

```
Acse11@Acse11 MINGW64 ~/Desktop/hello (version2)
$ git config --global alias.co checkout

Acse11@Acse11 MINGW64 ~/Desktop/hello (version2)
$ git config --global alias.br branch

Acse11@Acse11 MINGW64 ~/Desktop/hello (version2)
$ git config --global alias.ci commit

Acse11@Acse11 MINGW64 ~/Desktop/hello (version2)
$ git config --global alias.st status
```

Рисунок 167 - Задание алиасов

```
Acse11@Acse11 MINGW64 ~/Desktop/hello (version2)
$ git config --global alias.unstage 'reset HEAD --'

Acse11@Acse11 MINGW64 ~/Desktop/hello (version2)
$ |
```

Рисунок 168 - Создание псевдонима исключения файла из индекса

```

Acse11@Acse11 MINGW64 ~/Desktop/hello (version2)
$ git reset HEAD --fileA
error: unknown option 'fileA'
usage: git reset [--mixed | --soft | --hard | --merge | --keep] [-q] [<commit>]
or: git reset [-q] [<tree-ish>] [--] <pathspec>...
or: git reset [-q] [--pathspec-from-file [--pathspec-file-nul]] [<tree-ish>]
or: git reset --patch [<tree-ish>] [--] [<pathspec>...]
or: DEPRECATED: git reset [-q] [--stdin [-z]] [<tree-ish>]

    -q, --quiet                be quiet, only report errors
    --no-refresh               skip refreshing the index after reset
    --mixed                   reset HEAD and index
    --soft                    reset only HEAD
    --hard                   reset HEAD, index and working tree
    --merge                   reset HEAD, index and working tree
    --keep                    reset HEAD but keep local changes
    --recurse-submodules[=<reset>] control recursive updating of submodules
    -p, --patch               select hunks interactively
    -N, --intent-to-add       record only the fact that removed paths will be added later
    --pathspec-from-file <file> read pathspec from file
                                with --pathspec-from-file, pathspec elements are separated with NUL character
    --pathspec-file-nul      DEPRECATED (use --pathspec-from-file=- instead): paths are separated with NUL character
    -z                       DEPRECATED (use --pathspec-from-file=- instead): read paths from <stdin>
    --stdin                  DEPRECATED (use --pathspec-from-file=- instead): read paths from <stdin>

Acse11@Acse11 MINGW64 ~/Desktop/hello (version2)
$ git reset HEAD -- fileA
Acse11@Acse11 MINGW64 ~/Desktop/hello (version2)
$

```

Рисунок 169 - Использование созданного псевдонима

```

Acse11@Acse11 MINGW64 ~/Desktop/hello (version2)
$ git config --global alias.last 'log -1 HEAD'
Acse11@Acse11 MINGW64 ~/Desktop/hello (version2)
$

```

Рисунок 170 - Создание алиаса для просмотра последнего коммита

```

Acse11@Acse11 MINGW64 ~/Desktop/hello (version2)
$ git last
commit 671248786143f20629401fb3cf4b569c3ad03aab (HEAD -> version2, tag: v2.1, shared/master, master)
Author: Acse11Meent <dshalkhykov@mail.ru>
Date: Tue Jun 24 22:12:43 2025 +0300

    Added shared comment to README

Acse11@Acse11 MINGW64 ~/Desktop/hello (version2)
$ |

```

Рисунок 171 - Результат работы созданного алиаса

```

Acse11@Acse11 MINGW64 ~/Desktop/hello (version2)
$ git config --global alias.visual '!gitk'

Acse11@Acse11 MINGW64 ~/Desktop/hello (version2)
$ |

```

Рисунок 172 - Создание псевдонима внешней команды

_УП_8

Работа с Git

Для начала следует создать репозиторий, создать 3 файла и добавить их в коммит (рисунок 1).


```
Acse11@Acse11 MINGW64 ~/Desktop/hello_two (version2)
$ git add README test.rb LICENSE

Acse11@Acse11 MINGW64 ~/Desktop/hello_two (version2)
$ git commit -a -m 'Initial commit of my project'
On branch version2
nothing to commit, working tree clean

Acse11@Acse11 MINGW64 ~/Desktop/hello_two (version2)
$ |
```

