



**МИНОБРНАУКИ РОССИИ**  
**федеральное государственное бюджетное образовательное учреждение**  
**высшего образования**  
**«Московский государственный технологический университет «СТАНКИН»**  
**(ФГБОУ ВО «МГТУ «СТАНКИН»)**

---

Институт цифровых  
интеллектуальных систем

Кафедра  
компьютерных систем управления

Дисциплина «Основы системного программного обеспечения»

**Отчет по лабораторной работе №1**  
**«Работа с системами контроля версий на примере Git Hub»**

**Выполнил**  
**студент гр. АДБ-21-07**

\_\_\_\_\_

(дата)

\_\_\_\_\_

(подпись)

**Маслова Е.П.**

**Проверил**  
**к.т.н., доцент**

\_\_\_\_\_

(дата)

\_\_\_\_\_

(подпись)

**Ковалев И.А.**

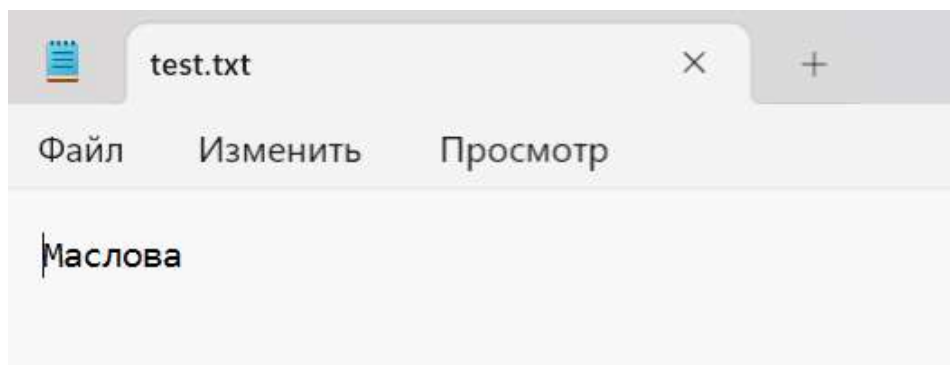
**Москва 2024 г.**

**Цель работы:** Познакомиться с принципом работы систем контроля версий на примере Git Hub

### Индивидуальное задание

#### 1. Создание локального репозитория

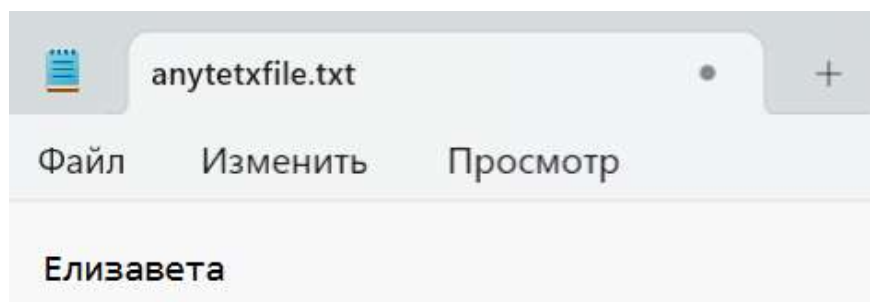
Создадим на диске папку со своей фамилией в «Документы», чтобы было меньше конфликтов с правом доступа. Создадим в ней файл test.txt и напишем в нем свою фамилию.



Перейдем в созданную нами папку в командной строке, используя команду cd. Проинициализируем эту папку как git репозиторий с помощью команды git init

```
PS C:\Users\maslo> cd C:\Users\maslo\Documents\LR1_OSP0
PS C:\Users\maslo\Documents\LR1_OSP0> git init
Initialized empty Git repository in C:/Users/maslo/Documents/LR1_OSP0/.git/
```

Теперь, используя проводник windows, создав в своей папке любой текстовый файл и пропишем в него имя.



Проверим состояние репозитория с помощью команды git status:

```
PS C:\Users\maslo\Documents\LR1_OSP0> git status
On branch master

No commits yet

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

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

В нашем случае файл новый и система еще не знает, нужно ли следить за изменениями в файле или его можно просто игнорировать. Для того, чтобы начать отслеживать новый файл, нужно его специальным образом объявить.

## 2. Фиксация изменений в области заготовленных файлов

```
PS C:\Users\maslo\Documents\LR1_OSP0> git add .
PS C:\Users\maslo\Documents\LR1_OSP0> git status
On branch master

No commits yet

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

Теперь закомитим наши файлы. Коммит представляет собой состояние репозитория в определенный момент времени как некий опечаток во времени. Коммит отмечается hash суммой и к которому мы можем в любой момент времени вернуться.

```
PS C:\Users\maslo\Documents\LR1_OSP0> git commit -m "first commit"
[master (root-commit) fb5ae2b] first commit
2 files changed, 2 insertions(+)
create mode 100644 anytetxfile.txt
create mode 100644 test.txt
```

## 3. Отправка коминат на сервер

Чтобы связать наш локальный репозиторий с репозиторием на GitHub, выполним следующую команду в терминале.

git remote add origin [https://github.com/Ilizmas/LR1\\_OSP0](https://github.com/Ilizmas/LR1_OSP0)

Можно вызвать команду, для просмотра, к какому проекту мы подключены

```
PS C:\Users\maslo\Documents\LR1_OSP0> git remote add origin https://github.com/Ilizmas/LR1_OSP0
PS C:\Users\maslo\Documents\LR1_OSP0> git remote -v
origin https://github.com/Ilizmas/LR1_OSP0 (fetch)
origin https://github.com/Ilizmas/LR1_OSP0 (push)
```

Если вдруг у нас на этом месте показывается несколько репозиториев, то скорее всего верхние команды привели к запутыванию веток, это можно разрешить следующей командой:

```
PS C:\Users\maslo\Documents\LR1_OSP0> git pull origin master --allow-unrelated-histories
From https://github.com/Ilizmas/LR1_OSP0
 * branch          master      -> FETCH_HEAD
 * [new branch]     master      -> origin/master
Already up to date.
```

#### 4. Запросим изменения с сервера

Если мы сделаем какие-то изменения, то другие пользователи могут скачать их с помощью этой команды:

```
PS C:\Users\maslo\Documents\LR1_OSP0> git pull origin master
From https://github.com/Ilizmas/LR1_OSP0
 * branch          master      -> FETCH_HEAD
Already up to date.
```

