

Министерство образования Республики Беларусь
Учреждение образования
«Белорусский государственный университет
информатики и радиоэлектроники»

ОТЧЕТ
по выполнению практического занятия
на тему
GIT How To

Выполнил:

Студент гр. 210902 Жигало В.А.

Проверил:

Давыдович К.И.

Минск 2024

01 Скачайте учебные материалы

Скачайте учебные материалы здесь:

- <https://githowto.com/githowto.zip>

Задание

githowto	18.11.2024 9:58	Папка с файлами
githowto	17.11.2024 23:10	Сжатая ZIP-папка 1 933 КБ

Решение

02 Распакуйте учебные материалы

Пакет учебных материалов должен иметь главную директорию `githowto` с двумя поддиректориями:

- `repositories`: пустая директория, в которой будут располагаться ваши репозитории.
- `files`: заранее упакованные файлы для того, чтобы вы могли продолжить работать с учебными материалами на любом этапе. Если вы застрянете, просто скопируйте нужный урок в свой репозиторий.

Задание

files	18.11.2024 9:59	Папка с файлами
repositories	18.11.2024 10:00	Папка с файлами

Решение

01 Установка имени и электронной почты

Если вы никогда ранее не использовали Git, для начала вам необходимо осуществить установку. Выполните следующие команды, чтобы Git узнал ваше имя и электронную почту. Эти данные используются для подписи изменений сделанных вами, что позволит отслеживать, кто и когда сделал изменения в файле.

Выполните

```
git config --global user.name "Your Name"
git config --global user.email "your_email@whatever.com"
```

Задание

```
C:\Users\vlad> git config --global user.name "Vladislav"
C:\Users\vlad> git config --global user.email "your_email@whatever.com"
```

Решение

02 Имя ветки по умолчанию

Мы будем использовать `main` в качестве имени ветки по умолчанию. Чтобы установить его, выполните следующую команду:

Выполните

```
git config --global init.defaultBranch main
```

Задание

```
C:\Users\vladz> git config --global init.defaultBranch main
```

Решение

03 Корректная обработка окончаний строк

Для пользователей Unix/Mac:

Выполните

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

Для пользователей Windows:

Выполните

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

Задание

```
C:\Users\vladz> git config --global core.autocrlf true
C:\Users\vladz> git config --global core.safecrlf warn
```

Решение

01 Создайте страницу «Hello, World»

Начните работу в пустой директории (например, `respositories`, если вы скачали архив с предыдущего шага) с создания пустой поддиректории `work`, затем войдите в неё и создайте там файл `hello.html` с таким содержанием:

Выполните

```
mkdir work
cd work
touch hello.html
```

Файл: hello.html

```
Hello, World
```

Задание

```
C:\Users\vladz> mkdir work

C:\Users\vladz> cd work

C:\Users\vladz\work> touch hello.html
'touch' is not recognized as an internal or external command,
operable program or batch file.
```

Решение

02 Создайте репозиторий

Теперь у вас есть директория с одним файлом. Чтобы создать Git-репозиторий из этой директории, выполните команду `git init`.

Выполните

```
git init
```

Результат

```
Initialized empty Git repository in /home/alex/ghowto/repositories/work/.git/
```

Задание

```
C:\Users\vladz\work> git init
Initialized empty Git repository in C:/Users/vladz/work/.git/
```

Решение

03 Добавьте страницу в репозиторий

Теперь давайте добавим в репозиторий страницу «Hello, World».

Выполните

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

Вы увидите...

Результат

```
$ git add hello.html
$ git commit -m "Initial commit"
[main (root-commit) 5836970] Initial commit
1 file changed, 1 insertion(+)
create mode 100644 hello.html
```

Задание

```
C:\Users\vladz\work> git add hello.html

C:\Users\vladz\work> git commit -m "Initial Commit"
[main (root-commit) 62a0d4c] Initial Commit
1 file changed, 1 insertion(+)
create mode 100644 hello.html
```

Решение

01 Проверьте состояние репозитория

Используйте команду `git status`, чтобы проверить текущее состояние репозитория.

Выполните

```
git status
```

Вы увидите:

Результат

```
$ git status
On branch main
nothing to commit, working tree clean
```

Задание

```
C:\Users\vladz\work> git status
On branch main
nothing to commit, working tree clean
```

Решение

01 Измените страницу «Hello, World»

Добавим кое-какие HTML-теги к нашему приветствию. Измените содержимое файла на:

Файл: hello.html

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

Задание

```
<> Laba4.html  <> hello.html X
C: > GitHowTo > githowto > repositories > work > <> hello.html > h1
1  <h1>Hello, World!</h1>
```

Решение

02 Проверьте состояние

Теперь проверьте состояние рабочей директории.

Выполните

```
git status
```

Вы увидите...

Результат

```
$ git status
On branch main
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")
```

Задание

```
C:\Users\vladz\work> git status
On branch main
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")
```

Решение

01 Добавьте изменения

Теперь дайте команду Git проиндексировать изменения. Проверьте состояние:

Выполните

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

Вы увидите:

Результат

```
$ git add hello.html
$ git status
On branch main
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   hello.html
```

Задание

```
C:\Users\vladz\work> git add hello.html

C:\Users\vladz\work> git status
On branch main
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   hello.html
```

Решение

01 Закоммитьте изменения

Достаточно об индексации. Давайте сделаем коммит того, что мы проиндексировали, в репозиторий.

Когда вы ранее использовали `git commit` для коммита первоначальной версии файла `hello.html` в репозиторий, вы включили метку `-m`, которая делает комментарий в командной строке. Команда `commit` позволит вам интерактивно редактировать комментарий для коммита. Теперь давайте это проверим.

Если вы опустите метку `-m` из командной строки, Git перенесет вас в редактор по вашему выбору. Редактор выбирается из следующего списка (в порядке приоритета):

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

У меня переменная `EDITOR` установлена в `vim`. Если вы предпочитаете GUI-редактор, то теперь можно использовать VS Code в качестве Git-редактора.

Сделайте коммит сейчас и проверьте состояние.

Выполните

```
git commit
```

Задание

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

Решение

02 Проверьте состояние

В конце давайте еще раз проверим состояние.

Выполните

```
git status
```

Вы увидите:

Результат

```
$ git status
On branch main
nothing to commit, working tree clean
```

Задание

```
C:\Users\vladz\work> git status
On branch main
nothing to commit, working tree clean
```

Решение

01 Первое изменение: Добавьте стандартные теги страницы

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

Файл: hello.html

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

Задание

```
Laba4.html  hello.html X
C: > Users > vladz > work > hello.html > html
1  <html>
2  |   <body>
3  |       <h1>Hello, World!</h1>
4  |   </body>
5  | </html>|
```

Решение

02 Добавьте это изменение

Теперь добавьте это изменение в индекс Git.

Выполните

```
git add hello.html
```

Задание

```
C:\Users\vladz\work> git add hello.html
```

Решение

03 Второе изменение: Добавьте заголовки HTML

Теперь добавьте заголовки HTML (секцию `<head>`) к странице «Hello, World».

Файл: hello.html

```
<html>
<head>
</head>
<body>
  <h1>hello, world!</h1>
</body>
</html>
```

Задание

```
Laba4.html  hello.html X
C: > Users > vladz > work > hello.html > html
1  <html>
2  |   <head>
3  |   </head>
4  |   <body>
5  |       <h1>Hello, World!</h1>
6  |   </body>
7  | </html>
```

Решение

04 Проверьте текущий статус

Выполните

```
git status
```

Вы увидите:

Результат

```
$ git status
On branch main
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
```

Задание

```
C:\Users\vladz\work> git status
On branch main
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
```

Решение

05 Коммит

Произведите коммит проиндексированного изменения (значение по умолчанию), а затем еще раз проверьте состояние.