Рисунок 1 - Индексация и коммит 3 файлов

Затем надо создать ветку testing и переключиться на нее (рисунки 2-3).

```
Acse11@Acse11 MINGW64 ~/Desktop/hello_two (version2)
$ git branch testing
```

Рисунок 2 - Создание ветки testing

```
Acse11@Acse11 MINGW64 ~/Desktop/hello_two (version2)
$ git checkout testing
Switched to branch 'testing'

Acse11@Acse11 MINGW64 ~/Desktop/hello_two (testing)
$
```

Рисунок 3 - Переключение на ветку testing

Далее надо внести изменения в файл test.rb и создать коммит (рисунок 4).

```
Acse11@Acse11 MINGW64 ~/Desktop/hello_two (master)
$ git add test.rb
Acse11@Acse11 MINGW64 ~/Desktop/hello_two (master)
$ git commit -m 'New changes'
[master 339534e] New changes
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 test.rb

Acse11@Acse11 MINGW64 ~/Desktop/hello_two (master)
$
```

Рисунок 4 - Индексация и коммит файла test.rb

Затем необходимо переключиться на ветку master и внести изменения в файл test.rb на этой ветке (рисунки 5-6).

```
Acse11@Acse11 MINGW64 ~/Desktop/hello_two (testing)
$ git checkout master
Switched to branch 'master'
```

Рисунок 5 - Переключение на ветку master

```
Acse11@Acse11 MINGW64 ~/Desktop/hello_two (master)
$ git add test.rb

Acse11@Acse11 MINGW64 ~/Desktop/hello_two (master)
$ git commit -m 'Another changes'
On branch master
nothing to commit, working tree clean
```

Рисунок 6 - Еще индексация и коммит test.rb

Команда `git checkout -b` позволяет сразу создать и переключиться на ветку (рисунок 7).

```
Acse11@Acse11 MINGW64 ~/Desktop/hello_two (master)
$ git checkout -b iss53
Switched to a new branch 'iss53'

Acse11@Acse11 MINGW64 ~/Desktop/hello_two (iss53)
$ |
```

Рисунок 7 - Создание и переключение на ветку iss53

В новой ветке нужно внести в файл изменения (рисунок 8).

```
Acse11@Acse11 MINGW64 ~/Desktop/hello_two (iss53)
$ git add index.html

Acse11@Acse11 MINGW64 ~/Desktop/hello_two (iss53)
$ git commit -m 'Added html'
On branch iss53
nothing to commit, working tree clean
```

Рисунок 8 - Индексация и коммит файла index.html

Далее нужно переключить ветку на master (рисунок 9).

```
Acse11@Acse11 MINGW64 ~/Desktop/hello_two (iss53)
$ git checkout master
Switched to branch 'master'

Acse11@Acse11 MINGW64 ~/Desktop/hello_two (master)
$
```

Рисунок 9 - Переключение на ветку master

Затем надо на ветке hotfix добавить изменения в файл index.html, а затем слить эту ветку и master (рисунки 10-11).

```
Acse11@Acse11 MINGW64 ~/Desktop/hello_two (hotfix)
$ git add index.html

Acse11@Acse11 MINGW64 ~/Desktop/hello_two (hotfix)
$ git commit -a -m 'Fixed the broken email address'
On branch hotfix
nothing to commit, working tree clean

Acse11@Acse11 MINGW64 ~/Desktop/hello_two (hotfix)
$ |
```

Рисунок 10 - Индексация и коммит index.html на ветке hotfix

```

Acse11@Acse11 MINGW64 ~/Desktop/hello_two (hotfix)
$ git checkout master
Switched to branch 'master'

Acse11@Acse11 MINGW64 ~/Desktop/hello_two (master)
$ git merge hotfix
Already up to date.

Acse11@Acse11 MINGW64 ~/Desktop/hello_two (master)
$ |

