

Использование системы контроля версий Git

Для выполнения представленных действий использовалась командная строка (git bash имеет более приятный интерфейс, однако все действия можно также выполнить и через обычную командную строку).

```
MINGW64/c/repoTest

Asus TUF@DESKTOP-C501IB0 MINGW64 /c/repoTest (master)
$ git status
On branch master
Your branch is up to date with 'origin/master'.

nothing to commit, working tree clean

Asus TUF@DESKTOP-C501IB0 MINGW64 /c/repoTest (master)
$
```

```
C:\Windows\system32\cmd.exe

C:\RepoTest>git status
On branch master
Your branch is up to date with 'origin/master'.

nothing to commit, working tree clean

C:\RepoTest>
```

Прежде чем начать работать непосредственно с Git, необходимо выбрать каталог, в котором будет находиться наш репозиторий.

В моём случае это каталог RepoTest, в который я перехожу командой `cd C:/RepoTest`.

```
MINGW64/c/RepoTest

Asus TUF@DESKTOP-C501IB0 MINGW64 ~ (master)
$ cd C:/RepoTest

Asus TUF@DESKTOP-C501IB0 MINGW64 /c/RepoTest
$
```

После выбора каталога инициализируем в нём git, с помощью команды `git init`.

```
Asus TUF@DESKTOP-C501IB0 MINGW64 /c/RepoTest
$ git init
Initialized empty Git repository in C:/RepoTest/.git/

Asus TUF@DESKTOP-C501IB0 MINGW64 /c/RepoTest (master)
$
```

Теперь удостоверимся, что инициализация удалась с помощью команды `ls -la`

```
Asus TUF@DESKTOP-C501IB0 MINGW64 /c/RepoTest (master)
$ ls -la
total 12
drwxr-xr-x 1 Asus TUF 197121 0 сен 25 00:28 ./
drwxr-xr-x 1 Asus TUF 197121 0 сен 25 00:25 ../
drwxr-xr-x 1 Asus TUF 197121 0 сен 25 00:28 .git/

Asus TUF@DESKTOP-C501IB0 MINGW64 /c/RepoTest (master)
$
```

Стецук Максим 2гр.1п.гр.

Проверим статус репозитория с помощью команды *git status*. Консоль вывела нам, что мы находимся в ветви master и закоммиченные файлы отсутствуют.

```
Asus TUF@DESKTOP-C5011B0 MINGW64 /c/RepoTest (master)
$ git status
On branch master

No commits yet

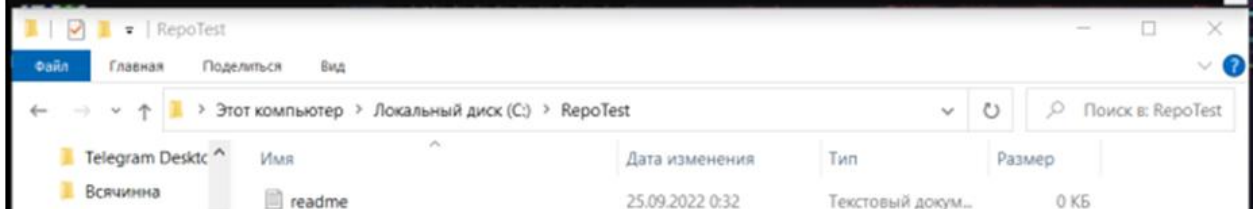
nothing to commit (create/copy files and use "git add" to track)

Asus TUF@DESKTOP-C5011B0 MINGW64 /c/RepoTest (master)
$
```

Создадим текстовый файл readme с помощью команды *touch readme.txt*.

```
Asus TUF@DESKTOP-C5011B0 MINGW64 /c/RepoTest (master)
$ touch readme.txt

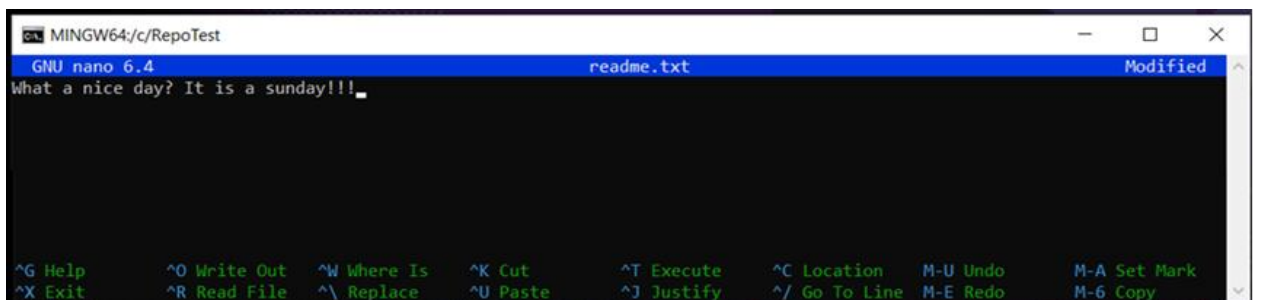
Asus TUF@DESKTOP-C5011B0 MINGW64 /c/RepoTest (master)
$
```



Для добавления или изменения текста в нашем новом файле воспользуемся командой *nano readme.txt*

```
Asus TUF@DESKTOP-C5011B0 MINGW64 /c/RepoTest (master)
$ nano readme.txt
```

Нам откроется окно, в котором мы сможем набирать новый и изменять уже имеющийся текст. Для сохранения изменений можно использовать сочетание клавиш *ctrl+s*, а для завершения редактирования сочетание клавиш *ctrl+x*.