Выполните

```
git commit -m "Added standard HTML page tags"
git status
```

Вы увидите:

Результат

```
$ git commit -m "Added standard HTML page tags"
[main 46afaff] Added standard HTML page tags
 1 file changed, 5 insertions(+), 1 deletion(-)
$ git status
On branch main
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")
```

Задание

```
C:\Users\vladz\work> git commit -m "Added standard HTML page tags"
[main 44ec824] Added standard HTML page tags
1 file changed, 5 insertions(+), 1 deletion(-)

C:\Users\vladz\work> git status
On branch main
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")
```

Решение

06 Добавьте второе изменение

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

Выполните

```
git add .
git status
```

Мы использовали текущую директорию (`.`) в качестве аргумента для добавления. Это самый короткий и удобный способ добавления всех изменений в текущей директории. Но поскольку Git добавляет в индекс *всё*, то *не лишним* будет проверить состояние репозитория перед запуском `add`, просто чтобы убедиться, что вы не добавили какой-то файл, который не следовало бы добавлять.

Я просто хотел показать вам трюк с `add .`, в дальнейшем мы будем добавлять все файлы явно.

Вы увидите:

Результат

```
$ git add .
$ git status
On branch main
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   hello.html
```

Задание

```
C:\Users\vladz\work> git add .

C:\Users\vladz\work> git status
On branch main
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   hello.html
```

Решение

07 Сделайте коммит второго изменения

Выполните

```
git commit -m "Added HTML header"
```

Результат

```
$ git commit -m "Added HTML header"
[main b7614c1] Added HTML header
1 file changed, 2 insertions(+)
```

Задание

```
C:\Users\vladz\work> git commit -m "Added HTML header"
[main 246dee1] Added HTML header
1 file changed, 2 insertions(+)
```

Решение

9. История

Цели

- Научиться просматривать историю проекта.

Получение списка произведенных изменений — функция команды `git log`.

Выполните

```
git log
```

Вы увидите:

Результат

```
$ git log
commit b7614c1aea1ffbc46400fe1a163842d6ec620a43
Author: Alexander Shvets <alex@github.com>
Date: Tue Nov 28 05:51:38 2023 -0600

    Added HTML header

commit 46afaff2232fc3d564c40f65cb82e7e94839a1bb
Author: Alexander Shvets <alex@github.com>
Date: Tue Nov 28 05:51:38 2023 -0600

    Added standard HTML page tags

commit 78433de967102f2b59d0a8a60eb397b2663ed282
Author: Alexander Shvets <alex@github.com>
Date: Tue Nov 28 05:51:38 2023 -0600

    Added h1 tag

commit 58369706affbc1c27fa03a65fc7a05847278045f
Author: Alexander Shvets <alex@github.com>
Date: Tue Nov 28 05:51:38 2023 -0600

    Initial commit
```

Задание

```
C:\Users\vladz\work> git log
commit 246dee12cbb6bd6201c786550e4718dab78f4cf (HEAD -> main)
Author: Vladislav <your_email@whatever.com>
Date: Tue Dec 10 00:41:46 2024 +0300

    Added HTML header

commit 44ec824e5fcb69baff029ffc20fbd46dc255d204
Author: Vladislav <your_email@whatever.com>
Date: Tue Dec 10 00:39:24 2024 +0300

    Added standard HTML page tags

commit 529ad15c14d20d9e6f84431ba47e9f88fe9d995c
Author: Vladislav <your_email@whatever.com>
Date: Tue Dec 10 00:34:28 2024 +0300

    Initial Commit

commit 62a0d4ce245ff17f30f283fc78f51059048a166d
Author: Vladislav <your_email@whatever.com>
Date: Tue Dec 10 00:15:52 2024 +0300

    Initial Commit
```

Решение

01 Однострочная история

Вы полностью контролируете то, что отображает `log`. Мне, например, нравится однострочный формат:

Выполните

```
git log --pretty=oneline
```

Вы увидите:

Результат

```
$ git log --oneline
b7614c1 Added HTML header
46afaff Added standard HTML page tags
78433de Added h1 tag
5836970 Initial commit
```

Задание

```
C:\Users\vladz\work> git log --pretty=oneline
246dee12cbb6bd6201c786550e4718dab78f4cf (HEAD -> main) Added HTML header
44ec824e5fcb69baff029ffc20fbd46dc255d204 Added standard HTML page tags
529ad15c14d20d9e6f84431ba47e9f88fe9d995c Initial Commit
62a0d4ce245ff17f30f283fc78f51059048a166d Initial Commit
```

Решение

04 Конечный формат лога

Со временем, я решил, что для большей части моей работы мне подходит следующий формат лога.

Выполните

```
git log --pretty=format:"%h %ad | %s%d [%an]" --date=short
```

Выглядит это примерно так:

Результат

```
$ git log --pretty=format:"%h %ad | %s%d [%an]" --date=short
b7614c1 2023-11-28 | Added HTML header (HEAD -> main) [Alexander Shvets]
46afaff 2023-11-28 | Added standard HTML page tags [Alexander Shvets]
78433de 2023-11-28 | Added h1 tag [Alexander Shvets]
5836970 2023-11-28 | Initial commit [Alexander Shvets]
```

Давайте рассмотрим его в деталях:

- `--pretty="..."` — определяет формат вывода.
- `%h` — укороченный хеш коммита.
- `%ad` — дата коммита.
- `|` — просто визуальный разделитель.
- `%s` — комментарий.
- `%d` — дополнения коммита («головы» веток или теги).
- `%an` — имя автора.
- `--date=short` — сохраняет формат даты коротким и симпатичным.

Таким образом, каждый раз, когда вы захотите посмотреть лог, вам придется много печатать. К счастью, существует несколько опций конфигурации Git, позволяющих настроить формат вывода истории по умолчанию:

Задание

```
C:\Users\vladz\work> git log --pretty=format:"%h %ad | %s%d [%an]" --date=short
246dee1 2024-12-10 | Added HTML header (HEAD -> main) [Vladislav]
44ec824 2024-12-10 | Added standard HTML page tags [Vladislav]
529ad15 2024-12-10 | Initial Commit [Vladislav]
62a0d4c 2024-12-10 | Initial Commit [Vladislav]
```

Решение

01 Получите хеши предыдущих коммитов

Выполните

```
git log
```

Результат

```
$ git log
b7614c1 2023-11-28 | Added HTML header (HEAD -> main) [Alexander Shvets]
46afaff 2023-11-28 | Added standard HTML page tags [Alexander Shvets]
78433de 2023-11-28 | Added h1 tag [Alexander Shvets]
5836970 2023-11-28 | Initial commit [Alexander Shvets]
```

Просмотрите историю изменений и найдите хеш первого коммита. Он должен быть в последней строке результата `git log`. Используйте этот хеш (достаточно первых 7 символов) в команде ниже. Затем проверьте содержимое файла `hello.html`.

Задание

```
C:\Users\vladz\work> git log
commit 246dee12cbb6bd6201c786550e4718dab78f4cf (HEAD -> main)
Author: Vladislav <your_email@whatever.com>
Date: 2024-12-10

    Added HTML header

commit 44ec824e5fcb69baff029ffc20fbd46dc255d204
Author: Vladislav <your_email@whatever.com>
Date: 2024-12-10

    Added standard HTML page tags

commit 529ad15c14d20d9e6f84431ba47e9f88fe9d995c
Author: Vladislav <your_email@whatever.com>
Date: 2024-12-10

    Initial Commit

commit 62a0d4ce245ff17f30f283fc78f51059048a166d
Author: Vladislav <your_email@whatever.com>
Date: 2024-12-10

    Initial Commit
```

Решение

Выполните

```
git checkout <hash>
cat hello.html
```

Многие команды Git принимают хеши коммитов в качестве аргументов. Хеши коммитов будут отличаться в разных репозиториях, поэтому когда вы видите, что в команде есть пометка `<hash>`, то это значит, что вам надо подставить вместо нее реальный хеш из вашего репозитория.

