

Лабораторная работа №4

«GitHub»

Цель: научиться работать с веб-сервисом для хостинга проектов и их совместной разработки GitHub. Создать свой публичный репозиторий.

Теоретическая часть:

GitHub — крупнейший веб-сервис для хостинга IT-проектов и их совместной разработки. Веб-сервис основан на системе контроля версий Git и разработан на Ruby on Rails и Erlang компанией GitHub, Inc (ранее Logical Awesome). Сервис бесплатен для проектов с открытым исходным кодом и (с 2019 года) небольших частных проектов, предоставляя им все возможности (включая SSL), а для крупных корпоративных проектов предлагаются различные платные тарифные планы.

Практическая часть:

1) Регистрация

Перейдите на сайт <https://github.com> и нажмите кнопку «Sign Up»



Введите свои данные: выберите имя, которое ещё не занято, укажите пароль и адрес электронной почты, к которому у вас есть доступ.

```
Welcome to GitHub!
Let's begin the adventure

Enter your email*
✓ [redacted]@yandex.ru

Create a password*
✓ .....

Enter a username*
→ [redacted]

Continue
```

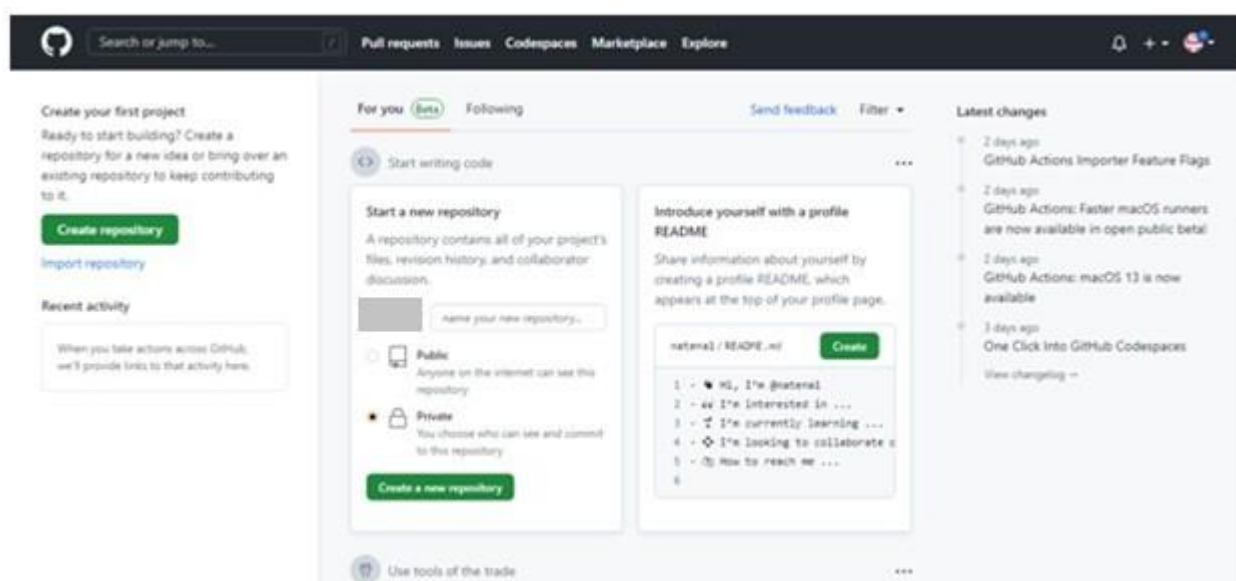
Система рекомендует использовать для пароля последовательность из 15 символов или 8, но с использованием хотя бы одной цифры и строчной буквы. Имя пользователя (username) должен быть уникальным. При этом он не может начинаться или заканчиваться дефисом или содержать символ подчеркивания.

Теперь нужно нажать кнопку «Continue», принять или отклонить предложение о подписке на рассылку и пройти валидацию.