Проверим статус git. Мы видим, что появился новый неотслеживаемый файл readme.txt.

```
Asus TUF@DESKTOP-C5011B0 MINGW64 /c/RepoTest (master)
$ git status
On branch master

No commits yet

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)

Asus TUF@DESKTOP-C5011B0 MINGW64 /c/RepoTest (master)
$
```

Стецук Максим 2гр.1п.гр.

Теперь нам необходимо начать отслеживать этот файл, для этого используем команду `git add readme.txt`, которая добавит данный файл непосредственно в git. Проверив статус git, видим, что новый файл появился и может быть закоммичен.

```
Asus TUF@DESKTOP-C501IB0 MINGW64 /c/RepoTest (master)
$ git add readme.txt
warning: in the working copy of 'readme.txt', LF will be replaced by CRLF the next time Git touches it

Asus TUF@DESKTOP-C501IB0 MINGW64 /c/RepoTest (master)
$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
    new file:   readme.txt

Asus TUF@DESKTOP-C501IB0 MINGW64 /c/RepoTest (master)
$
```

Теперь зафиксируем (закоммитим) данный файл с помощью команды `git commit -m "First file"`, где в кавычках написан комментарий о том, что именно мы коммитим. При проверке статуса git мы увидим, что файлов для фиксирования больше нет.

```
Asus TUF@DESKTOP-C501IB0 MINGW64 /c/RepoTest (master)
$ git commit -m "First file"
[master (root-commit) 5663789] First file
 1 file changed, 1 insertion(+)
 create mode 100644 readme.txt

Asus TUF@DESKTOP-C501IB0 MINGW64 /c/RepoTest (master)
$ git status
On branch master
nothing to commit, working tree clean

Asus TUF@DESKTOP-C501IB0 MINGW64 /c/RepoTest (master)
$
```

Изменим текст в файле `readme.txt` и проверим статус. Мы видим, что файл изменился и его необходимо снова добавить.

```
Asus TUF@DESKTOP-C501IB0 MINGW64 /c/RepoTest (master)
$ nano readme.txt

Asus TUF@DESKTOP-C501IB0 MINGW64 /c/RepoTest (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:   readme.txt

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

Asus TUF@DESKTOP-C501IB0 MINGW64 /c/RepoTest (master)
$
```

Снова добавляем его с помощью команды `git add`.

```
Asus TUF@DESKTOP-C501IB0 MINGW64 /c/RepoTest (master)
$ git add readme.txt
warning: in the working copy of 'readme.txt', LF will be replaced by CRLF the next time Git touches it

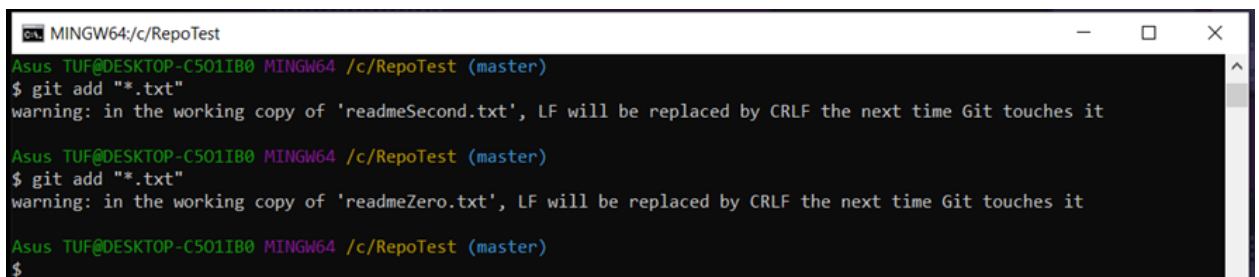
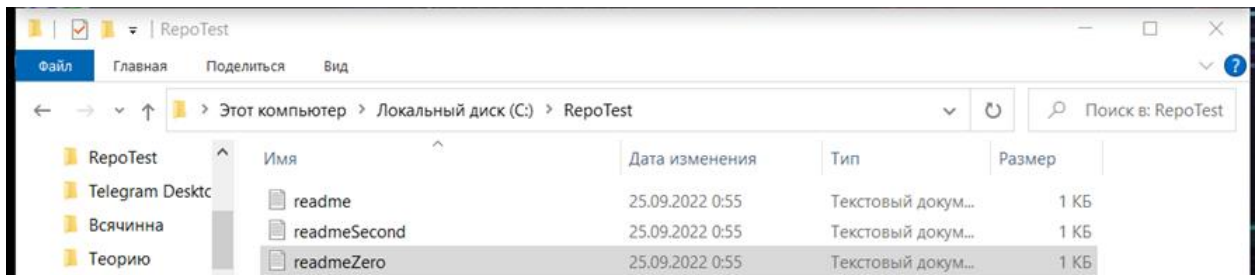
Asus TUF@DESKTOP-C501IB0 MINGW64 /c/RepoTest (master)
$
```