Вы увидите:

Результат

```
$ git checkout 5836970
Note: switching to '5836970'.

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 5836970 Initial commit
$ cat hello.html
Hello, World!
```

Задание

```
C:\Users\vladz\work> git checkout 62a0d4c
Note: switching to '62a0d4c'.

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 62a0d4c Initial Commit

C:\Users\vladz\work> cat hello.html
'cat' is not recognized as an internal or external command,
operable program or batch file.
```

Решение

Выполните

```
git switch main
cat hello.html
```

Вы увидите:

Результат

```
$ git switch main
Previous HEAD position was 5836970 Initial commit
Switched to branch 'main'
$ cat hello.html
<html>
  <head>
  </head>
  <body>
    <h1>Hello, World!</h1>
  </body>
</html>
```

Задание

```
C:\Users\vladz\work> git switch main
Previous HEAD position was 62a0d4c Initial Commit
Switched to branch 'main'

C:\Users\vladz\work> cat hello.html
'cat' is not recognized as an internal or external command,
operable program or batch file.
```

Решение

Выполните

```
git tag v1
git log
```

Результат

```
$ git tag v1
$ git log
b7614c1 2023-11-28 | Added HTML header (HEAD -> main, tag: v1) [Alexander Shvets]
46afaff 2023-11-28 | Added standard HTML page tags [Alexander Shvets]
78433de 2023-11-28 | Added h1 tag [Alexander Shvets]
5836970 2023-11-28 | Initial commit [Alexander Shvets]
```

Задание

```
C:\Users\vladz\work> git tag v1

C:\Users\vladz\work> git log
commit 246dee12cbb6bd6201c786550e4718dab78f4cf (HEAD -> main, tag: v1)
Author: Vladislav <your_email@whatever.com>
Date: 2024-12-10

    Added HTML header

commit 44ec824e5fcb69baff029ffc20fbd46dc255d204
Author: Vladislav <your_email@whatever.com>
Date: 2024-12-10

    Added standard HTML page tags

commit 529ad15c14d20d9e6f84431ba47e9f88fe9d995c
Author: Vladislav <your_email@whatever.com>
Date: 2024-12-10

    Initial Commit

commit 62a0d4ce245ff17f30f283fc78f51059048a166d
Author: Vladislav <your_email@whatever.com>
Date: 2024-12-10

    Initial Commit
```

Решение

Выполните

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

Результат

```
$ git checkout v1^  
Note: switching to 'v1^'.  
  
You are in 'detached HEAD' state. You can look around, make experimental  
changes and commit them, and you can discard any commits you make in this  
state without impacting any branches by switching back to a branch.  
  
If you want to create a new branch to retain commits you create, you may  
do so (now or later) by using -c with the switch command. Example:  
  
    git switch -c <new-branch-name>  
  
Or undo this operation with:  
  
    git switch -  
  
Turn off this advice by setting config variable advice.detachedHead to false  
  
HEAD is now at 46afaff Added standard HTML page tags  
$ cat hello.html  
<html>  
  <body>  
    <h1>Hello, World!</h1>  
  </body>  
</html>
```

Задание

```
C:\Users\vladz\work> git checkout v1~1  
Note: switching to 'v1~1'.  
  
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 44ec824 Added standard HTML page tags  
  
C:\Users\vladz\work> cat hello.html  
'cat' is not recognized as an internal or external command,  
operable program or batch file.
```

Решение

Выполните

```
git tag v1-beta
git log
```

Результат

```
$ git tag v1-beta
$ git log
46afaff 2023-11-28 | Added standard HTML page tags (HEAD, tag: v1-beta) [Alexander Shvets]
78433de 2023-11-28 | Added h1 tag [Alexander Shvets]
5836970 2023-11-28 | Initial commit [Alexander Shvets]
```

Задание

```
C:\Users\vladz\work> git tag v1-beta

C:\Users\vladz\work> git log
commit 44ec824e5fcb69baff029ffc20fbd46dc255d204 (HEAD, tag: v1-beta)
Author: Vladislav <your_email@whatever.com>
Date: 2024-12-10

    Added standard HTML page tags

commit 529ad15c14d20d9e6f84431ba47e9f88fe9d995c
Author: Vladislav <your_email@whatever.com>
Date: 2024-12-10

    Initial Commit

commit 62a0d4ce245ff17f30f283fc78f51059048a166d
Author: Vladislav <your_email@whatever.com>
Date: 2024-12-10

    Initial Commit
```

Решение

03 Переключение по имени тега

Теперь попробуйте попереключаться между двумя отмеченными версиями.

Выполните

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

Результат

```
$ git checkout v1
Previous HEAD position was 46afaff Added standard HTML page tags
HEAD is now at b7614c1 Added HTML header
$ git checkout v1-beta
Previous HEAD position was b7614c1 Added HTML header
HEAD is now at 46afaff Added standard HTML page tags
```

Задание

```
C:\Users\vladz\work> git checkout v1
Previous HEAD position was 44ec824 Added standard HTML page tags
HEAD is now at 246dee1 Added HTML header

C:\Users\vladz\work> git checkout v1-beta
Previous HEAD position was 246dee1 Added HTML header
HEAD is now at 44ec824 Added standard HTML page tags
```

Решение

04 Просмотр тегов с помощью команды `tag`

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

Выполните

```
git tag
```

Результат

```
$ git tag
v1
v1-beta
```

Задание

```
C:\Users\vladz\work> git tag
v1
v1-beta
```

Решение

05 Просмотр Тегов в логах

Вы также можете посмотреть теги в логе.

Выполните

```
git log main --all
```

Результат

```
$ git log main --all
b7614c1 2023-11-28 | Added HTML header (tag: v1, main) [Alexander Shvets]
46afaff 2023-11-28 | Added standard HTML page tags (HEAD, tag: v1-beta) [Alexander Shvets]
78433de 2023-11-28 | Added h1 tag [Alexander Shvets]
5836970 2023-11-28 | Initial commit [Alexander Shvets]
```

Задание

```
C:\Users\vladz\work> git log main --all
commit 246dee12cbb6bd6201c786550e4718dab78f4cf (tag: v1, main)
Author: Vladislav <your_email@whatever.com>
Date: 2024-12-10

    Added HTML header

commit 44ec824e5fcb69baff029ffc20fbd46dc255d204 (HEAD, tag: v1-beta)
Author: Vladislav <your_email@whatever.com>
Date: 2024-12-10

    Added standard HTML page tags

commit 529ad15c14d20d9e6f84431ba47e9f88fe9d995c
Author: Vladislav <your_email@whatever.com>
Date: 2024-12-10

    Initial Commit

commit 62a0d4ce245ff17f30f283fc78f51059048a166d
Author: Vladislav <your_email@whatever.com>
Date: 2024-12-10

    Initial Commit
```

Решение

01 Переключитесь на ветку `main`

Убедитесь, что вы находитесь на последнем коммите ветки `main`, прежде чем продолжить работу.

Выполните

```
git switch main
```

Задание

```
C:\Users\vladz\work> git switch main
Previous HEAD position was 44ec824 Added standard HTML page tags
Switched to branch 'main'
```

Решение

02 Измените `hello.html`

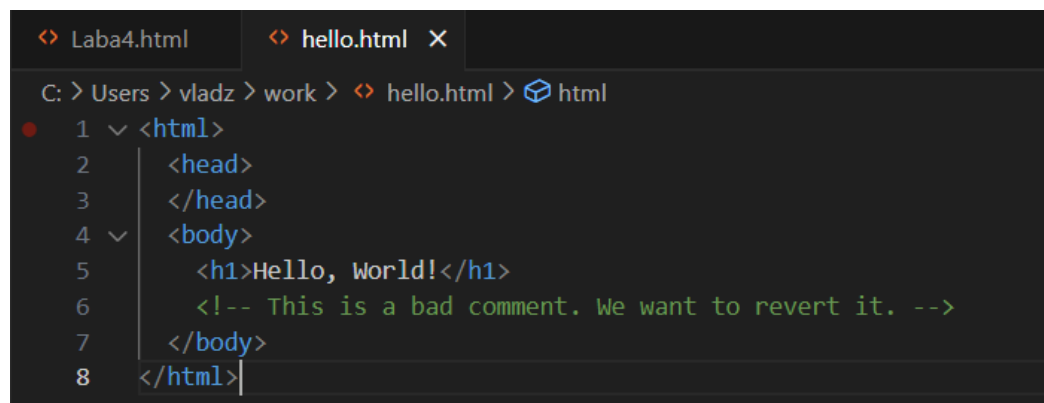
Иногда после того как вы изменили файл в рабочей директории, вы передумали и хотите просто вернуться к тому, что уже было закоммичено. Команда `checkout` справится с этой задачей.