#### 5. Теперь перешлем локальный коммит на сервер

Теперь отправим коммит на сервер, команда, предназначенная для этого — push. Она принимает два параметра: имя удаленного репозитория (мы назвали наш origin) и ветку, в которую необходимо внести изменения (master — это ветка по умолчанию для всех репозиториев).

```
PS C:\Users\maslo\Documents\LR1_OSP0> git push origin master
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 16 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (6/6), 605 bytes | 121.00 KiB/s, done.
Total 6 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/Ilizmas/LR1_OSP0
 801433d..373522a master -> master
```

Посмотрим наши изменения с использованием команды git log:

```

PS C:\Users\maslo\Documents\LR1_OSP0> git log
commit 373522a92a247b293ad1559102e87870e499dc81 (HEAD -> master, origin/master)
Merge: fb5ae2b 801433d
Author: Ilizmas <maslovaelizaveta1916@yandex.ru>
Date: Mon May 6 19:28:23 2024 +0300

    first commit

commit fb5ae2bf8baf7c82c2e40983eac7f1bc0ada607f
Author: Ilizmas <maslovaelizaveta1916@yandex.ru>
Date: Mon May 6 19:21:26 2024 +0300

    first commit

commit 801433dea5ee533a929eafa46ce54162a1394e8b
Author: Ilizmas <128056811+Ilizmas@users.noreply.github.com>
Date: Mon May 6 19:14:32 2024 +0300

    Initial commit

```

## 6. Создание новой ветки

Во время разработки новой функциональности считается хорошей практикой работать с копией оригинального проекта, которую называют веткой.

Создадим новую ветку:

```

PS C:\Users\maslo\Documents\LR1_OSP0> git branch second
PS C:\Users\maslo\Documents\LR1_OSP0> git branch
* master
  second

```

Переключимся на другую ветку, используя команду `git checkout` «название ветки». Создадим новый файл в нашем локальном репозитории и напишем в нем свою фамилию, добавим в область подготовленных файлов, закоммитим и отправим на сервер.

```

PS C:\Users\maslo\Documents\LR1_OSP0> git checkout second
Switched to branch 'second'
PS C:\Users\maslo\Documents\LR1_OSP0> git status
On branch second
Untracked files:
  (use "git add <file>..." to include in what will be committed)
    newfile.txt

nothing added to commit but untracked files present (use "git add" to track)
PS C:\Users\maslo\Documents\LR1_OSP0> git add newfile.txt
PS C:\Users\maslo\Documents\LR1_OSP0> git status
On branch second
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    new file:   newfile.txt

```

```

PS C:\Users\maslo\Documents\LR1_OSP0> git commit -m "add newnewfile, try add"
[second 0c496fb] add newnewfile, try add
1 file changed, 1 insertion(+)
create mode 100644 newnewfile.txt
PS C:\Users\maslo\Documents\LR1_OSP0> git push origin second
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 16 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 506 bytes | 101.00 KiB/s, done.
Total 4 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (1/1), done.
remote:
remote: Create a pull request for 'second' on GitHub by visiting:
remote:   https://github.com/Ilizmas/LR1_OSP0/pull/new/second
remote:
To https://github.com/Ilizmas/LR1_OSP0
 * [new branch]      second -> second

```

## 7. Слияние веток

Переключимся на ветку master

```

PS C:\Users\maslo\Documents\LR1_OSP0> git checkout master
Switched to branch 'master'

```

```

PS C:\Users\maslo\Documents\LR1_OSP0> git branch
* master
second

```

Проведем слияние двух веток через с помощью команды merge:

```

PS C:\Users\maslo\Documents\LR1_OSP0> git merge second
Updating 373522a..0c496fb
Fast-forward
 newfile.txt      | 1 +
 newnewfile.txt   | 1 +
 2 files changed, 2 insertions(+)
 create mode 100644 newfile.txt
 create mode 100644 newnewfile.txt

```

Все прошло успешно, можно удалить ветку second:

```

PS C:\Users\maslo\Documents\LR1_OSP0> git branch -d second
Deleted branch second (was 0c496fb).

```

```

PS C:\Users\maslo\Documents\LR1_OSP0> git branch
* master

```

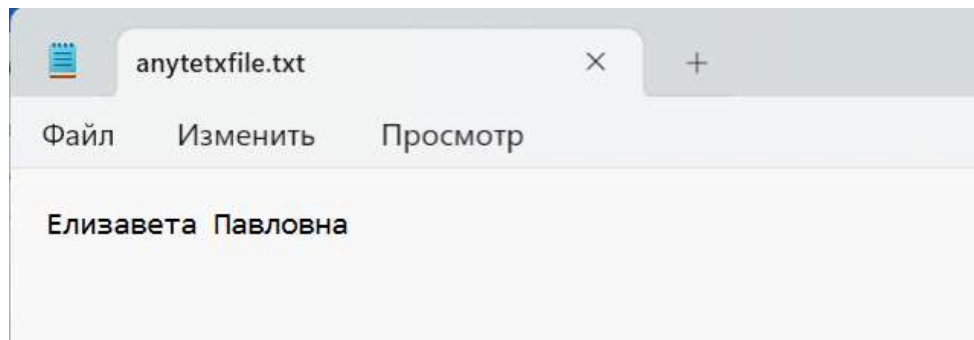


Теперь усложним задание, предположив, что в двух ветках могут быть одинаковые файлы и над ними работают разные разработчики:

- 1) Создадим ветку с названием newdev

```
PS C:\Users\maslo\Documents\LR1_OSP0> git branch newdev
PS C:\Users\maslo\Documents\LR1_OSP0> git branch
* master
newdev
```