Теперь сделаем 2 копии файла `readme.txt`, изменив их названия. В моём случае это будут файлы `readmeSecond.txt` и `readmeZero.txt`. Данные файлы

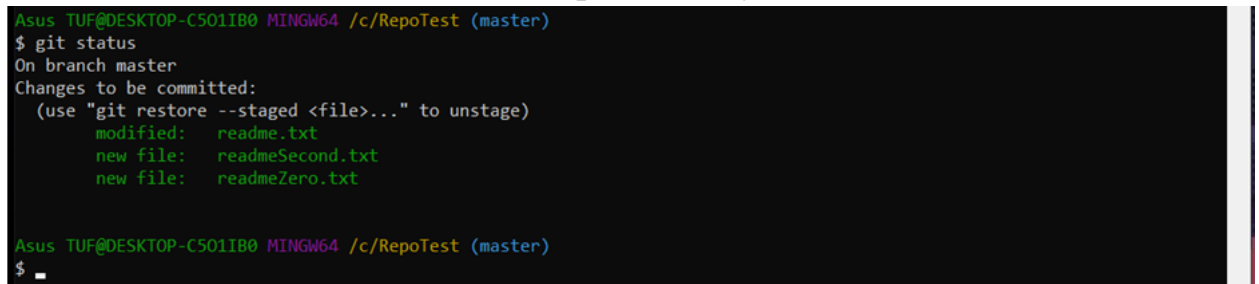
Стецук Максим 2гр.1п.гр.

также необходимо добавить в наш git. Также как и с файлом `readme.txt`, мы можем их добавить по очереди с помощью простого `git add` имя файла, но можно использовать эту команду с маской, которая укажет на все файлы типа `.txt`, а именно `git add ".txt"`, где `".txt"` является маской.

(я немного ошибся и сначала сделал одну копию файла, из-за чего данную команду я использовал дважды)



Теперь, при проверке статуса, нам высветится, что один файл был изменён, а ещё два было добавлено. И все эти файлы могут быть закоммичены.



Зафиксируем их. Добавим комментарий, что мы фиксируем все добавленные файлы.



Для просмотра истории коммитов используется команда `git log`. После её использования нам выводится список, в котором указано, что именно мы сделали (наш комментарий), а также время, когда мы это сделали. Важно отметить, что изменения в списке расположены от более нового к более старому.

Стецук Максим 2гр.1п.гр.

```
Asus TUF@DESKTOP-C501I80 MINGW64 /c/RepoTest (master)
$ git log
commit a59658891bc1afe4cbd8a71cde8dc1dd7940cee0 (HEAD -> master)
Author: XtulenchikX <TulenchikPlay@yandex.ru>
Date:   Sun Sep 25 01:06:32 2022 +0300

    All files were added

commit 5663789a9de4abd59632ea80ab69a4247a104fd1
Author: XtulenchikX <TulenchikPlay@yandex.ru>
Date:   Sun Sep 25 00:51:50 2022 +0300

    First file

Asus TUF@DESKTOP-C501I80 MINGW64 /c/RepoTest (master)
$
```