На втором этапе регистрации нужно выбрать тарифный план. GitHub - бесплатный сервис, но предоставляет некоторые платные возможности. Выбираем тарифный план (бесплатный) и продолжаем регистрацию. Пользование GitHub бесплатно до тех пор, пока ваш код открыт для всех. Платные версии позволяют создавать закрытые, приватные репозитории, которые видите только вы.

После прохождения всех этапов регистрации на сайте на указанную вами почту придет письмо. Не забудьте верифицировать аккаунт: откройте первое письмо на почте от GitHub и пройдите по ссылке. Иначе вы не сможете создавать репозитории.

Войдите в аккаунт:



Теперь у вас есть профиль на GitHub.

2) Основные настройки профиля

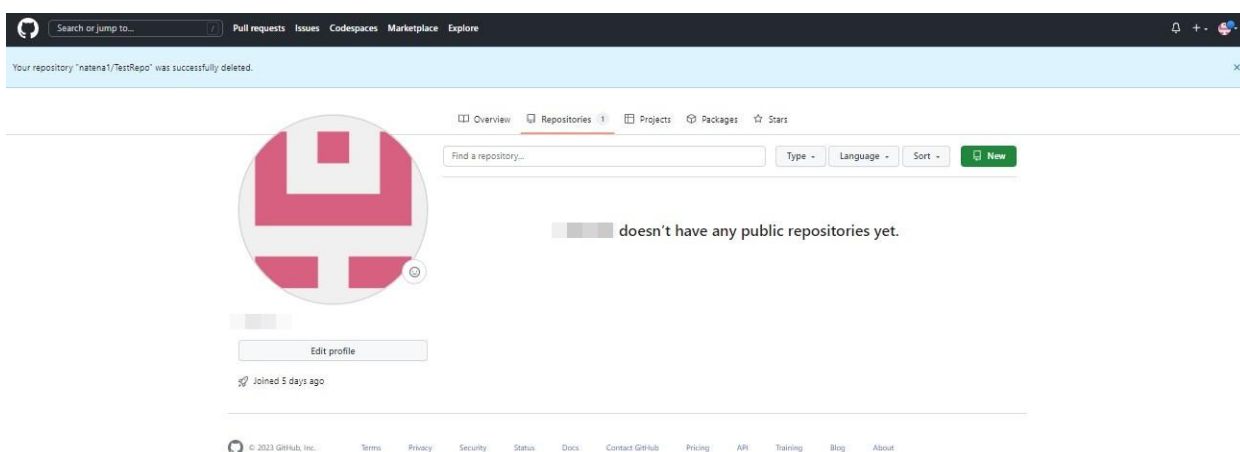
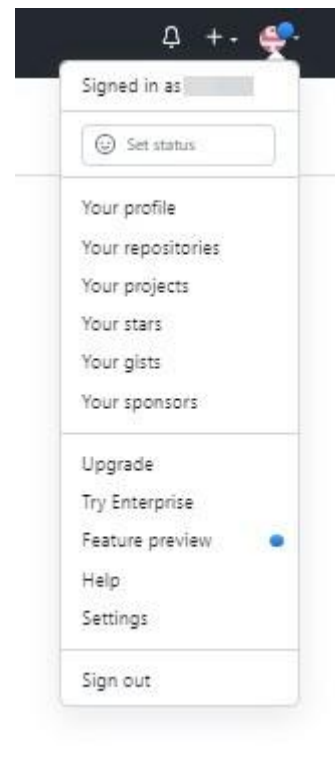
Чтобы открыть меню параметров нажмите на иконку вашего профиля в правом верхнем углу.

Profile – основная информация о вашем профиле: ваши репозитории, организации в которых вы состоите и т.д.