Внесите изменение в файл `hello.html` в виде нежелательного комментария.

Файл: `hello.html`

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

Задание



```
Lab4.html  hello.html X
C: > Users > vladz > work > hello.html > html
1  <html>
2    <head>
3    </head>
4  <body>
5    <h1>Hello, World!</h1>
6    <!-- This is a bad comment. We want to revert it. -->
7    </body>
8  </html>
```

Решение

03 Проверьте состояние

Сначала проверьте состояние рабочей директории.

Выполните

```
git status
```

Результат

```
$ git status
On branch main
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")
```

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

Задание

```
C:\Users\vladz\work> git status
On branch main
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")
```

Решение

04 Отмена изменений в рабочем каталоге

Используйте команду `checkout` для переключения в версию файла `hello.html` в репозитории.

Выполните

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

Результат

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

Задание

```
C:\Users\vladz\work> git checkout hello.html
Updated 1 path from the index

C:\Users\vladz\work> git status
On branch main
nothing to commit, working tree clean

C:\Users\vladz\work> cat hello.html
'cat' is not recognized as an internal or external command,
operable program or batch file.
```

Решение

01 Измените файл и проиндексируйте изменения

Внесите изменение в файл `hello.html` в виде нежелательного комментария

Файл: hello.html

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

Проиндексируйте это изменение.

Выполните

```
git add hello.html
```

Задание

```
C:\Users\vladz\work> git add hello.html
```

Решение

02 Проверьте состояние

Проверьте состояние нежелательного изменения.

Выполните

```
git status
```

Результат

```
$ git status
On branch main
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   hello.html
```

Задание

```
C:\Users\vladz\work> git status
On branch main
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   hello.html
```

Решение

03 Сбросьте области подготовки

Команда `reset` сбрасывает область подготовки к `HEAD`. Это очищает область подготовки от изменений, которые мы только что проиндексировали.

Выполните

```
git reset HEAD hello.html
```

Результат

```
$ git reset HEAD hello.html
Unstaged changes after reset:
M   hello.html
```

Задание

```
C:\Users\vladz\work> git reset HEAD hello.html
Unstaged changes after reset:
M       hello.html
```

Решение

04 Переключитесь на версию коммита

Выполните

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

Результат

```
$ git checkout hello.html
Updated 1 path from the index
$ git status
On branch main
nothing to commit, working tree clean
```

Задание

```
C:\Users\vladz\work> git checkout hello.html
Updated 1 path from the index

C:\Users\vladz\work> git status
On branch main
nothing to commit, working tree clean
```

Решение

14. Отмена коммитов

Цели

- Научиться отменять коммиты в локальном репозитории.

01 Отмена коммитов

Иногда вы понимаете, что новые коммиты являются неверными, и хотите их отменить. Есть несколько способов решения этого вопроса, здесь мы будем использовать самый безопасный.

Мы отменим коммит путем создания нового коммита, отменяющего нежелательные изменения.

02 Измените файл и сделайте коммит

Измените файл `hello.html` на следующий.

Файл: `hello.html`

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

Выполните

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

Задание

```
C:\Users\vladz\work> git add hello.html

C:\Users\vladz\work> git commit -m "Oops, we didn't want this commit"
[main ec6311b] Oops, we didn't want this commit
1 file changed, 1 insertion(+)
```

Решение

04 Проверьте лог

Проверка лога показывает нежелательные и отмененные коммиты в наш репозиторий.

Выполните

```
git log
```

Результат

```
$ git log
86364a1 2023-11-28 | Revert "Oops, we didn't want this commit" (HEAD -> main) [Alexander Shvets]
6a44bec 2023-11-28 | Oops, we didn't want this commit [Alexander Shvets]
b7614c1 2023-11-28 | Added HTML header (tag: v1) [Alexander Shvets]
46afaff 2023-11-28 | Added standard HTML page tags (tag: v1-beta) [Alexander Shvets]
78433de 2023-11-28 | Added h1 tag [Alexander Shvets]
5836970 2023-11-28 | Initial commit [Alexander Shvets]
```

Эта техника будет работать с любым коммитом (хотя, возможно, возникнут конфликты). Она безопасна в использовании даже в публичных ветках удаленных репозиториях.

Задание


```
C:\Users\vladz\work> git log
commit ec6311b85df3b8cfc245f3c1fc43d907e3d1362d (HEAD -> main)
Author: Vladislav <your_email@whatever.com>
Date: 2024-12-10

    Oops, we didn't want this commit

commit 246dee12cbb6bd6201c786550e4718dab78f4cf (tag: v1)
Author: Vladislav <your_email@whatever.com>
Date: 2024-12-10

    Added HTML header

commit 44ec824e5fcb69baff029ffc20fbd46dc255d204 (tag: v1-beta)
Author: Vladislav <your_email@whatever.com>
Date: 2024-12-10

    Added standard HTML page tags

commit 529ad15c14d20d9e6f84431ba47e9f88fe9d995c
Author: Vladislav <your_email@whatever.com>
Date: 2024-12-10

    Initial Commit

commit 62a0d4ce245ff17f30f283fc78f51059048a166d
Author: Vladislav <your_email@whatever.com>
Date: 2024-12-10
```

Решение

03 Для начала отметьте эту ветку

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

Выполните

```
git tag oops
```

Результат

```
$ git log
86364a1 2023-11-28 | Revert "Oops, we didn't want this commit" (HEAD -> main, tag: oops) [Alexander Shvets]
6a44bec 2023-11-28 | Oops, we didn't want this commit [Alexander Shvets]
b7614c1 2023-11-28 | Added HTML header (tag: v1) [Alexander Shvets]
46afaff 2023-11-28 | Added standard HTML page tags (tag: v1-beta) [Alexander Shvets]
78433de 2023-11-28 | Added h1 tag [Alexander Shvets]
5836970 2023-11-28 | Initial commit [Alexander Shvets]
```

Задание

```
C:\Users\vladz\work> git tag oops
```

Решение

04 Сброс к коммиту, предшествующему `oops`

Глядя на историю лога (см. выше), мы видим, что коммит с тегом `v1` является коммитом, предшествующим ошибочному коммиту. Давайте сбросим ветку до этой точки. Поскольку ветка имеет тег, мы можем использовать имя тега в команде сброса `reset` (если она не имеет тега, мы можем использовать хеш коммита).

Выполните

```
git reset --hard v1
git log
```

Результат

```
$ git reset --hard v1
HEAD is now at b7614c1 Added HTML header
$ git log
b7614c1 2023-11-28 | Added HTML header (HEAD -> main, tag: v1) [Alexander Shvets]
46afaff 2023-11-28 | Added standard HTML page tags (tag: v1-beta) [Alexander Shvets]
78433de 2023-11-28 | Added h1 tag [Alexander Shvets]
5836970 2023-11-28 | Initial commit [Alexander Shvets]
```

Задание

```
C:\Users\vladz\work> git reset --hard v1
HEAD is now at 246dee1 Added HTML header

C:\Users\vladz\work> git log
commit 246dee12cbbe6bd6201c786550e4718dab78f4cf (HEAD -> main, tag: v1)
Author: Vladislav <your_email@whatever.com>
Date: 2024-12-10

    Added HTML header

commit 44ec824e5fcb69baff029ffc20fbd46dc255d204 (tag: v1-beta)
Author: Vladislav <your_email@whatever.com>
Date: 2024-12-10

    Added standard HTML page tags

commit 529ad15c14d20d9e6f84431ba47e9f88fe9d995c
Author: Vladislav <your_email@whatever.com>
Date: 2024-12-10

    Initial Commit
```

Решение

05 Ничего никогда не теряется

Что же случается с ошибочными коммитами? Оказывается, что коммиты все еще находятся в репозитории. На самом деле, мы все еще можем на них ссылаться. Помните, в начале этого урока мы создали для отмененного коммита тег `oops`? Давайте посмотрим на *все* коммиты.