Для получения более подробной информации о каждом из коммитов используется команда `git log --summary`.

```
Asus TUF@DESKTOP-C501I80 MINGW64 /c/RepoTest (master)
$ git log --summary
commit a59658891bc1afe4cbd8a71cde8dc1dd7940cee0 (HEAD -> master)
Author: XtulenchikX <TulenchikPlay@yandex.ru>
Date:   Sun Sep 25 01:06:32 2022 +0300

    All files were added

    create mode 100644 readmeSecond.txt
    create mode 100644 readmeZero.txt

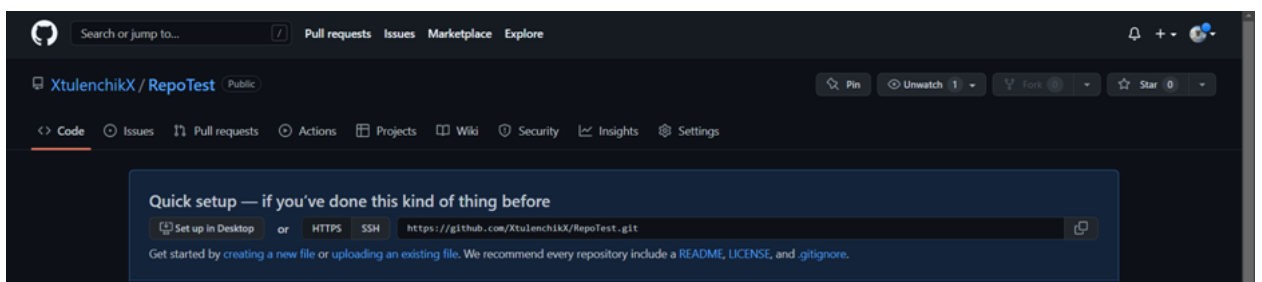
commit 5663789a9de4abd59632ea80ab69a4247a104fd1
Author: XtulenchikX <TulenchikPlay@yandex.ru>
Date:   Sun Sep 25 00:51:50 2022 +0300

    First file

    create mode 100644 readme.txt

Asus TUF@DESKTOP-C501I80 MINGW64 /c/RepoTest (master)
$
```

Для выгрузки в удалённый репозиторий необходимо выполнить несколько действий. Для начала необходимо создать пустой репозиторий и получить его URL адрес.



Теперь добавим наш удалённый репозиторий. Для этого воспользуемся командой `git remote add origin URL адрес`, в моём случае команда будет такой: `git remote add origin https://github.com/XtulenchikX/RepoTest`

```
Asus TUF@DESKTOP-C501I80 MINGW64 /c/RepoTest (master)
$ git remote add origin https://github.com/XtulenchikX/RepoTest

Asus TUF@DESKTOP-C501I80 MINGW64 /c/RepoTest (master)
$
```

Переходим к непосредственной выгрузке файлов в удалённый репозиторий. Для этого используем команду `git push -u origin master`, где master это ветка с которой мы работаем. После применения данной команды появляется

Стецук Максим 2гр.1п.гр.

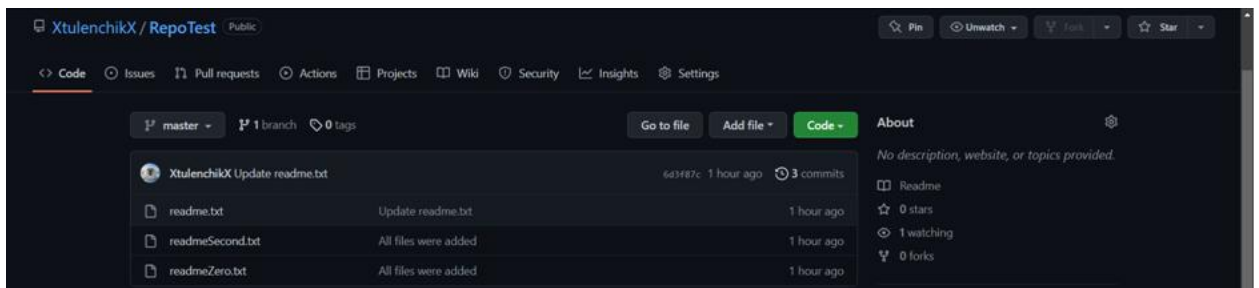
возможность авторизации через github (автоматически всплывает окно и открывается браузерная версия github).

После авторизации происходит выгрузка файлов, о которой выводится подробная информация.

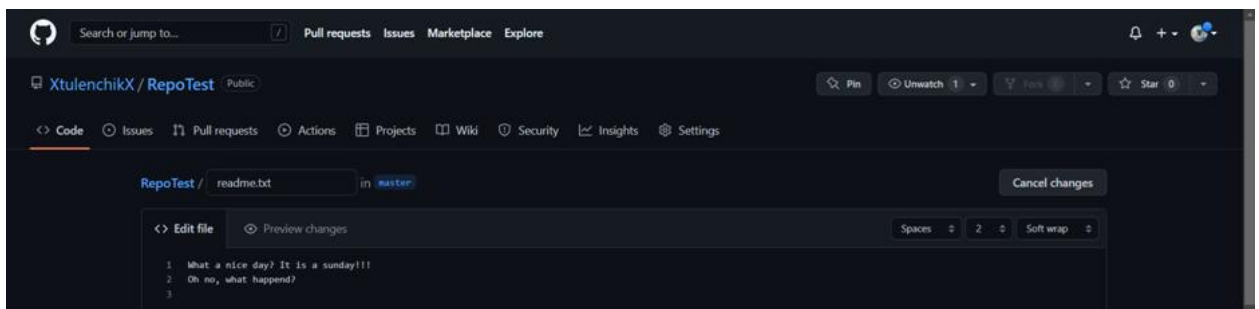
```
Asus TUF@DESKTOP-C501IB0 MINGW64 /c/RepoTest (master)
$ git push -u origin master
info: please complete authentication in your browser...
Enumerating objects: 6, done.
Counting objects: 100% (6/6), done.
Delta compression using up to 12 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (6/6), 544 bytes | 544.00 KiB/s, done.
Total 6 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/XtulenchikX/RepoTest
 * [new branch]      master -> master
branch 'master' set up to track 'origin/master'.

Asus TUF@DESKTOP-C501IB0 MINGW64 /c/RepoTest (master)
$
```

Файлы появляются в репозитории в GitHub.