Repositories – ваши репозитории

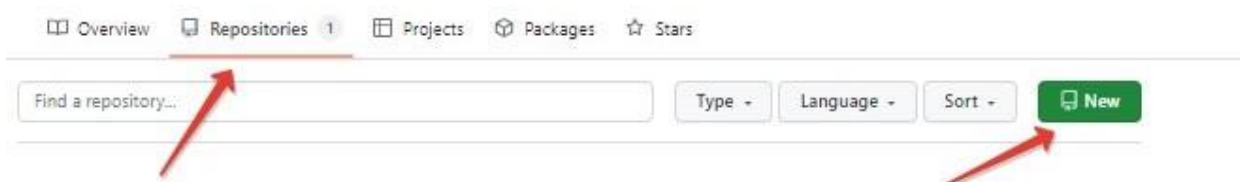
Settings – настройки вашего аккаунта

Ознакомьтесь с пунктами меню, а также со страницей вашего профиля (Your profile). При желании можете изменить аватар и добавить личную информацию в профиль.

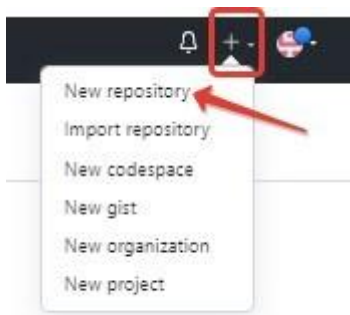


3) Создание нового репозитория

Для создания нового репозитория перейдите в пункт «Repositories» и нажмите кнопку «New».



Или нажмите в правом верхнем углу на кнопку «+» и выберите пункт меню «New repository».



Следующим шагом укажите основные параметры репозитория:

- Name – название репозитория (укажем имя «TestRepo»);
- Description – описание (можно оставить пустым);
- Тип репозитория – публичный/приватный (выбираем публичный);
- README file – стандартный для репозитория файл с описанием (ставим галочку);
- gitignore – файл со списком игнорируемых расширений, файлов, папок, описанных в нем.

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Owner *



Repository name *

/ TestRepo

✓ TestRepo is available.

Great repository names are short and memorable. Need inspiration? How about [silver-barnacle](#)?

Description (optional)



Public

Anyone on the internet can see this repository. You choose who can commit.



Private

You choose who can see and commit to this repository.

Initialize this repository with:

☒ Add a README file

This is where you can write a long description for your project. [Learn more about READMEs.](#)

Add .gitignore

.gitignore template: None

Choose which files not to track from a list of templates. [Learn more about ignoring files.](#)

Choose a license

License: None

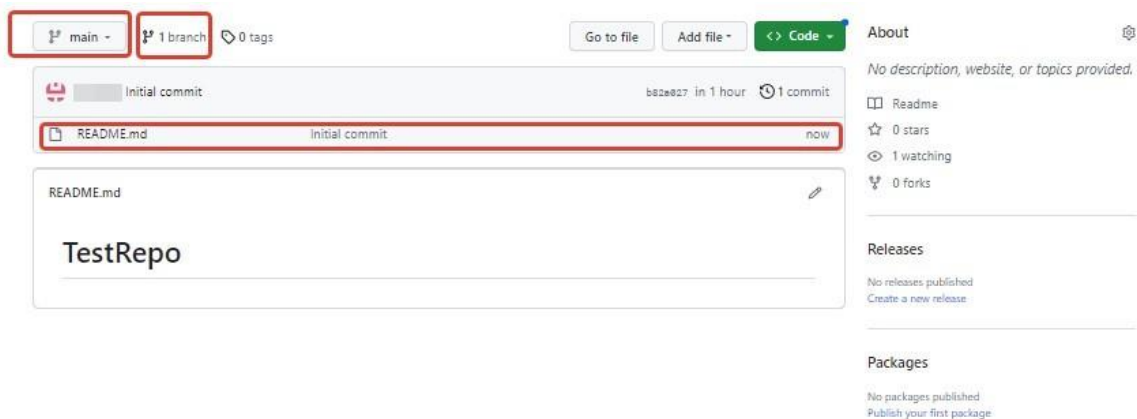
A license tells others what they can and can't do with your code. [Learn more about licenses.](#)

This will set `main` as the default branch. Change the default name in your [settings](#).

You are creating a public repository in your personal account.