Выполните

```
git log --all
```

Результат

```
$ git log --all
b7614c1 2023-11-28 | Added HTML header (HEAD -> main, tag: v1) [Alexander Shvets]
86364a1 2023-11-28 | Revert "Oops, we didn't want this commit" (tag: oops) [Alexander Shvets]
46afaff 2023-11-28 | Added standard HTML page tags (tag: v1-beta) [Alexander Shvets]
6a44bec 2023-11-28 | Oops, we didn't want this commit [Alexander Shvets]
78433de 2023-11-28 | Added h1 tag [Alexander Shvets]
5836970 2023-11-28 | Initial commit [Alexander Shvets]
```

Задание

```
C:\Users\vladz\work> git log --all
commit ec6311b85df3b8cfc245f3c1fc43d907e3d1362d (tag: oops)
Author: Vladislav <your_email@whatever.com>
Date: 2024-12-10

    Oops, we didn't want this commit

commit 246dee12cbb6bd6201c786550e4718dab78f4cf (HEAD -> main, tag: v1)
Author: Vladislav <your_email@whatever.com>
Date: 2024-12-10

    Added HTML header

commit 44ec824e5fcb69baff029ffc20fbd46dc255d204 (tag: v1-beta)
Author: Vladislav <your_email@whatever.com>
Date: 2024-12-10

    Added standard HTML page tags

commit 529ad15c14d20d9e6f84431ba47e9f88fe9d995c
Author: Vladislav <your_email@whatever.com>
Date: 2024-12-10

    Initial Commit

commit 62a0d4ce245ff17f30f283fc78f51059048a166d
Author: Vladislav <your_email@whatever.com>
Date: 2024-12-10
```

Решение

01 Удаление тега oops

Тег `oops` свою функцию выполнил, давайте удалим его. Это позволит внутреннему механизму Git убрать остаточные коммиты, на которые теперь не ссылаются никакие ветки или теги.

Выполните

```
git tag -d oops
git log --all
```

Результат

```
$ git tag -d oops
Deleted tag 'oops' (was 86364a1)
$ git log --all
b7614c1 2023-11-28 | Added HTML header (HEAD -> main, tag: v1) [Alexander Shvets]
46afaff 2023-11-28 | Added standard HTML page tags (tag: v1-beta) [Alexander Shvets]
78433de 2023-11-28 | Added h1 tag [Alexander Shvets]
5836970 2023-11-28 | Initial commit [Alexander Shvets]
```

Задание

```
C:\Users\vladz\work> git tag -d oops
Deleted tag 'oops' (was ec6311b)
```

Решение

01 Измените страницу, а затем сделайте коммит

Добавьте в страницу комментарий автора.

Файл: hello.html

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

Выполните

```
git add hello.html
git commit -m "Added copyright statement"
git log
```

Задание

```
C:\Users\vladz\work> git add hello.html

C:\Users\vladz\work> git commit -m "Added copyright statement"
[main 5de070d] Added copyright statement
1 file changed, 1 insertion(+)
```

Решение

02 Ой... необходим email

Однако после создания коммита вы понимаете, что любой хороший комментарий должен включать электронную почту автора. Обновите страницу `hello.html`, включив в нее email.

Файл: hello.html

```
<!-- Author: Alexander Shvets (alex@github.com) -->
<html>
  <head>
  </head>
  <body>
    <h1>Hello, World!</h1>
  </body>
</html>
```

Задание

```
<> Laba4.html <> hello.html X
C: > Users > vladz > work > <> hello.html > html
1 <!-- Author: Alexander Shvets (alex@github.com) -->
2 <html>
3   <head>
4   </head>
5   <body>
6     <h1>Hello, World!</h1>
7   </body>
8 </html>
```

Решение

03 Измените предыдущий коммит

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

Выполните

```
git add hello.html
git commit --amend -m "Added copyright statement with email"
```

Результат

```
$ git add hello.html
$ git commit --amend -m "Added copyright statement with email"
[main 9288a33] Added copyright statement with email
Date: Tue Nov 28 05:51:38 2023 -0600
1 file changed, 1 insertion(+)
```

Задание

```
C:\Users\vladz\work> git add hello.html

C:\Users\vladz\work> git commit --amend -m "Added copyright statement with email"
[main f100662] Added copyright statement with email
Date: Tue Dec 10 01:23:53 2024 +0300
1 file changed, 1 insertion(+)
```

Решение

01 Создайте ветку

Пришло время сделать нашу страницу более стильной с помощью CSS. Мы будем развивать эту возможность в новой ветке под названием `style`.

Выполните

```
git switch -c style
git status
```

Старожилы могут возразить, что их учили создавать ветки командой `git checkout -b style`. Помните, я упоминал, что команда `checkout` перегружена функциями и флагами? Старый способ все еще работает, но он не рекомендуется. Новая команда `git switch` более выразительна и менее восприимчива к ошибкам. Кроме того, в ней меньше флагов и опций, поэтому ее легче запомнить.

Результат

```
$ git switch -c style
Switched to a new branch 'style'
$ git status
On branch style
nothing to commit, working tree clean
```

Задание

```
C:\Users\vladz\work> git switch -c style
Switched to a new branch 'style'

C:\Users\vladz\work> git status
On branch style
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        git tag oops

nothing added to commit but untracked files present (use "git add" to track)
```

Решение

02 Добавьте файл стилей `style.css`

Выполните

```
touch style.css
```

Файл: style.css

```
h1 {  
  color: red;  
}
```

Выполните

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

Задание

```
C:\Users\vladz\work> git add style.css  
  
C:\Users\vladz\work> git commit -m "Added css stylesheet"  
[style 11b3aa0] Added css stylesheet  
1 file changed, 3 insertions(+)  
create mode 100644 style.css
```

Решение

03 Измените `hello.html`, чтобы он использовал `style.css`.

Файл: hello.html

```
<!-- Author: Alexander Shvets (alex@github.com) -->  
<html>  
  <head>  
    <link type="text/css" rel="stylesheet" media="all" href="style.css" />  
  </head>  
  <body>  
    <h1>Hello, World!</h1>  
  </body>  
</html>
```

Выполните

```
git add hello.html  
git commit -m "Included stylesheet into hello.html"
```

Задание

```
C:\Users\vladz\work> git add hello.html  
  
C:\Users\vladz\work> git commit -m "Included stylesheet into hello.html"  
[style 4e3ff1e] Included stylesheet into hello.html  
1 file changed, 1 insertion(+)
```

Решение

01 Переключение на ветку `main`

Просто используйте команду `git switch` для переключения между ветками.

Выполните

```
git switch main
cat hello.html
```

Результат

```
$ git switch main
Switched to branch 'main'
$ cat hello.html
<!-- Author: Alexander Shvets (alex@github.com) -->
<html>
  <head>
</head>
  <body>
    <h1>Hello, World!</h1>
  </body>
</html>
```

Задание

```
C:\Users\vladz\work> git switch main
Switched to branch 'main'

C:\Users\vladz\work> cat hello.html
'cat' is not recognized as an internal or external command,
operable program or batch file.
```

Решение

02 Вернемся к ветке `style`

Выполните

```
git switch style
cat hello.html
```

Результат

```
$ git switch style
Switched to branch 'style'
$ cat hello.html
<!-- Author: Alexander Shvets (alex@github.com) -->
<html>
  <head>
    <link type="text/css" rel="stylesheet" media="all" href="style.css" />
  </head>
  <body>
    <h1>Hello, World!</h1>
  </body>
</html>
```

Задание

```
C:\Users\vladz\work> git switch style
Switched to branch 'style'

C:\Users\vladz\work> cat hello.html
'cat' is not recognized as an internal or external command,
operable program or batch file.
```

Решение

02 Просмотр различий для конкретного файла

Возможность просмотра истории изменений для конкретного файла очень полезна. Она позволяет увидеть, что именно изменилось, а также кто и когда внес эти изменения. Кроме того, существует возможность увидеть изменения, связанные с конкретным коммитом. Я постоянно пользуюсь этим, чтобы разобраться, почему та или иная штука была реализована именно так в настоящей версии кода.

Команда `show` используется для просмотра изменений в конкретном коммите. Посмотрим изменения в файле `hello.html` в коммите, с тегом `v1` (можно использовать любую ссылку на коммит, например, метку `HEAD`, хеш коммита, имя ветки или тега и т.д.).