- 2) Переключимся на нее. Добавим в файл с именем отчество. Зафиксируем изменения:



```
PS C:\Users\maslo\Documents\LR1_OSP0> git checkout newdev
Switched to branch 'newdev'
PS C:\Users\maslo\Documents\LR1_OSP0> git status
On branch newdev
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:   anytetxfile.txt

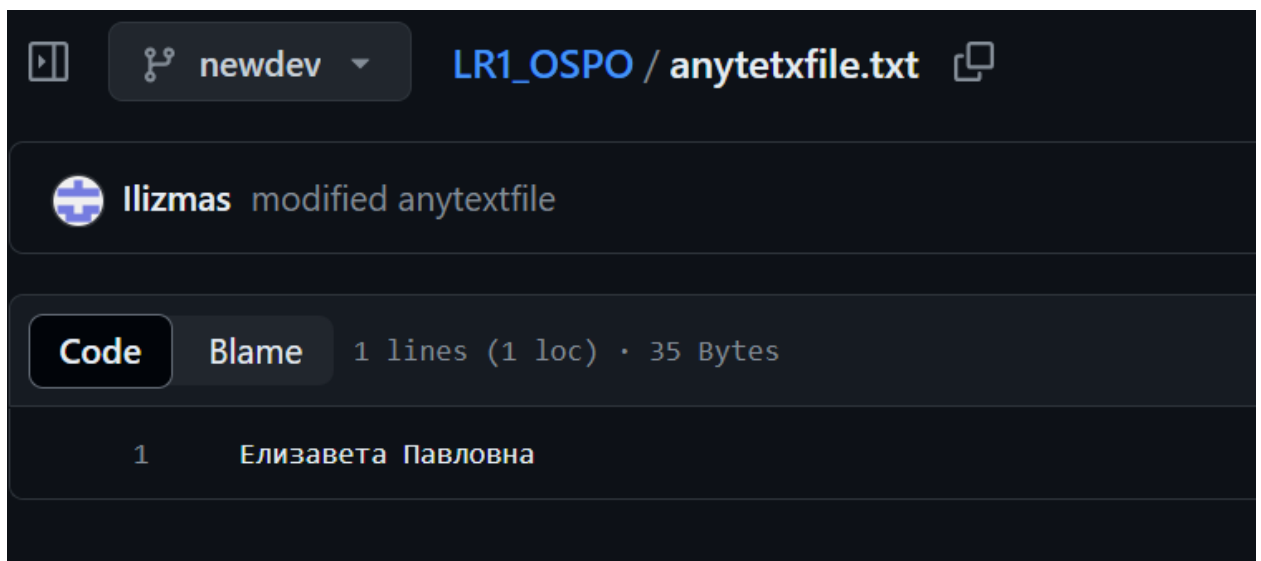
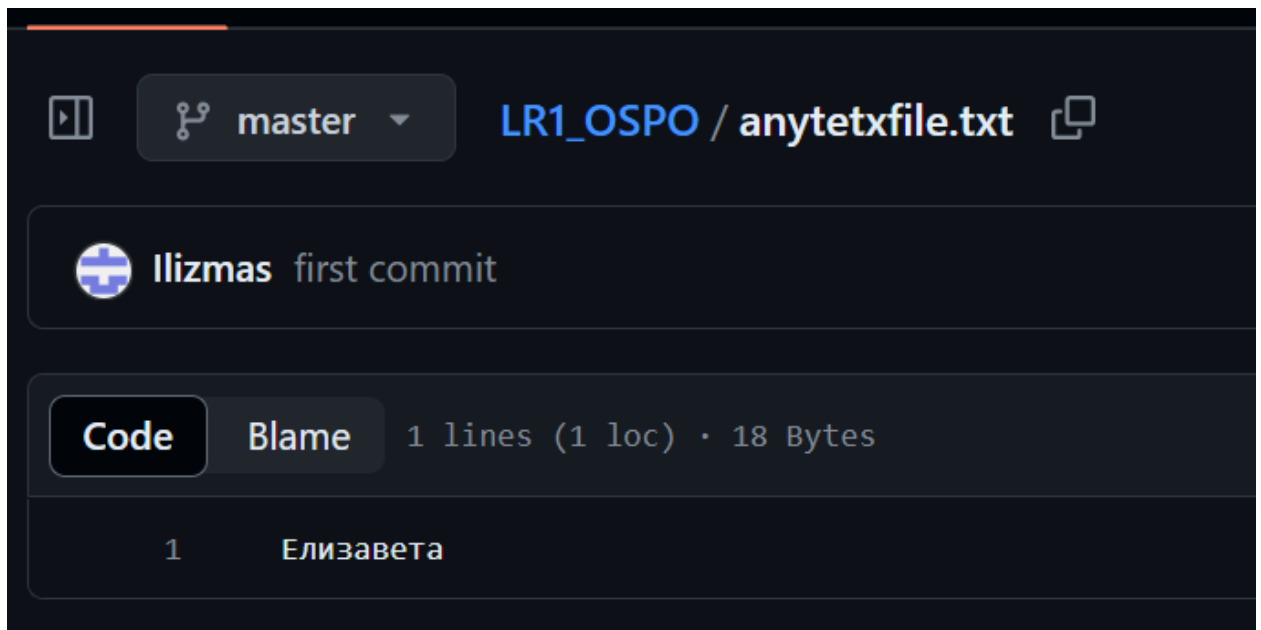
no changes added to commit (use "git add" and/or "git commit -a")
PS C:\Users\maslo\Documents\LR1_OSP0> git add .
PS C:\Users\maslo\Documents\LR1_OSP0> git status
On branch newdev
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    modified:   anytetxfile.txt
```

Закомитим изменения

```
PS C:\Users\maslo\Documents\LR1_OSP0> git commit -m "modified anytextfile"
[newdev 459f407] modified anytextfile
1 file changed, 1 insertion(+), 1 deletion(-)
PS C:\Users\maslo\Documents\LR1_OSP0> git log
commit 459f407cbac9310d4d01e0d67142f7c195ddff77 (HEAD -> newdev)
Author: Ilizmas <maslovaelizaveta1916@yandex.ru>
Date:   Mon May 6 21:26:25 2024 +0300

    modified anytextfile
```

Попробуем переключиться обратно на ветку master и посмотреть файл: в нем только имя, отчество в другой ветке