Create repository

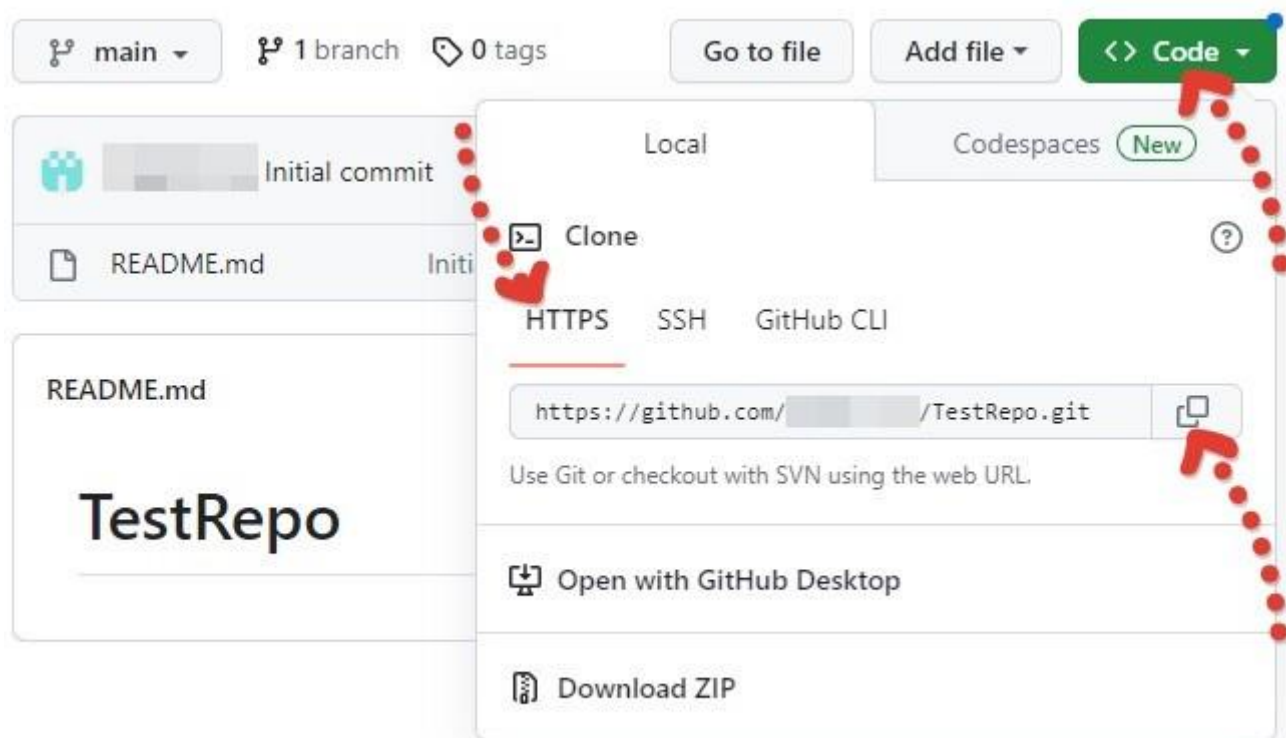
Нажимаем на кнопку «Create repository» и репозиторий создан.