Как получить все файлы обратно? Для того чтобы увидеть наглядно изменения, отредактируем файл readme.txt.



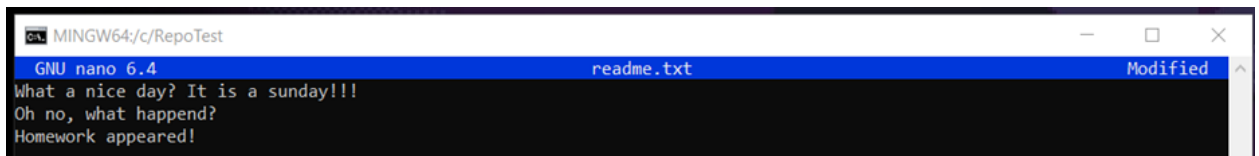
Теперь в консоли прописываем команду *git pull origin master*, которая возвращает файлы из репозитория. После использования данной команды выводится информация о том, какие конкретно файлы были изменены и что в них поменялось.

```
Asus TUF@DESKTOP-C501IB0 MINGW64 /c/RepoTest (master)
$ git pull origin master
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), 720 bytes | 120.00 KiB/s, done.
From https://github.com/XtulenchikX/RepoTest
 * branch            master       -> FETCH_HEAD
  a596588..6d3f87c  master       -> origin/master
Updating a596588..6d3f87c
Fast-forward
 readme.txt | 2 +-
 1 file changed, 1 insertion(+), 1 deletion(-)

Asus TUF@DESKTOP-C501IB0 MINGW64 /c/RepoTest (master)
$
```

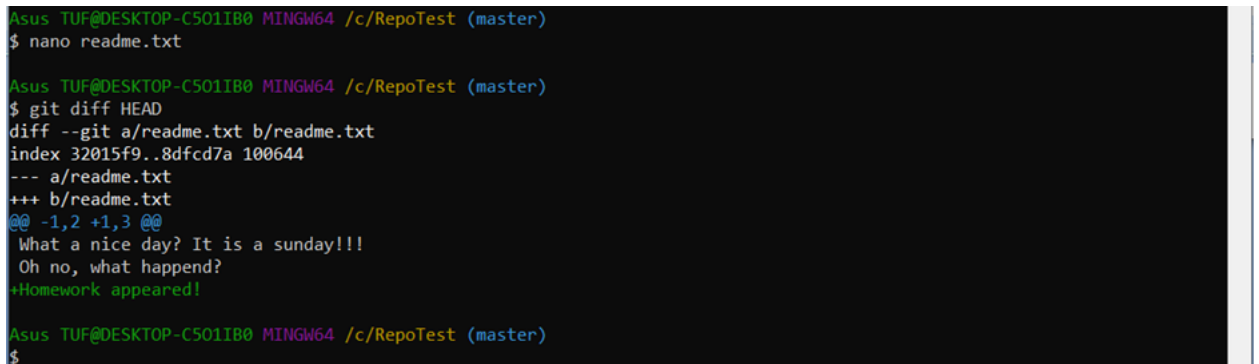
Стецук Максим 2гр.1п.гр.

Снова меняем файл `readme.txt` с помощью команды `nano readme.txt`.



```
MINGW64/c/RepoTest
GNU nano 6.4      readme.txt      Modified
What a nice day? It is a sunday!!!
Oh no, what happend?
Homework appeared!
```

Для проверки различий между закоммиченной версией (та версия файлов, которую мы получили) и той версией с которой мы провели изменения используется команда `git diff HEAD`, где `HEAD` выполняет роль указателя на зафиксированную (закоммиченную) версию.



```
Asus TUF@DESKTOP-C501IB0 MINGW64 /c/RepoTest (master)
$ nano readme.txt

Asus TUF@DESKTOP-C501IB0 MINGW64 /c/RepoTest (master)
$ git diff HEAD
diff --git a/readme.txt b/readme.txt
index 32015f9..8dfcd7a 100644
--- a/readme.txt
+++ b/readme.txt
@@ -1,2 +1,3 @@
 What a nice day? It is a sunday!!!
 Oh no, what happend?
+Homework appeared!

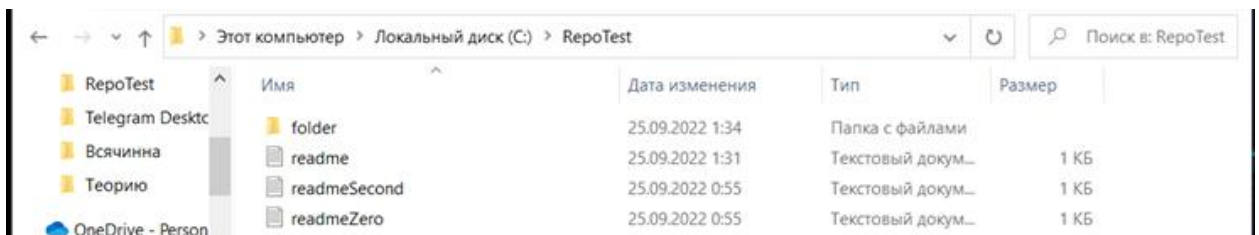
Asus TUF@DESKTOP-C501IB0 MINGW64 /c/RepoTest (master)
$
```