Выполните

```
git show v1
```

Результат

```
$ git show v1
b7614c1 2023-11-28 | Added HTML header (tag: v1) [Alexander Shvets]

diff --git a/hello.html b/hello.html
index 6da0629..0d576c4 100644
--- a/hello.html
+++ b/hello.html
@@ -1,4 +1,6 @@
<html>
+ <head>
+ </head>
<body>
  <h1>Hello, World!</h1>
</body>
```

Задание

```
C:\Users\vladz\work> git show v1
commit 246dee12cbb6bd6201c786550e4718dab78f4cf (tag: v1)
Author: Vladislav <your_email@whatever.com>
Date: 2024-12-10

    Added HTML header

diff --git a/hello.html b/hello.html
index 103d1db..807ed91 100644
--- a/hello.html
+++ b/hello.html
@@ -1,4 +1,6 @@
<html>
+ <head>
+ </head>
<body>
  <h1>Hello, World!</h1>
</body>
```

Решение

04 Переместите `style.css` безопасным способом

В большинстве операционных систем переименование и перемещение файлов — это одно и то же. Итак, давайте переместим наш файл `style.css` в директорию `css`, но на этот раз сделаем это безопасно с помощью команды `git mv`. Эта команда гарантирует, что перемещение будет записано в истории Git как перемещение.

Выполните

```
mkdir css
git mv style.css css/style.css
git status
```

Результат

```
$ mkdir css
$ git mv style.css css/style.css
$ git status
On branch style
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    renamed:   style.css -> css/style.css
    renamed:   hello.html -> index.html
```

Задание

```
C:\Users\vladz\work> mkdir css

C:\Users\vladz\work> git mv style.css css/style.css

C:\Users\vladz\work> git status
On branch style
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    renamed:    style.css -> css/style.css
    new file:   git tag oops
    renamed:    hello.html -> index.html
```

Решение

Выполните

```
git commit -m "Renamed hello.html; moved style.css"
git log css/style.css
git log --follow css/style.css
```

Результат

```
$ git commit -m "Renamed hello.html; moved style.css"
[style 0ee0113] Renamed hello.html; moved style.css
 2 files changed, 0 insertions(+), 0 deletions(-)
 rename style.css => css/style.css (100%)
 rename hello.html => index.html (100%)
$ git log css/style.css
0ee0113 2023-11-28 | Renamed hello.html; moved style.css (HEAD -> style) [Alexander Shvets]
$ git log --follow css/style.css
0ee0113 2023-11-28 | Renamed hello.html; moved style.css (HEAD -> style) [Alexander Shvets]
555372e 2023-11-28 | Added css stylesheet [Alexander Shvets]
```

Задание

```
C:\Users\vladz\work> git log css/style.css
commit c73816b5bd477d07773ee40f23456712f9a7ffe5 (HEAD -> style)
Author: Your Name <your_email@whatever.com>
Date: 2024-12-10

    Renamed hello.html; moved style.css

C:\Users\vladz\work> git log --follow css/style.css
commit c73816b5bd477d07773ee40f23456712f9a7ffe5 (HEAD -> style)
Author: Your Name <your_email@whatever.com>
Date: 2024-12-10

    Renamed hello.html; moved style.css

commit 11b3aa056c1496351c31cc639cfadaa0261fd999
Author: Your Name <your_email@whatever.com>
Date: 2024-12-10

    Added css stylesheet
```

Решение

02 Сделайте коммит файла README в ветку main

В настоящее время мы находимся в ветке `style`. Файл `README` не является частью этой ветки, поэтому перед коммитом мы должны переключиться на ветку `main`.

Выполните

```
git switch main
git add README
git commit -m "Added README"
```

Результат

```
$ git switch main
Switched to branch 'main'
$ git add README
$ git commit -m "Added README"
[main ee16740] Added README
1 file changed, 1 insertion(+)
create mode 100644 README
```

Задание

```
C:\Users\vladz\work> git switch main
Switched to branch 'main'

C:\Users\vladz\work> git add README
fatal: pathspec 'README' did not match any files

C:\Users\vladz\work> git commit -m "Added README"
On branch main
Untracked files:
  (use "git add <file>..." to include in what will be committed)
    README.txt

nothing added to commit but untracked files present (use "git add" to track)
```

Решение

01 Слияние веток

Слияние переносит изменения из двух веток в одну. Давайте вернемся к ветке `style` и сольем `main` со `style`.

Выполните

```
git switch style
git merge main
git log --all --graph
```

Результат

```
$ git switch style
Switched to branch 'style'
$ git merge main
Merge made by the 'ort' strategy.
 README | 1 +
 1 file changed, 1 insertion(+)
 create mode 100644 README
$ git log --all --graph
* a33deed 2023-11-28 | Merge branch 'main' into style (HEAD -> style) [Alexander Shvets]
| \
| * ee16740 2023-11-28 | Added README (main) [Alexander Shvets]
| * | 0ee0113 2023-11-28 | Renamed hello.html; moved style.css [Alexander Shvets]
| * | 903eb1d 2023-11-28 | Included stylesheet into hello.html [Alexander Shvets]
| * | 555372e 2023-11-28 | Added css stylesheet [Alexander Shvets]
| /
* 9288a33 2023-11-28 | Added copyright statement with email [Alexander Shvets]
* b7614c1 2023-11-28 | Added HTML header (tag: v1) [Alexander Shvets]
* 46afaff 2023-11-28 | Added standard HTML page tags (tag: v1-beta) [Alexander Shvets]
* 78433de 2023-11-28 | Added h1 tag [Alexander Shvets]
* 5836970 2023-11-28 | Initial commit [Alexander Shvets]
```

Задание

```
C:\Users\vladz\work> git switch style
Switched to branch 'style'

C:\Users\vladz\work> git merge main
Already up to date.

C:\Users\vladz\work> git log --all --graph
* commit c73816b5bd477d07773ee40f23456712f9a7ffe5 (HEAD -> style)
| Author: Your Name <your_email@whatever.com>
| Date: 2024-12-10
|
| Renamed hello.html; moved style.css
* commit 4e3ff1e514188908908404853aa7b73ff258986b
| Author: Your Name <your_email@whatever.com>
| Date: 2024-12-10
|
| Included stylesheet into hello.html
* commit 11b3aa056c1496351c31cc639cfadaa0261fd999
| Author: Your Name <your_email@whatever.com>
| Date: 2024-12-10
|
| Added css stylesheet
```

Решение

01 Вернитесь в `main` и создайте конфликт

Помните, что в нашей ветке `main` страница по-прежнему называется `hello.html`? Переключитесь обратно на ветку `main` и внесите следующие изменения:

```
git switch main
```

Файл: `hello.html`

```
<!-- Author: Alexander Shvets (alex@github.com) -->
<html>
  <head>
    <title>Hello World Page</title>
  </head>
  <body>
    <h1>Hello, World!</h1>
    <p>Let's learn Git together.</p>
  </body>
</html>
```

Выполните

```
git add hello.html
git commit -m "Added meta title"
```

Задание

```
C:\Users\vladz\work> git switch main
Switched to branch 'main'

C:\Users\vladz\work> git add hello.html

C:\Users\vladz\work> git commit -m "Added meta title"
On branch main
Untracked files:
  (use "git add <file>..." to include in what will be committed)
    README.txt
    index.html

nothing added to commit but untracked files present (use "git add" to track)
```

Решение

01 Слияние `main` в ветку `style`

Давайте вернемся в ветку `style` и сольем в нее все последние изменения из ветки `main`.

Выполните

```
git switch style
git merge main
```

Результат

```
$ git switch style
Switched to branch 'style'
$ git merge main
Auto-merging index.html
CONFLICT (content): Merge conflict in index.html
Automatic merge failed; fix conflicts and then commit the result.
```

Похоже, что у нас возник конфликт. Ничего удивительного! Посмотрим, что скажет по этому поводу Git:

Выполните

```
git status
```

Результат

```
$ git status
On branch style
You have unmerged paths.
  (fix conflicts and run "git commit")
  (use "git merge --abort" to abort the merge)

Unmerged paths:
  (use "git add <file>..." to mark resolution)
    both modified:   index.html

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

Задание

```
C:\Users\vladz\work>git switch style
error: The following untracked working tree files would be overwritten by checkout:
    index.html
Please move or remove them before you switch branches.
Aborting

C:\Users\vladz\work> git merge main
Already up to date.