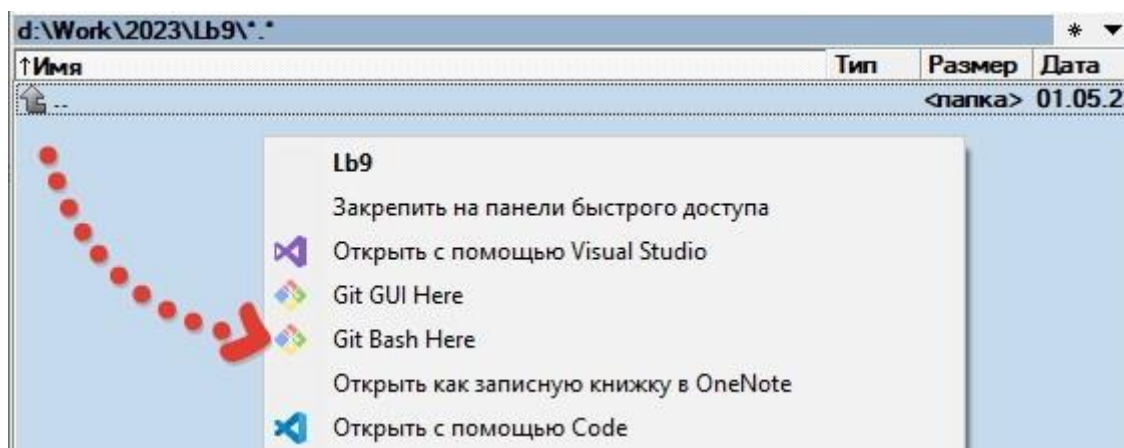
После создания репозитория вы в него попадаете и видите, что имеется одна основная ветка – ветка «main», а в репозитории в данный момент находится один файл – README.md, который был создан автоматически.

4) Клонирование репозитория

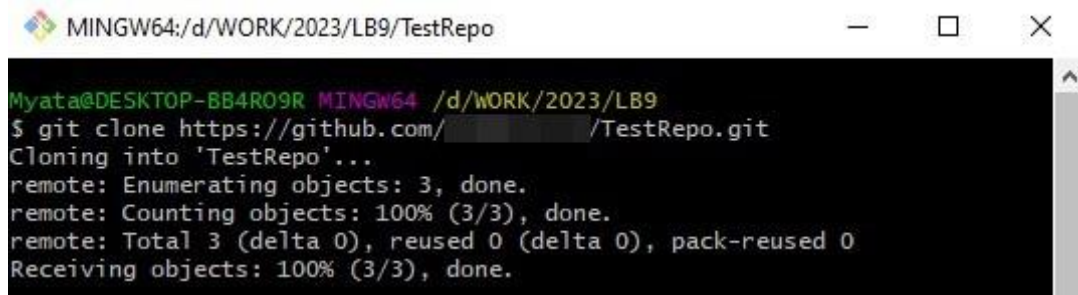
Для того чтобы клонировать репозиторий нужна ссылка на него. Получить ее можно на главной странице репозитория. Для этого нажмите на кнопку «Code» и скопируйте ссылку, убедившись, что выбран протокол HTTPS.



Создайте на локальном компьютере папку, в которой будете работать с удаленным репозиторием (например, LR4). Далее в этой папке открываем git-консоль (Git Bash)