Теперь добавим каталог в наш `git`, для этого напишем команду `mkdir folder`, где `folder` является названием для нашего каталога. После применения данной команды, в каталоге `RepoTest`, появился каталог `folder`.



```
Asus TUF@DESKTOP-C501IB0 MINGW64 /c/RepoTest (master)
$ mkdir folder

Asus TUF@DESKTOP-C501IB0 MINGW64 /c/RepoTest (master)
$
```

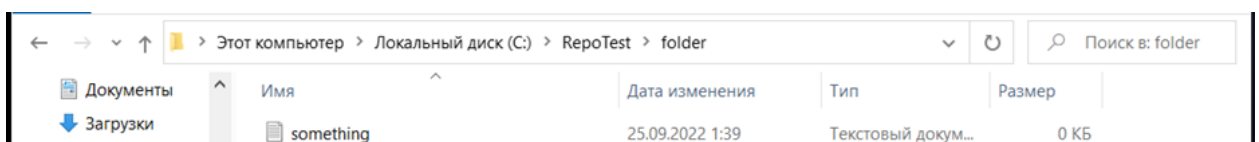


С помощью команды `touch folder/something.txt` создадим файл в каталоге, в данном случае `folder/something.txt` указывает на то, что текстовый файл с названием `something` будет создан в каталоге `folder`.



```
Asus TUF@DESKTOP-C501IB0 MINGW64 /c/RepoTest (master)
$ touch folder/something.txt

Asus TUF@DESKTOP-C501IB0 MINGW64 /c/RepoTest (master)
$
```



Стецук Максим 2гр.1п.гр.

При проверке статуса мы видим сообщение о том, что появился новый каталог с названием `folder`, но он не отслеживается.

```
Asus TUF@DESKTOP-C501IB0 MINGW64 /c/RepoTest (master)
$ git status
On branch master
Your branch is up to date with 'origin/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:   readme.txt

Untracked files:
  (use "git add <file>..." to include in what will be committed)
    folder/

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

Asus TUF@DESKTOP-C501IB0 MINGW64 /c/RepoTest (master)
$
```

С помощью команды `git add` добавим каталог и файл в нём в отслеживаемые файлы. В моём случае полная команда будет выглядеть так: `git add folder/something.txt`. Если посмотреть `git status`, то мы увидим, что наш каталог и файл успешно добавлены и могут быть закоммичены.

```
Asus TUF@DESKTOP-C501IB0 MINGW64 /c/RepoTest (master)
$ git add folder/something.txt

Asus TUF@DESKTOP-C501IB0 MINGW64 /c/RepoTest (master)
$ git status
On branch master
Your branch is up to date with 'origin/master'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    new file:   folder/something.txt

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

Asus TUF@DESKTOP-C501IB0 MINGW64 /c/RepoTest (master)
$
```

С помощью команды `git diff --staged` мы можем посмотреть изменения, которые были сделаны в стадии `staged`.

```
Asus TUF@DESKTOP-C501IB0 MINGW64 /c/RepoTest (master)
$ git diff --staged
diff --git a/folder/something.txt b/folder/something.txt
new file mode 100644
index 0000000..e69de29

Asus TUF@DESKTOP-C501IB0 MINGW64 /c/RepoTest (master)
$
```

Для отмены данных изменений используется команда `git reset`, в моём случае вся команда это `git reset folder/something.txt`. После этого, снова применив команду `git diff --staged` мы увидим, что изменений нет.

```
Asus TUF@DESKTOP-C501IB0 MINGW64 /c/RepoTest (master)
$ git reset folder/something.txt
Unstaged changes after reset:
M   readme.txt

Asus TUF@DESKTOP-C501IB0 MINGW64 /c/RepoTest (master)
$ git diff --staged

Asus TUF@DESKTOP-C501IB0 MINGW64 /c/RepoTest (master)
$
```


Стецук Максим 2гр.1п.гр.