C:\Users\vladz\work> git status
On branch main
Untracked files:
  (use "git add <file>..." to include in what will be committed)
    README.txt
    index.html

nothing added to commit but untracked files present (use "git add" to track)
```

Решение

02 Отмена слияния

Прежде чем мы приступим к разрешению нашего конфликта, хочу заметить, что сразу бросаться к разрешению конфликта не всегда оптимально. Конфликт может быть вызван изменениями, о которых вы не знаете. Или же изменения слишком велики, чтобы разрешить конфликт сразу. По этой причине Git позволяет прервать слияние и вернуться к предыдущему состоянию. Для этого можно воспользоваться командой `git merge --abort`, как это было предложено командой `status`, которую мы выполнили ранее.

Выполните

```
git merge --abort
git status
```

Результат

```
$ git merge --abort
$ git status
On branch style
nothing to commit, working tree clean
```

Задание

```
C:\Users\vladz\work> git merge --abort
fatal: There is no merge to abort (MERGE_HEAD missing).

C:\Users\vladz\work> git status
On branch main
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        README.txt
        index.html

nothing added to commit but untracked files present (use "git add" to track)
```

Решение

01 Сбросьте ветку `style`

Давайте вернемся во времени на ветке `style` к точке *перед* тем, как мы слили ее с веткой `main`. Мы можем сбросить ветку к любому коммиту при помощи команды `reset`. По сути, это изменение указателя ветки на любую точку дерева коммитов.

В этом случае мы хотим вернуться в ветке `style` в точку перед слиянием с `main`. Нам необходимо найти последний коммит перед слиянием.

Выполните

```
git switch style
git log --graph
```

Результат

```
$ git switch style
Already on 'style'
$ git log --graph
* 79ac6fa 2023-11-28 | Resolved merge conflict (HEAD -> style) [Alexander Shvets]
|\
| * 85c14e9 2023-11-28 | Added meta title (main) [Alexander Shvets]
| * a33deed 2023-11-28 | Merge branch 'main' into style [Alexander Shvets]
|/
| * ee16748 2023-11-28 | Added README [Alexander Shvets]
| * 0ee0113 2023-11-28 | Renamed hello.html; moved style.css [Alexander Shvets]
| * 903eb1d 2023-11-28 | Included stylesheet into hello.html [Alexander Shvets]
| * 555372e 2023-11-28 | Added css stylesheet [Alexander Shvets]
|/
* 9288a33 2023-11-28 | Added copyright statement with email [Alexander Shvets]
* b7614c1 2023-11-28 | Added HTML header (tag: v1) [Alexander Shvets]
* 46afaff 2023-11-28 | Added standard HTML page tags (tag: v1-beta) [Alexander Shvets]
* 78433de 2023-11-28 | Added h1 tag [Alexander Shvets]
* 5836979 2023-11-28 | Initial commit [Alexander Shvets]
```

Это немного трудно читать, но, глядя на данные, мы видим, что коммит «Renamed hello.html; moved style.css» был последним на ветке `style` перед слиянием. Давайте сбросим ветку `style` к этому коммиту. Чтобы сослаться на этот коммит, мы либо используем его хеш, либо посчитаем, что этот коммит находится за 2 коммита до `HEAD`, то есть `HEAD~2` в нотации Git.

Выполните

```
git reset --hard HEAD~2
```

Результат

```
$ git reset --hard HEAD~2
HEAD is now at 0ee0113 Renamed hello.html; moved style.css
```

Задание

```
C:\Users\vladz\work> git switch style
Switched to branch 'style'

C:\Users\vladz\work> git reset --hard HEAD~2
HEAD is now at 11b3aa0 Added css stylesheet
```

Решение

01 Перебазируем ветку `style` на ветку `main`

Выполните

```
git switch style
git rebase main
git status
```

Результат

```
$ git switch style
Already on 'style'
$ git rebase main
Rebasing (1/3)Rebasing (2/3)Auto-merging hello.html
CONFLICT (content): Merge conflict in hello.html
error: could not apply 903eb1d... Included stylesheet into hello.html
hint: Resolve all conflicts manually, mark them as resolved with
hint: "git add/rm <conflicted_files>", then run "git rebase --continue".
hint: You can instead skip this commit: run "git rebase --skip".
hint: To abort and get back to the state before "git rebase", run "git rebase --abort".
Could not apply 903eb1d... Included stylesheet into hello.html
$ git status
interactive rebase in progress; onto 85c14e9
Last commands done (2 commands done):
  pick 555372e Added css stylesheet
  pick 903eb1d Included stylesheet into hello.html
Next command to do (1 remaining command):
  pick 0ee0113 Renamed hello.html; moved style.css
(use "git rebase --edit-todo" to view and edit)
You are currently rebasing branch 'style' on '85c14e9'.
(fix conflicts and then run "git rebase --continue")
(use "git rebase --skip" to skip this patch)
(use "git rebase --abort" to check out the original branch)

Unmerged paths:
  (use "git restore --staged <file>..." to unstage)
  (use "git add <file>..." to mark resolution)
    both modified:   hello.html

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

Задание

```
C:\Users\vladz\work> git switch style
Already on 'style'

C:\Users\vladz\work> git rebase main
Successfully rebased and updated refs/heads/style.

C:\Users\vladz\work> git status
On branch style
Untracked files:
  (use "git add <file>..." to include in what will be committed)
    README.txt

nothing added to commit but untracked files present (use "git add" to track)
```

Решение

01 Слейте `style` в `main`

Выполните

```
git switch main
git merge style
```

Результат

```
$ git switch main
Switched to branch 'main'
$ git merge style
Updating 85c14e9..39a1e0f
Fast-forward
 css/style.css      | 3 +++
 hello.html => index.html | 1 +
 2 files changed, 4 insertions(+)
 create mode 100644 css/style.css
 rename hello.html => index.html (73%)
```

Поскольку последний коммит в `main` предшествует последнему коммиту ветки `style`, Git может выполнить ускоренное слияние, просто переместив указатель ветки вперед, на тот же коммит, что и ветка `style`.

При ускоренном слиянии конфликты не возникают. Кроме того, при ускоренном слиянии не создается фиксация слияния.

Задание

```
C:\Users\vladz\work> git switch main
A      README.txt
Switched to branch 'main'

C:\Users\vladz\work> git merge style
Updating 9b759f0..b3cb576
Fast-forward
 style.css | 3 +++
 1 file changed, 3 insertions(+)
 create mode 100644 style.css
```

Решение

02 Создайте клон репозитория `work`

Давайте создадим клон репозитория.

Выполните

```
git clone work home
ls
```

Результат

```
$ git clone work home
Cloning into 'home'...
done.
$ ls
home
work
```

В вашем списке репозиториях теперь должно быть два репозитория: оригинальный репозиторий `work` и клонированный репозиторий `home`.

Задание

```
C:\Users\vladz\work> cd ..

C:\Users\vladz> pwd
'pwd' is not recognized as an internal or external command,
operable program or batch file.

C:\Users\vladz> ls
'ls' is not recognized as an internal or external command,
operable program or batch file.

C:\Users\vladz> git clone work home
Cloning into 'home'...
done.

C:\Users\vladz> ls
'ls' is not recognized as an internal or external command,
operable program or batch file.
```

Решение

32. Что такое origin?

Цели

- Узнать об именах удаленных репозиториях.