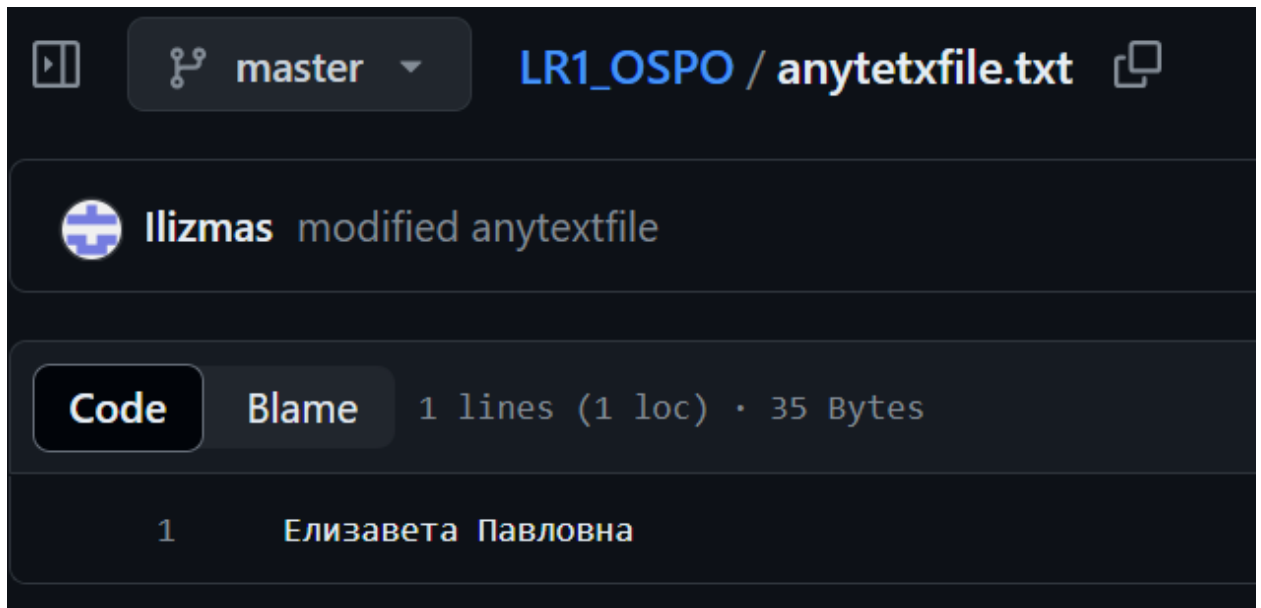
```
PS C:\Users\maslo\Documents\LR1_OSPO> git branch
master
* newdev
```

Объединим ветки



```
PS C:\Users\maslo\Documents\LR1_OSP0> git checkout master
Switched to branch 'master'
PS C:\Users\maslo\Documents\LR1_OSP0> git merge newdev
Updating bf20150..459f407
Fast-forward
 anytetxfile.txt | 2 +-
 1 file changed, 1 insertion(+), 1 deletion(-)
```

Откроем файл снова, там есть и имя, и отчество



```
PS C:\Users\maslo\Documents\LR1_OSP0> git push origin master
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/Ilizmas/LR1_OSP0
 bf20150..459f407  master -> master
PS C:\Users\maslo\Documents\LR1_OSP0> git commit -m "merge newdev"
On branch master
nothing to commit, working tree clean
```

Переключимся на ветку newdev и удалим в отчестве несколько букв. Зафиксируем, закоммитим. Переключимся на ветку master, добавим к отчеству несколько букв. Зафиксируем, закоммитим. Объединим ветки с помощью git merge newdev

## 8. Просмотр изменений и разрешение конфликтов

Нужно разрешить конфликты, которые возникли. Для этого введем команду git diff для просмотра изменений.

```
PS C:\Users\maslo\Documents\LR1_OSP0> git diff
```

У нас ничего не произошло, что может говорить о том, что конфликтов нет.

Если были бы конфликты, то мы бы увидели, что приложение пометит строки, в каких есть конфликты.

Над разделителем ===== мы бы увидели последний (HEAD) коммит, а под ним — конфликтующий. Таким образом, мы можем увидеть, чем они отличаются и решать, какая версия лучше. Или вовсе написать новую. В этом случае мы выбираем нужную строку (какую строку оставить, может вообще все хотим оставить), удаляем разделители, сохраняем файл, фиксируем изменения, коммитим, отправляем на сервер.

Так как у нас нет никаких проблем, объединяем ветки.

```
PS C:\Users\maslo\Documents\LR1_OSP0> git merge newdev
Already up to date.
```

## 9. Удаление веток на сервере

У нас есть несмерженная ветка, мы хотим ее удалить. Для этого необходимо использовать команду:

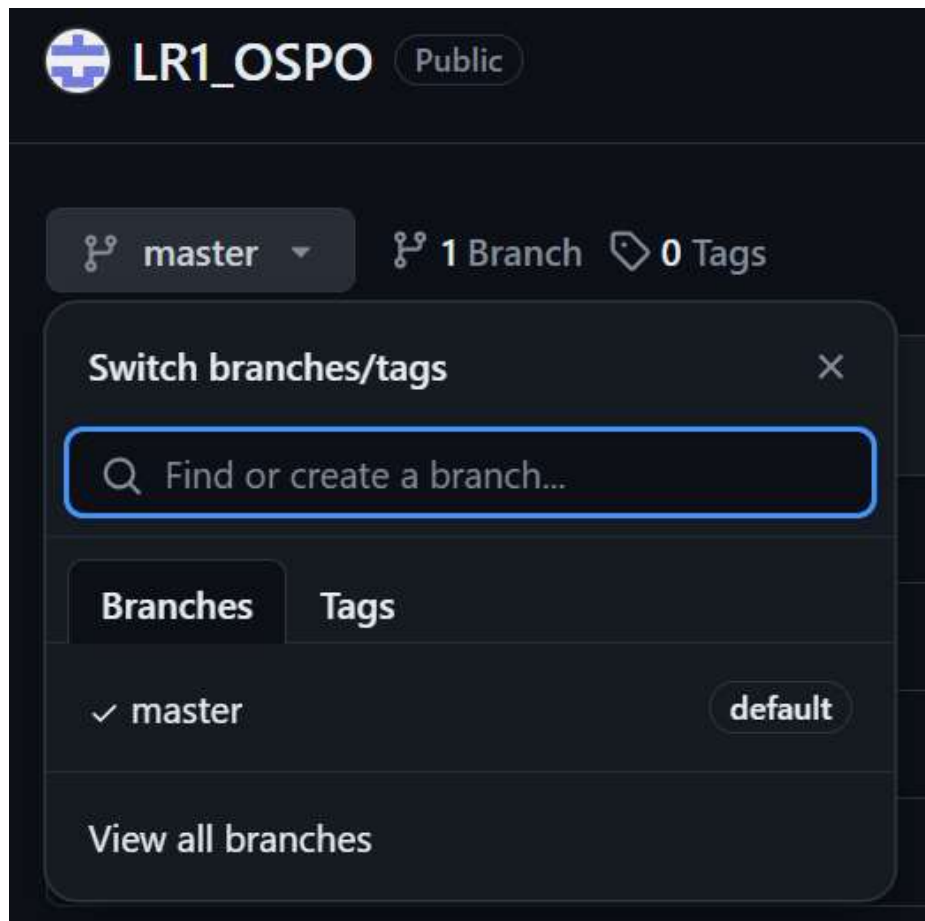
```
git branch -D second
```

```
PS C:\Users\maslo\Documents\LR1_OSP0> git checkout master
Already on 'master'
PS C:\Users\maslo\Documents\LR1_OSP0> git branch
* master
  newdev
PS C:\Users\maslo\Documents\LR1_OSP0> git branch -d newdev
Deleted branch newdev (was 459f407).
PS C:\Users\maslo\Documents\LR1_OSP0> git branch
* master
```