```

Рисунок 11 - Переключение на ветку master и объединение с веткой hotfix

После слияния ветку hotfix можно удалить (рисунок 12).1

```

Acse11@Acse11 MINGW64 ~/Desktop/hello_two (master)
$ git branch -d hotfix
Deleted branch hotfix (was 339534e).

Acse11@Acse11 MINGW64 ~/Desktop/hello_two (master)
$ |

```

Рисунок 12 - Удаление ветки hotfix

Затем требуется внести изменения в iss53, переключиться на master и слить эти ветки (рисунки 13-14).

```

Acse11@Acse11 MINGW64 ~/Desktop/hello_two (master)
$ git add index.html

Acse11@Acse11 MINGW64 ~/Desktop/hello_two (master)
$ git commit -a -m 'Finished the new footer'
On branch master
nothing to commit, working tree clean

Acse11@Acse11 MINGW64 ~/Desktop/hello_two (master)
$ |

```

Рисунок 13 - Индексация и коммит index.html на ветке iss53

```

Acse11@Acse11 MINGW64 ~/Desktop/hello_two (master)
$ git checkout master
Already on 'master'

Acse11@Acse11 MINGW64 ~/Desktop/hello_two (master)
$ git merge iss53
Already up to date.

Acse11@Acse11 MINGW64 ~/Desktop/hello_two (master)
$

```

Рисунок 14 - Слияние веток

После этого ветку iss53 нужно удалить (рисунок 15).

```

Acse11@Acse11 MINGW64 ~/Desktop/hello_two (master)
$ git branch -d iss53
Deleted branch iss53 (was 339534e).

Acse11@Acse11 MINGW64 ~/Desktop/hello_two (master)
$

```

_УП_9

Работа с Git

Команда `git branch` делает несколько больше, чем просто создаёт и удаляет ветки. При запуске без параметров, можно получить простой список имеющихся веток (рисунок 1). Символ *, стоящий перед веткой `master` указывает на ветку, на которую указывает HEAD).

```
Acse11@Acse11 MINGW64 ~/Desktop/hello_two (master)
$ git branch
* master
  style
  testing
  version2
```

Рисунок 173 - Список существующих веток

Чтобы посмотреть последний коммит на каждой из веток, необходимо выполнить команду `git branch -v` (рисунок 2).

```
Acse11@Acse11 MINGW64 ~/Desktop/hello_two (master)
$ git branch -v
* master    339534e New changes
  style     254c5e0 Life is great
  testing   594be4d Initial commit of my project
  version2  594be4d Initial commit of my project
```

Рисунок 174 - Список веток с последними коммитами

Опции `--merged` и `--no-merged` могут отфильтровать этот список для вывода только тех веток, которые слиты или ещё не слиты в текущую ветку (рисунки 3-4).

```
Acse11@Acse11 MINGW64 ~/Desktop/hello_two (master)
$ git branch --merged
* master
  style
```

Рисунок 175 - Список веток слитых с текущей

```
Acse11@Acse11 MINGW64 ~/Desktop/hello_two (master)
$ git branch --no-merged
  testing
  version2
```

Рисунок 176 - Список веток не слитых с текущей

Затем следует удалить ветку `testing` (рисунок 5). При наличии ошибок для удаления можно использовать параметр `-D`.

```

Acse11@Acse11 MINGW64 ~/Desktop/hello_two (master)
$ git branch -D testing
Deleted branch testing (was 5f33d13).

Acse11@Acse11 MINGW64 ~/Desktop/hello_two (master)
$

```

Рисунок 177 - Удаление ветки

Для получения списка удалённых веток и дополнительной информации используется команда `git remote show` (рисунок 6).

```

Acse11@Acse11 MINGW64 ~/Desktop/hello_two (master)
$ git remote show shared
* remote shared
Fetch URL: ../hello.git
Push URL: ../hello.git
HEAD branch: master
Remote branches:
  master tracked
  style new (next fetch will store in remotes/shared)
Local refs configured for 'git push':
  master pushes to master (fast-forwardable)
  style pushes to style (up to date)

Acse11@Acse11 MINGW64 ~/Desktop/hello_two (master)
$ |