Выполните

```
git remote
```

Результат

```
$ git remote
origin
```

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

Задание

```
C:\Users\vladz\home> git remote
origin
```

Решение

Выполните

```
git remote show origin
```

Результат

```
$ git remote show origin
* remote origin
  Fetch URL: /home/alex/ghowto/repositories/work
  Push URL: /home/alex/ghowto/repositories/work
  HEAD branch: main
  Remote branches:
    main tracked
    style tracked
  Local branch configured for 'git pull':
    main merges with remote main
  Local ref configured for 'git push':
    main pushes to main (up to date)
```

Задание

```
C:\Users\vladz\home> git remote show origin
* remote origin
Fetch URL: C:/Users/vladz/work
Push URL: C:/Users/vladz/work
HEAD branch: main
Remote branches:
  main tracked
  style tracked
Local branch configured for 'git pull':
  main merges with remote main
Local ref configured for 'git push':
  main pushes to main (up to date)
```

Решение

Выполните

```
git branch
```

Результат

```
$ git branch
* main
```

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

01 Список удаленных веток

Для того чтобы увидеть все ветки, попробуйте следующую команду:

Выполните

```
git branch -a
```

Результат

```
$ git branch -a
* main
remotes/origin/HEAD -> origin/main
remotes/origin/style
remotes/origin/main
```

Задание

```
C:\Users\vladz\home>git branch
* main

C:\Users\vladz\home>git branch -a
* main
remotes/origin/HEAD -> origin/main
remotes/origin/main
remotes/origin/style
```

Решение

01 Внесите изменения в оригинальный репозиторий `work`

Выполните

```
cd ../work
```

Сейчас мы находимся в репозитории `work`.

Внесите следующие изменения в файл `README`:

Файл: `README`

```
This is the Hello World example from the Git tutorial.  
(changed in origin)
```

Теперь добавьте это изменение и сделайте коммит

Выполните

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

Задание

```
C:\Users\vladz\home>cd ../work  
  
C:\Users\vladz\work>git add README  
fatal: pathspec 'README' did not match any files  
  
C:\Users\vladz\work>git commit -m "Changed README in original repo"  
[main b0b8291] Changed README in original repo  
1 file changed, 1 insertion(+)  
create mode 100644 README.txt
```

Решение

Выполните

```
cd ../home  
git fetch  
git log --all
```

Сейчас мы находимся в репозитории `home`.

Результат

```
$ cd ../home  
$ git fetch  
From /home/alex/ghowto/repositories/work  
 39a1e0f..71df43a  main      -> origin/main  
$ git log --all --graph  
* 71df43a 2023-11-28 | Changed README in original repo (origin/main, origin/HEAD) [Alexander Shvets]  
* 39a1e0f 2023-11-28 | Renamed hello.html; moved style.css (HEAD -> main, origin/style) [Alexander Shvets]  
* 23149b5 2023-11-28 | Included stylesheet into hello.html [Alexander Shvets]  
* b9e6de1 2023-11-28 | Added css stylesheet [Alexander Shvets]  
* 85c14e9 2023-11-28 | Added meta title [Alexander Shvets]  
* ee16740 2023-11-28 | Added README [Alexander Shvets]  
* 9288a33 2023-11-28 | Added copyright statement with email [Alexander Shvets]  
* b7614c1 2023-11-28 | Added HTML header (tag: v1) [Alexander Shvets]  
* 46afaff 2023-11-28 | Added standard HTML page tags (tag: v1-beta) [Alexander Shvets]  
* 78433de 2023-11-28 | Added h1 tag [Alexander Shvets]  
* 5836970 2023-11-28 | Initial commit [Alexander Shvets]
```

Задание

```
C:\Users\vladz\work>cd ../home

C:\Users\vladz\home>git fetch
remote: Enumerating objects: 4, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
Unpacking objects: 100% (3/3), 402 bytes | 57.00 KiB/s, done.
From C:/Users/vladz/work
   b3cb576..b0b8291  main       -> origin/main

C:\Users\vladz\home>git log --all
commit b0b82914b5935e82d5dd435ad450afe9015f32da (origin/main, origin/HEAD)
Author: Your Name <your_email@whatever.com>
Date: 2024-12-10

    Changed README in original repo

commit b3cb576dc02c789e28796258309a0f41fde8e044 (HEAD -> main, origin/style)
Author: Your Name <your_email@whatever.com>
Date: 2024-12-10

    Added css stylesheet
```

Решение

01 Слейте подтянутые изменения в локальную ветку `main`

Выполните

```
git merge origin/main
```

Результат

```
$ git merge origin/main
Updating 39a1e0f..71df43a
Fast-forward
 README | 3 +++
 1 file changed, 2 insertions(+), 1 deletion(-)
```

Задание

```
C:\Users\vladz\home> git merge origin/main
Updating b3cb576..b0b8291
Fast-forward
 README.txt | 1 +
 1 file changed, 1 insertion(+)
 create mode 100644 README.txt
```

Решение

01 Добавьте локальную ветку, которая отслеживает удаленную ветку

Выполните

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

Результат

```
$ git branch --track style origin/style
Branch 'style' set up to track remote branch 'style' from 'origin'.
$ git branch -a
* main
  style
remotes/origin/HEAD -> origin/main
remotes/origin/main
remotes/origin/style
$ git log --all --graph --max-count=2
* 71df43a 2023-11-28 | Changed README in original repo (HEAD -> main, origin/main, origin/HEAD) [Alexander Shvets]
* 39a1e0f 2023-11-28 | Renamed hello.html; moved style.css (origin/style, style) [Alexander Shvets]
```

Задание

```

C:\Users\vladz\home> git branch --track style origin/style
branch 'style' set up to track 'origin/style'.

C:\Users\vladz\home> git branch -a
* main
  style
  remotes/origin/HEAD -> origin/main
  remotes/origin/main
  remotes/origin/style

C:\Users\vladz\home> git log --max-count=2
commit b0b82914b5935e82d5dd435ad450afe9015f32da (HEAD -> main, origin/main, origin/HEAD)
Author: Your Name <your_email@whatever.com>
Date: 2024-12-10

    Changed README in original repo

commit b3cb576dc02c789e28796258309a0f41fde8e044 (origin/style, style)
Author: Your Name <your_email@whatever.com>
Date: 2024-12-10

    Added css stylesheet

```

Решение

01 Создайте чистый репозиторий

Выполните

```

cd ..
git clone --bare work work.git
ls work.git

```

Сейчас мы находимся в директории `repositories`.

Результат

```

$ git clone --bare work work.git
Cloning into bare repository 'work.git'...
done.
$ ls work.git
branches
config
description
HEAD
hooks
info
objects
packed-refs
refs

```

Задание

```

C:\Users\vladz\home> cd ..

C:\Users\vladz> git clone --bare work work.git
Cloning into bare repository 'work.git'...
done.

C:\Users\vladz> ls work.git
'ls' is not recognized as an internal or external command,
operable program or batch file.

```

Решение

Файл: README

This is the Hello World example from the Git tutorial.
(changed in the origin and pushed to shared)

Выполните

```
git switch main
git add README
git commit -m "Added shared comment to readme"
```

Теперь отправьте изменения в общий репозиторий.

Выполните

```
git push shared main
```

Задание

```
C:\Users\vladz\work> git switch main
M       README.txt
Already on 'main'

C:\Users\vladz\work> git add README
fatal: pathspec 'README' did not match any files

C:\Users\vladz\work> git commit -m "Added shared comment to readme"
On branch main
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:   README.txt

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

C:\Users\vladz\work> git push shared main
Everything up-to-date
```

Решение

Выполните

```
cd ../home
```

Сейчас мы находимся в репозитории `home`.

Продолжите с...

Выполните

```
git remote add shared ../work.git
git branch --track shared main
git pull shared main
cat README
```

Задание

```

C:\Users\vladz\work> cd ../home

C:\Users\vladz\home> git remote add shared ../work.git

C:\Users\vladz\home> git branch --track shared main
branch 'shared' set up to track 'main'.

C:\Users\vladz\home> git pull shared main
From ../work
 * branch          main          -> FETCH_HEAD
 * [new branch]    main          -> shared/main
Already up to date.

C:\Users\vladz\home> cat README
'cat' is not recognized as an internal or external command,
operable program or batch file.

```

Решение

01 Запуск Git-сервера

Выполните

```

# (From the "repositories" directory)
git daemon --verbose --export-all --base-path=.

```

Задание

```

C:\Users\vladz\home> # (From the "repositories" directory)
'#' is not recognized as an internal or external command,
operable program or batch file.

C:\Users\vladz\home> git daemon --verbose --export-all --base-path=.
[11956] Ready to rumble

```

Решение