Теперь вернём файл `readme.txt` в исходно состояние. Для этого используем команду `git checkout -- readme.txt`. А также посмотрим статус нашего `git`.

```
Asus TUF@DESKTOP-C501IB0 MINGW64 /c/RepoTest (master)
$ git checkout -- readme.txt

Asus TUF@DESKTOP-C501IB0 MINGW64 /c/RepoTest (master)
$ git status
On branch master
Your branch is up to date with 'origin/master'.

Untracked files:
  (use "git add <file>..." to include in what will be committed)
    folder/

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

Asus TUF@DESKTOP-C501IB0 MINGW64 /c/RepoTest (master)
$
```

Чтобы убедиться, что файл `readme.txt` вернулся в исходное состояние с помощью команды `cat readme.txt` выведем текст этого файла в консоли.

```
Asus TUF@DESKTOP-C501IB0 MINGW64 /c/RepoTest (master)
$ cat readme.txt
What a nice day? It is a sunday!!!
Oh no, what happend?

Asus TUF@DESKTOP-C501IB0 MINGW64 /c/RepoTest (master)
$
```

Теперь создадим отдельную ветку, в которой будем изменять наши файлы. Она будет служить для очистки. Создаётся ветка командой `git branch clean`, где вместо `clean` может быть любое другое название. А также с помощью просто `git branch` посмотрим, в какой ветке мы сейчас работаем.

```
Asus TUF@DESKTOP-C501IB0 MINGW64 /c/RepoTest (master)
$ git branch clean

Asus TUF@DESKTOP-C501IB0 MINGW64 /c/RepoTest (master)
$ git branch
  clean
* master

Asus TUF@DESKTOP-C501IB0 MINGW64 /c/RepoTest (master)
$
```

Используем команду `git checkout clean`, чтобы смениться на ветку `clean`. И снова посмотрим, в какой ветке мы находимся.

```
Asus TUF@DESKTOP-C501IB0 MINGW64 /c/RepoTest (master)
$ git checkout clean
Switched to branch 'clean'

Asus TUF@DESKTOP-C501IB0 MINGW64 /c/RepoTest (clean)
$ git branch
* clean
  master

Asus TUF@DESKTOP-C501IB0 MINGW64 /c/RepoTest (clean)
$
```

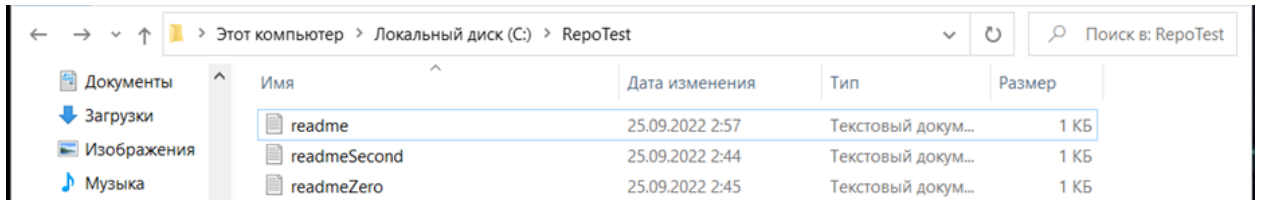
С помощью команды `rm -r folder` удалим наш каталог в данной ветке.

```
Asus TUF@DESKTOP-C501IB0 MINGW64 /c/RepoTest (clean)
$ rm -r folder

Asus TUF@DESKTOP-C501IB0 MINGW64 /c/RepoTest (clean)
$ git status
On branch clean
nothing to commit, working tree clean

Asus TUF@DESKTOP-C501IB0 MINGW64 /c/RepoTest (clean)
$
```

Стецук Максим 2гр.1п.гр.



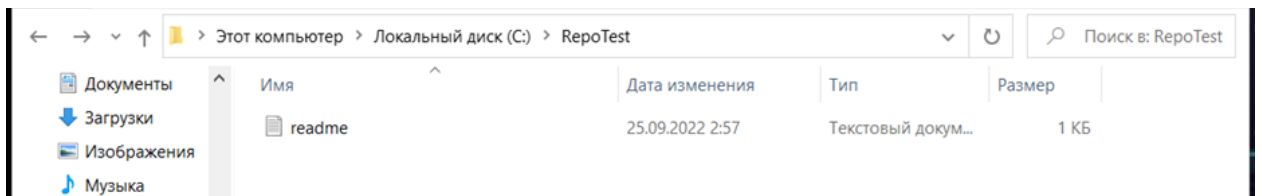
Теперь удалим все файлы кроме readme.txt с помощью команды `git rm имя файла`. Я использовал команды `git rm readmeSecond.txt` и `git rm readmeZero.txt`. При проверке статуса будет показано, что были удалены два файла.

```
Asus TUF@DESKTOP-C501IB0 MINGW64 /c/RepoTest (clean)
$ git rm readmeZero.txt
rm 'readmeZero.txt'

Asus TUF@DESKTOP-C501IB0 MINGW64 /c/RepoTest (clean)
$ git rm readmeSecond.txt
rm 'readmeSecond.txt'

Asus TUF@DESKTOP-C501IB0 MINGW64 /c/RepoTest (clean)
$ git status
On branch clean
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        deleted:   readmeSecond.txt
        deleted:   readmeZero.txt

Asus TUF@DESKTOP-C501IB0 MINGW64 /c/RepoTest (clean)
$
```



С помощью команды `git commit`, зафиксируем данные изменения, чтобы дальше выполнить слияние ветвей.