Однако если посмотреть на github, то наши ветки там остались. Для полного удаления используем команды:

```
PS C:\Users\maslo\Documents\LR1_OSP0> git push origin --delete newdev
To https://github.com/Ilizmas/LR1_OSP0
- [deleted]          newdev
PS C:\Users\maslo\Documents\LR1_OSP0> git push origin --delete second
```

В GitHub их больше нет, осталась только одна ветка:



## 10. Возврат к предыдущему состоянию

Гит позволяет вернуть выбранный файл к состоянию на момент определенного коммита. Это делается с помощью команды `checkout`, которую мы ранее использовали для переключения между ветками. Но она также может быть использована для переключения между коммитами. Чтобы посмотреть все коммиты, можно использовать команду `git log`, а потом нужно выбрать необходимый коммит, на который хотим откатиться. Нам достаточно указать его первые несколько символов:

```
PS C:\Users\maslo\Documents\LR1_OSP0> git checkout 459f407cbac
Note: switching to '459f407cbac'.

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 459f407 modified anytextfile
```

Если посмотреть файлы, то мы увидим, что они поменялись — мы вернулись назад. Это смогло произойти, потому что создалась псевдо-ветка начинающаяся на этом коммите. Посмотрите ветки:

```
PS C:\Users\maslo\Documents\LR1_OSP0> git branch
* (HEAD detached at 459f407)
master
```

По идее нужно создавать новую ветку и продолжать в ней работать, а потом смержиться, что мы и сделаем.

Так как мы абсолютно уверены, что коммит, к которому мы откатились единственно правильный, а все последующие неверные, их можно удалить с помощью команды:

`git reset --hard HEAD` — удаляет все, что не закоммичено.

## 11. Исправление коммита

Возможна ситуация, когда мы закрепили файлы, но еще не коммитили, и хотим убрать файлы из области закрепления. Для этого используется команда:

`git reset HEAD`

Наши файлы останутся такими же, но уйдут из области закрепления и снова будет показано, что есть измененные файлы:

```

PS C:\Users\maslo\Documents\LR1_OSP0> git checkout master
Switched to branch 'master'
PS C:\Users\maslo\Documents\LR1_OSP0> 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:   newfile.txt

no changes added to commit (use "git add" and/or "git commit -a")
PS C:\Users\maslo\Documents\LR1_OSP0> git add .
PS C:\Users\maslo\Documents\LR1_OSP0> git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   newfile.txt

PS C:\Users\maslo\Documents\LR1_OSP0> git reset HEAD
Unstaged changes after reset:
M       newfile.txt
PS C:\Users\maslo\Documents\LR1_OSP0> 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:   newfile.txt

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

```

Теперь снова добавим в область закрепления и закомитим, но еще не отправляем на сервер, можно вызвать ту же команду и тогда все, что осталось незакомиченным будет удалено.

## 12. Отправка только нужных файлов на сервер

В большинстве проектов есть файлы или целые директории, в которые мы не хотим (и, скорее всего, не захотим) коммитить. Мы можем удостовериться, что они случайно не попадут в `git add -A` при помощи файла «.gitignore»

Создадим вручную файл под названием «.gitignore» и сохраним его в директорию проекта. Файл «.gitignore» должен быть добавлен, закомичен и отправлен на сервер, как любой другой файл в проекте.

Также создадим два файла.

```

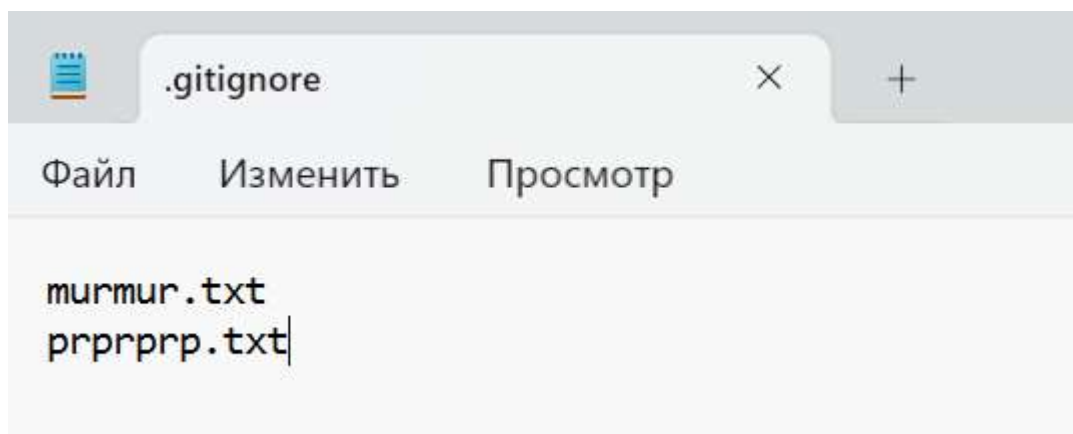
PS C:\Users\maslo\Documents\LR1_OSP0> git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
    modified:   .gitignore

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

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

```

Впишем в «.gitignore» файлы, который не хотим отправлять на сервер.  
Зафиксируем изменения, закоммитим, отправим на сервер



При проверке статуса видим, что система игнорирует файлы, которые находятся внутри «.gitignore».

```

PS C:\Users\maslo\Documents\LR1_OSP0> git add .gitignore
PS C:\Users\maslo\Documents\LR1_OSP0> git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    modified:   .gitignore

PS C:\Users\maslo\Documents\LR1_OSP0> git commit -m "try add 3 .gitignore"
[master 8a2aeld] try add 3 .gitignore
1 file changed, 2 insertions(+), 1 deletion(-)
PS C:\Users\maslo\Documents\LR1_OSP0> git push origin master
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 16 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 301 bytes | 301.00 KiB/s, done.