```

Рисунок 178 - Просмотр удаленных веток

Для отправления изменений на удалённый сервер используется команда `git push <remote> <branch>` (рисунок 7).

```

Acse11@Acse11 MINGW64 ~/Desktop/hello_two (master)
$ git push shared master
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Delta compression using up to 12 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (2/2), 241 bytes | 241.00 KiB/s, done.
Total 2 (delta 1), reused 0 (delta 0), pack-reused 0
To ../hello.git
  6712487..339534e master -> master

Acse11@Acse11 MINGW64 ~/Desktop/hello_two (master)
$

```

Рисунок 179 - Отправка изменений

Далее при получении обновлений с сервера будет показана ссылка на удаленную ветку (рисунок 8).

```

Acse11@Acse11 MINGW64 ~/Desktop/hello_two (master)
$ git fetch shared
Acse11@Acse11 MINGW64 ~/Desktop/hello_two (master)

```

Рисунок 180 - Выполнение команды `git fetch`

При необходимости можно создать локальную ветку на основе удаленной (рисунок 9).


```

Acse11@Acse11 MINGW64 ~/Desktop/hello_two (master)
$ git checkout -b serv shared/master
Switched to a new branch 'serv'
branch 'serv' set up to track 'shared/master'.

Acse11@Acse11 MINGW64 ~/Desktop/hello_two (serv)
$ |

```

Рисунок 181 - Создание ветки на основе удаленной ветки

Для удаления веток на удаленном сервере используется команда, показанная на рисунке 10.

```

Acse11@Acse11 MINGW64 ~/Desktop/hello_two (serv)
$ git push shared :serv
error: unable to delete 'serv': remote ref does not exist
error: failed to push some refs to '../hello.git'

```

Рисунок 182 - Удаление ветки на сервере

Простой способ выполнить слияние двух веток – это команда merge. Другой способ – использование команды rebase, что означает перебазирование (рисунок 11). Это работает следующим образом: берётся общий родительский снимок двух веток (текущей, и той, поверх которой вы выполняете перебазирование), определяется дельта каждого коммита текущей ветки и сохраняется во временный файл, текущая ветка устанавливается на последний коммит ветки, поверх которой выполняется перебазирование, а затем по очереди применяются дельты из временных файлов.

Далее после этого надо переключиться на ветку master и выполнить перемотку.

```

Acse11@Acse11 MINGW64 ~/Desktop/hello_two (serv)
$ git checkout exp
Switched to branch 'exp'

Acse11@Acse11 MINGW64 ~/Desktop/hello_two (exp)
$ git rebase master
Current branch exp is up to date.

```

Рисунок 183 - Перемещение изменений

При наличии ответвления от ветки (сначала было ответвление на ветку se, а затем от нее на ветку cl), чтобы переместить изменения можно осуществить действия, показанные на рисунках 12-15.

```

Acse11@Acse11 MINGW64 ~/Desktop/hello_two (master)
$ git rebase --onto master se cl
Current branch cl is up to date.

Acse11@Acse11 MINGW64 ~/Desktop/hello_two (cl)
$

```

Рисунок 184 - Перемещение изменений с параметром onto

```

Acse11@Acse11 MINGW64 ~/Desktop/hello_two (c1)
$ git checkout master
Switched to branch 'master'

Acse11@Acse11 MINGW64 ~/Desktop/hello_two (master)
$ git merge c1
Already up to date.

Acse11@Acse11 MINGW64 ~/Desktop/hello_two (master)
$

```

Рисунок 185 - Слияние веток master и c1

```

Acse11@Acse11 MINGW64 ~/Desktop/hello_two (master)
$ git rebase master se
Current branch se is up to date.