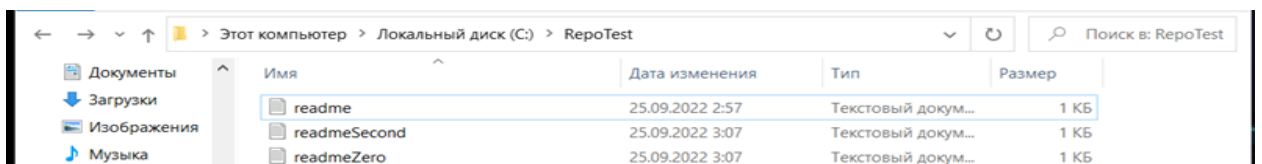
```
Asus TUF@DESKTOP-C501IB0 MINGW64 /c/RepoTest (clean)
$ git commit -m "Deleted files and folder"
[clean e7fe584] Deleted files and folder
2 files changed, 4 deletions(-)
delete mode 100644 readmeSecond.txt
delete mode 100644 readmeZero.txt

Asus TUF@DESKTOP-C501IB0 MINGW64 /c/RepoTest (clean)
$
```

Теперь возвращаемся в ветку master. Заметим, что в ветке master, удалённые файлы до сих пор существуют.

```
Asus TUF@DESKTOP-C501IB0 MINGW64 /c/RepoTest (clean)
$ git checkout master
Switched to branch 'master'
Your branch is up to date with 'origin/master'.

Asus TUF@DESKTOP-C501IB0 MINGW64 /c/RepoTest (master)
$
```

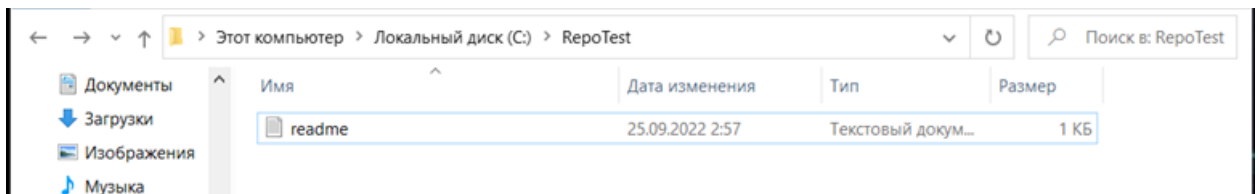


Стецук Максим 2гр.1п.гр.

Теперь проведём слияние веток, чтобы удалённые файлы в ветке для очистки, также исчезли в ветке master. Для этого используем команду `git merge clean`, где вместо `clean` может быть использовано имя любой другой созданной ветви, с которой мы хотим слить данную ветвь. После выполнения данной команды, все файлы кроме `readme.txt` будут удалены.

```
Asus TUF@DESKTOP-C501I80 MINGW64 /c/RepoTest (master)
$ git merge clean
Updating 6d3f87c..e7fe584
Fast-forward
 readmeSecond.txt | 2 --
 readmeZero.txt   | 2 --
 2 files changed, 4 deletions(-)
 delete mode 100644 readmeSecond.txt
 delete mode 100644 readmeZero.txt

Asus TUF@DESKTOP-C501I80 MINGW64 /c/RepoTest (master)
$
```



Так как ветвь `clean` нам больше не нужна (она создавалась только для удаления файлов), удалим её с помощью команды `git branch -d clean`.

```
Asus TUF@DESKTOP-C501I80 MINGW64 /c/RepoTest (master)
$ git branch -d clean
Deleted branch clean (was e7fe584).

Asus TUF@DESKTOP-C501I80 MINGW64 /c/RepoTest (master)
$
```

С помощью `git push` выгружаем изменения в удалённый репозиторий.

```
Asus TUF@DESKTOP-C501I80 MINGW64 /c/RepoTest (master)
$ git push
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Delta compression using up to 12 threads
Compressing objects: 100% (1/1), done.
Writing objects: 100% (2/2), 252 bytes | 252.00 KiB/s, done.
Total 2 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/XtulenchikX/RepoTest
 6d3f87c..e7fe584 master -> master

Asus TUF@DESKTOP-C501I80 MINGW64 /c/RepoTest (master)
$
```

Если обновить страницу GitHub, то мы увидим, что теперь в репозитории только один файл, а именно `readme.txt`, который мы оставили при удалении остальных.