```

### 13. Совместная работа с git

Выберем себе напарника, дадим ему доступ



Kizyarich has been added as a collaborator on the repository.

Создадим новую папку и перейдем в нее в консоли.

Сделаем clone проекта, к которому нам дали доступ

`git clone https://github.com/Kizyarich/fanta`

Создадим свою ветку, чтобы не мешать работе напарника, в ней создадим и добавим файл, закоммитим и отправим на сервер.

```

PS C:\Users\maslo> cd C:\Users\maslo\Desktop\new
PS C:\Users\maslo\Desktop\new> git clone https://github.com/Kizyarich/fanta
Cloning into 'fanta'...
remote: Enumerating objects: 29, done.
remote: Counting objects: 100% (29/29), done.
remote: Compressing objects: 100% (13/13), done.
remote: Total 29 (delta 7), reused 25 (delta 6), pack-reused 0
Receiving objects: 100% (29/29), done.
Resolving deltas: 100% (7/7), done.
PS C:\Users\maslo\Desktop\new> git branch
PS C:\Users\maslo\Desktop\new> cd C:\Users\maslo\Desktop\new\fanta
PS C:\Users\maslo\Desktop\new\fanta> git branch
* master
PS C:\Users\maslo\Desktop\new\fanta> git branch libbranch
PS C:\Users\maslo\Desktop\new\fanta> git branch
  libbranch
* master
PS C:\Users\maslo\Desktop\new\fanta> git checkout libbranch
Switched to branch 'libbranch'
PS C:\Users\maslo\Desktop\new\fanta> git status
On branch libbranch
Untracked files:
  (use "git add <file>..." to include in what will be committed)
      proba_collab.txt

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

```

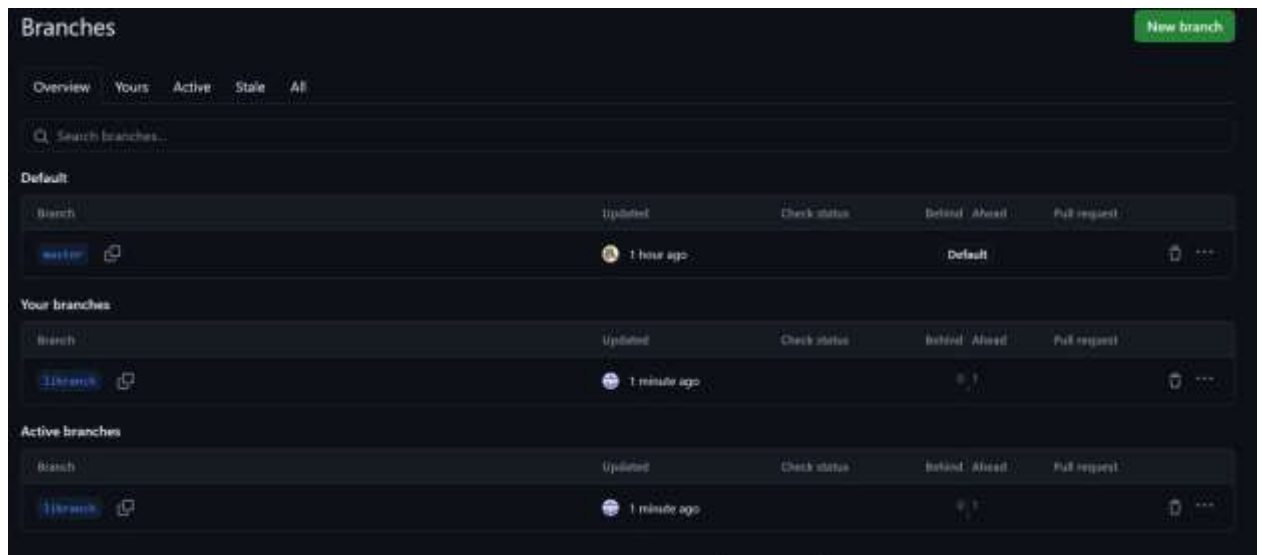
```

PS C:\Users\maslo\Desktop\new\fanta> git add .
PS C:\Users\maslo\Desktop\new\fanta> git status
On branch libbranch
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
      new file:   proba_collab.txt

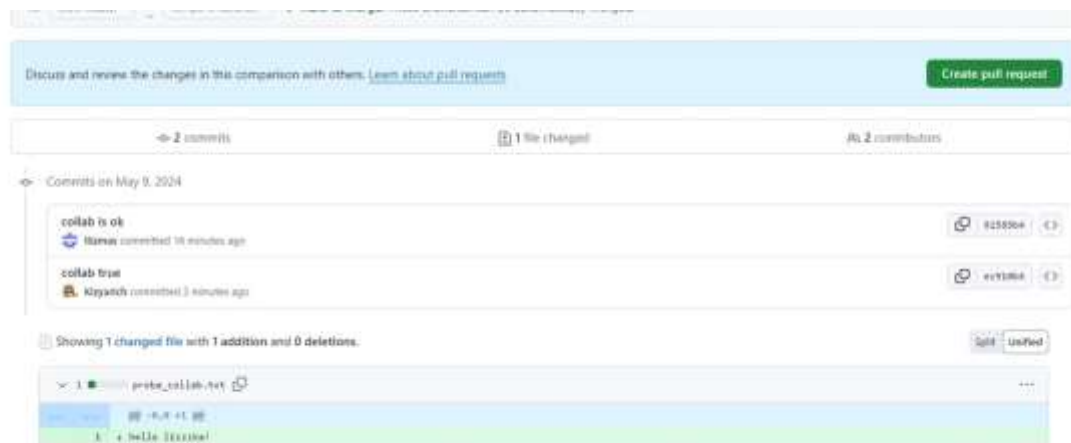
PS C:\Users\maslo\Desktop\new\fanta> git commit -m "collab is ok"
[libbranch 91589b4] collab is ok
 1 file changed, 1 insertion(+)
 create mode 100644 proba_collab.txt
PS C:\Users\maslo\Desktop\new\fanta> git push origin libbranch
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 16 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 305 bytes | 152.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
remote:
remote: Create a pull request for 'libbranch' on GitHub by visiting:
remote:   https://github.com/Kizyarich/fanta/pull/new/libbranch
remote:
To https://github.com/Kizyarich/fanta
 * [new branch]      libbranch -> libbranch