```

Рисунок 186 - Перемещение изменений

```

Acse11@Acse11 MINGW64 ~/Desktop/hello_two (se)
$ git checkout master
Switched to branch 'master'

Acse11@Acse11 MINGW64 ~/Desktop/hello_two (master)
$ git merge se
Already up to date.

```

Рисунок 187 - Слияние веток master и se

После этого перемещение будет осуществлено и ветки можно удалить (рисунок 16).

```

Acse11@Acse11 MINGW64 ~/Desktop/hello_two (master)
$ git branch -d c1
Deleted branch c1 (was 339534e).

Acse11@Acse11 MINGW64 ~/Desktop/hello_two (master)
$ git branch -d se
Deleted branch se (was 339534e).

```

Рисунок 188 - Удаление веток c1 и se

_УП_10

Уважаемые студенты!

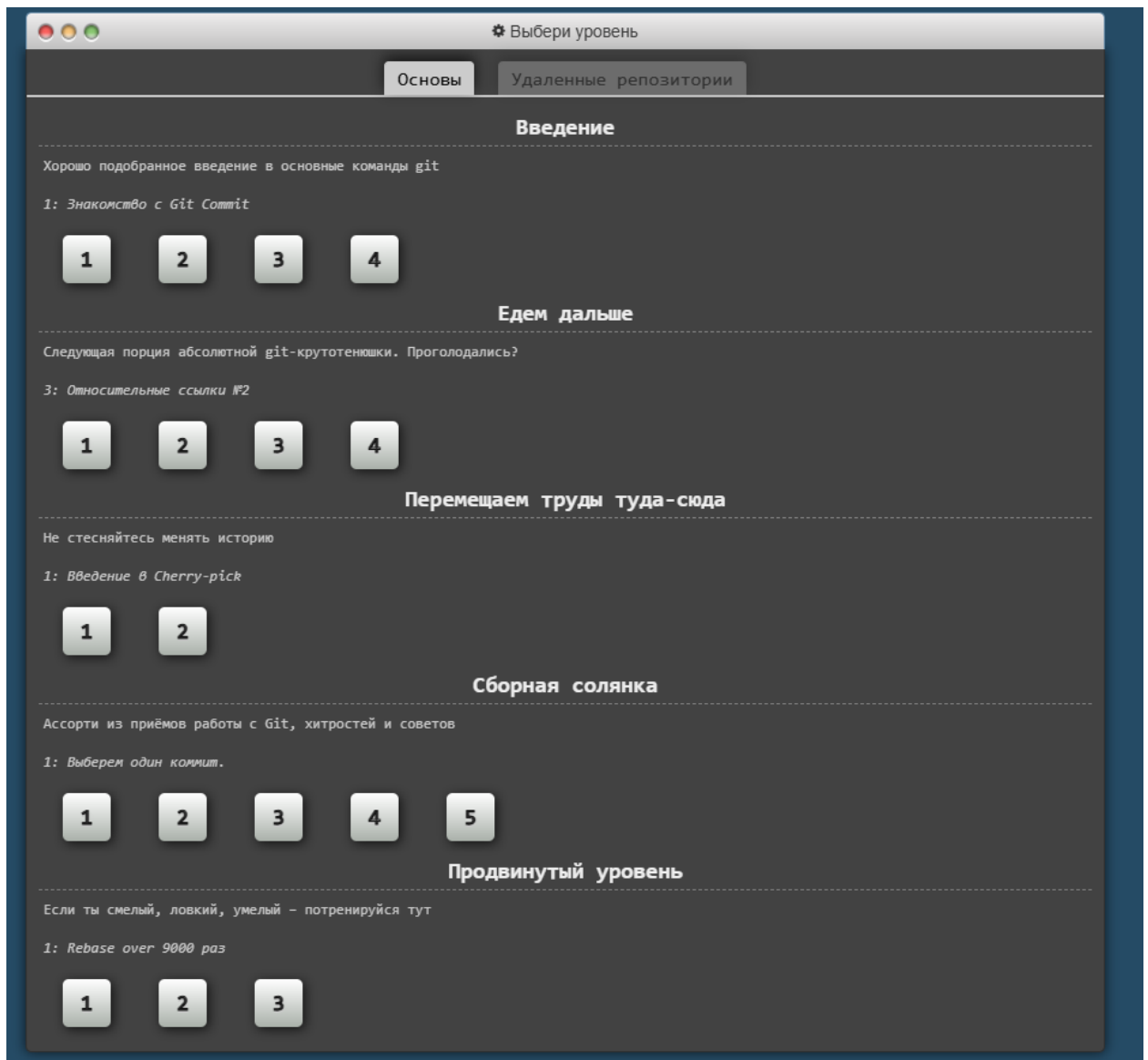
В данном файле должны располагаться скриншоты и описание выполненных работ с ресурса:

Все скриншоты с выполненными заданиями должны быть индивидуальными. Можете их подписать, либо на заднем фоне сделать обозначение ваших Ф.И.О.

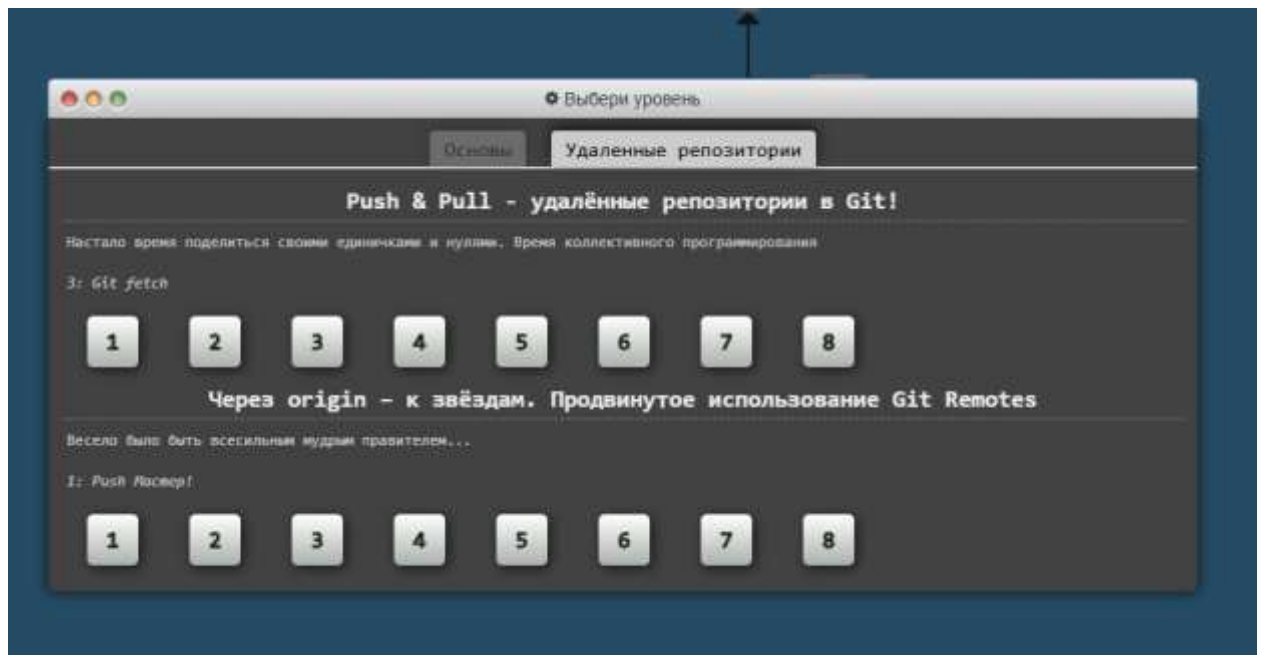
*Если ваша работа будет выполнена не самостоятельно, то комплект отчетной документации по практике будет возвращен на доработку.

<https://learngitbranching.js.org/>

Выполняем все уровни:



И не забываем про удаленные репозитории:



Где?