```

На сайте изменения видны



Попросим напарника скачать изменения, изменить скачанный файл и отправить:



Скачаем изменения

```

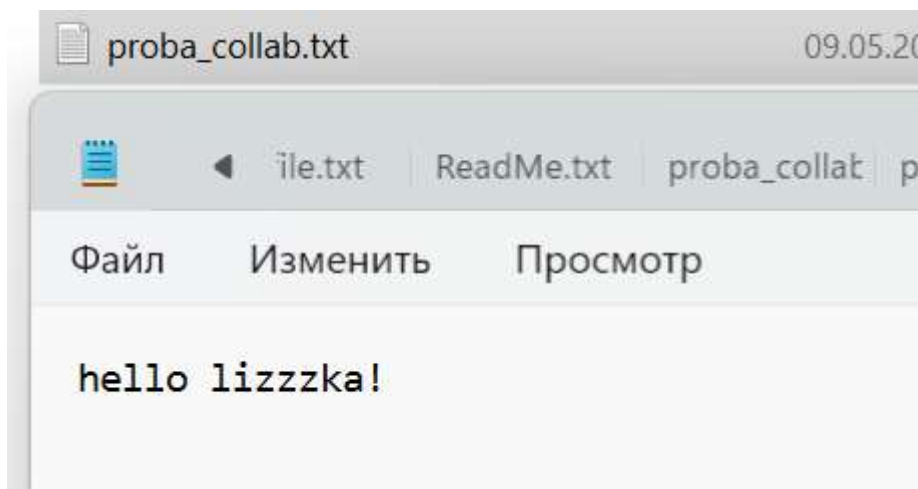
PS C:\Users\maslo\Desktop\new\fanta> git pull origin libbranch
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (1/1), done.
remote: Total 3 (delta 1), reused 3 (delta 1), pack-reused 0
Unpacking objects: 100% (3/3), 251 bytes | 22.00 KiB/s, done.
From https://github.com/Kizyarich/fanta
* branch          libbranch    -> FETCH_HEAD
  91589b4..ec910b4 libbranch    -> origin/libbranch
hint: Waiting for your editor to close the file... code --wait: l
ine 1: code: command not found
error: There was a problem with the editor 'code --wait'.
Not committing merge; use 'git commit' to complete the merge.
PS C:\Users\maslo\Desktop\new\fanta> git status
On branch libbranch
All conflicts fixed but you are still merging.
  (use "git commit" to conclude merge)

Changes to be committed:
  modified:   proba_collab.txt

PS C:\Users\maslo\Desktop\new\fanta> git add .
PS C:\Users\maslo\Desktop\new\fanta> git status
On branch libbranch
All conflicts fixed but you are still merging.
  (use "git commit" to conclude merge)

```

Изменения скачались



Отправим «вирус» напарнику

```

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

PS C:\Users\maslo\Desktop\new\fanta> git add .


```








```

PS C:\Users\maslo\Desktop\new\fanta> git commit -m "added troyan :)"
[libbranch 88d34d2] added troyan :)
PS C:\Users\maslo\Desktop\new\fanta> git push origin libbranch
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 16 threads
Compressing objects: 100% (5/5), done.
Writing objects: 100% (5/5), 609 bytes | 203.00 KiB/s, done.
Total 5 (delta 2), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (2/2), completed with 1 local object.
To https://github.com/Kizyarich/fanta
   ec910b4..88d34d2  libbranch -> libbranch

```

This branch is 4 commits ahead of master.

 **ilizmas** added troyan :) 88d34d2 - 48 minutes ago 14 Commits

 .gitignore	end ignore	2 hours ago
 README.md	Initial commit	3 hours ago
 anyfile.txt	end comm	2 hours ago
 proba_collab.txt	collab true	53 minutes ago
 pryamnew.txt	second commit	3 hours ago
 test.txt	first commit	3 hours ago
 troyan.bat	added troyan :)	52 minutes ago

### Commits

libbranch

Commits on May 9, 2024

added troyan :) 88d34d2 48 minutes ago

Вернемся к прошлой версии:



```

PS C:\Users\maslo\Desktop\new\fanta> git log
commit 88d34d2bba2ba870d88b7d570eb38f1bed4ec045 (HEAD -> libbranch, origin/libbranch)
Merge: 25eb1c6 ec910b4
Author: Ilizmas <maslovaelizaveta1916@yandex.ru>
Date: Thu May 9 23:25:40 2024 +0300

    added trojan :)

commit 25eb1c68898c78ef6ba58d8a43487754ebb10f5e
Author: Ilizmas <maslovaelizaveta1916@yandex.ru>
Date: Thu May 9 23:21:36 2024 +0300

    added trojan :)

commit ec910b4b9fbc2f6e9a9db86f6cb87e53751b85ea
Author: Kizyarich <iw.sabin@yandex.ru>
Date: Thu May 9 23:20:36 2024 +0300

    collab true

```

```

PS C:\Users\maslo\Desktop\new\fanta> git checkout ec910b4
Note: switching to 'ec910b4'.

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 ec910b4 collab true
PS C:\Users\maslo\Desktop\new\fanta>
PS C:\Users\maslo\Desktop\new\fanta> git branch
* (HEAD detached at ec910b4)
  libbranch
  master

```

Так как мы работаем над проектом с другим человеком, то будет правильнее переключиться на работающий коммит, проверить что все в нем работаете создать от него новую ветку и провести слияние.



```

PS C:\Users\maslo\Desktop\new\fanta> git switch -c normbranch
Switched to a new branch 'normbranch'
PS C:\Users\maslo\Desktop\new\fanta> git branch
  libbranch
  master
* normbranch
PS C:\Users\maslo\Desktop\new\fanta> git status
On branch normbranch
nothing to commit, working tree clean
PS C:\Users\maslo\Desktop\new\fanta> git log
commit ec910b4b9fbc2f6e9a9db86f6cb87e53751b85ea (HEAD -> normbranch)
Author: Kizyarich <iw.sabin@yandex.ru>
Date: Thu May 9 23:20:36 2024 +0300

    collab true

```

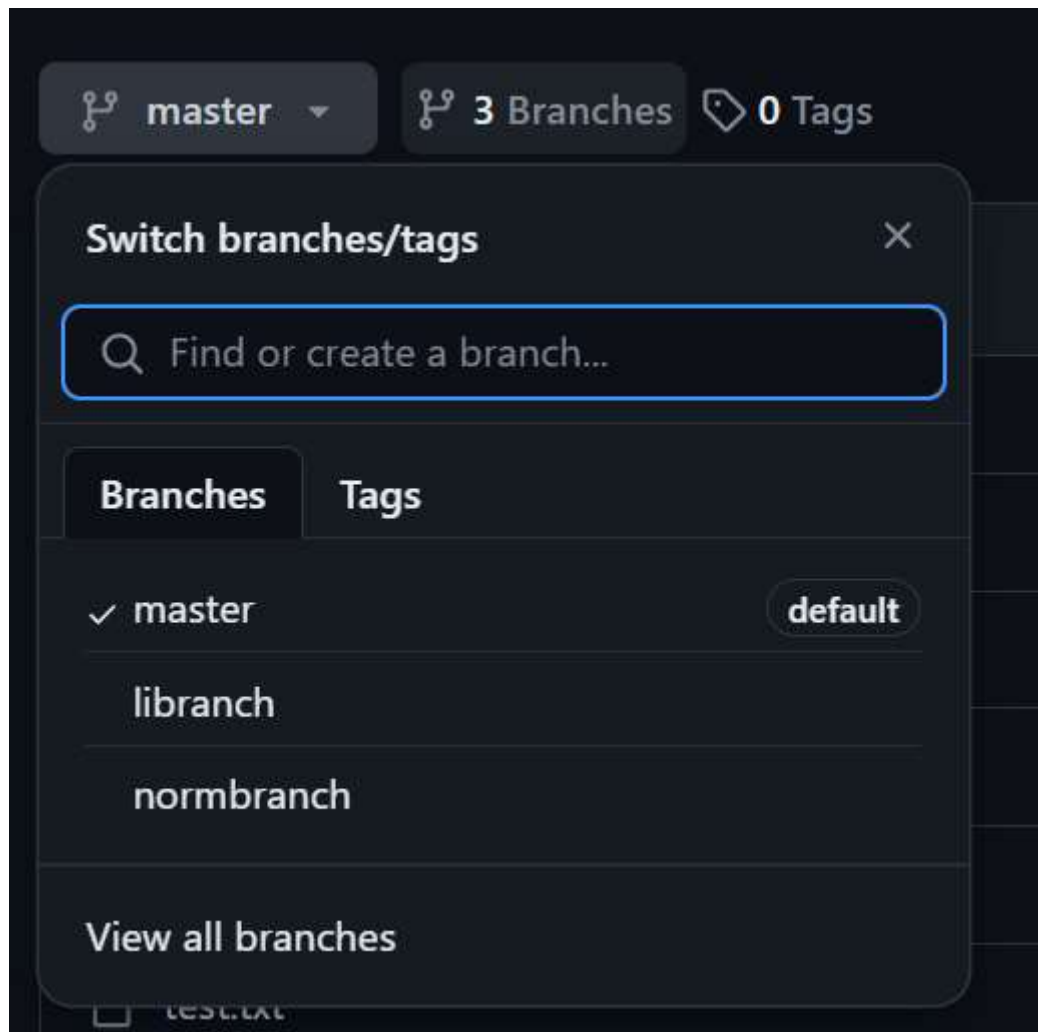
Отправим изменения на сервер, оставив вопрос слияния веток автору проекта:

```

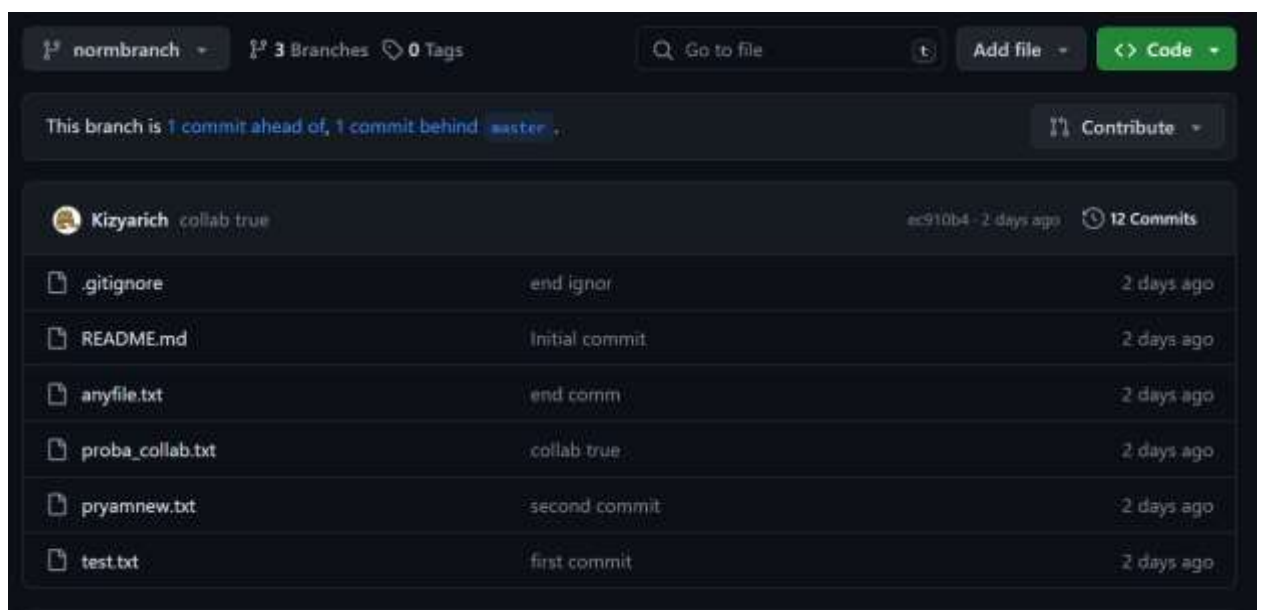
PS C:\Users\maslo\Desktop\new\fanta> git push origin normbranch
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
remote:
remote: Create a pull request for 'normbranch' on GitHub by visiting:
remote:   https://github.com/Kizyarich/fanta/pull/new/normbranch
remote:
To https://github.com/Kizyarich/fanta
 * [new branch]      normbranch -> normbranch

```

На сайте видны изменения:



В этой версии вируса уже нет.



**Вывод:** В ходе лабораторной работы были изучены принципы работы систем контроля версий на примере Git Hub. Была проведена совместная работа над одним проектом.