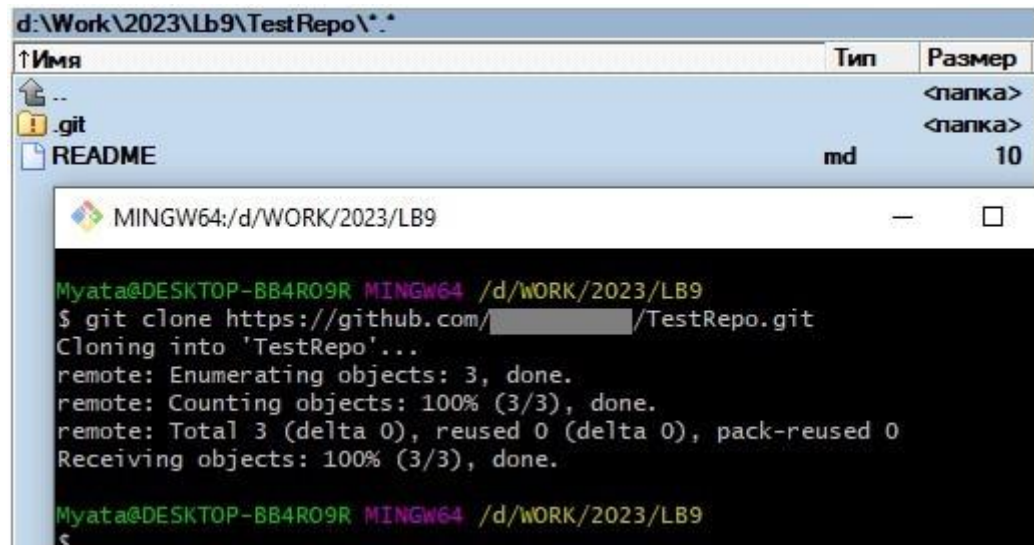


и прописываем команду `git clone «ссылка на репозиторий»`



```
MINGW64:/d/WORK/2023/LB9/TestRepo
Myata@DESKTOP-BB4R09R MINGW64 /d/WORK/2023/LB9
$ git clone https://github.com/[redacted]/TestRepo.git
Cloning into 'TestRepo'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (3/3), done.
```

Команда `clone` создаёт в папке LR4 копию репозитория TestRepo с указанного адреса с Github, содержащий файл README.md и папку .git.



Автоматически Github подключит к локальному репозиторию удалённый и назовёт его origin. Перейдите в папку TestRepo с помощью команды `cd` и проверьте это командой `git remote -v`. В нашем локальном репозитории автоматически создалась локальная ветка main, которую связана с веткой main в удаленном репозитории на Github. Git автоматически создает ветку master, а Github при связывании создает main и клонирует её в локальный репозиторий, т.е. у нас сразу появляется ветка с именем main и ничего дополнительно делать не нужно. Проверьте это командой `git status`.

```

Myata@DESKTOP-BB4R09R MINGW64 /d/WORK/2023/LB9
$ cd TestRepo

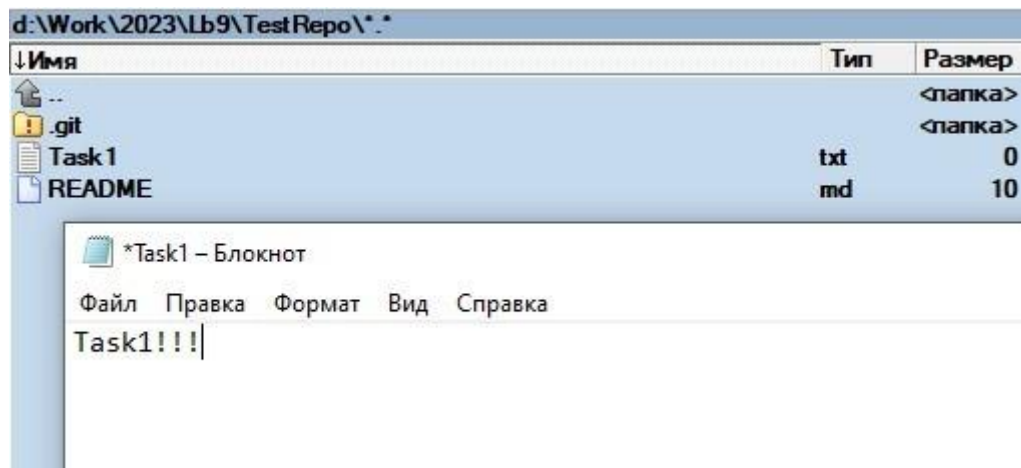
Myata@DESKTOP-BB4R09R MINGW64 /d/WORK/2023/LB9/TestRepo (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean

Myata@DESKTOP-BB4R09R MINGW64 /d/WORK/2023/LB9/TestRepo (main)
$ git remote -v
origin  https://github.com/[REDACTED]/TestRepo.git (fetch)
origin  https://github.com/[REDACTED]/TestRepo.git (push)

```

Теперь рассмотрим, как синхронизировать локальный репозиторий (клон) с оригиналом, который находится на Github. В клонированной папке TestRepo создайте локально файл Task1.txt, напишите в нем текст «Task1!!!», закройте его, сохранив изменения.



Теперь у нас репозиторий содержит 2 файла, а на Github только один.

Для того чтобы зафиксировать изменения, выполните команду

```
git add Task1.txt
```

и создайте коммит:

```

Myata@DESKTOP-BB4R09R MINGW64 /d/WORK/2023/LB9/TestRepo (main)
$ git add Task1.txt

Myata@DESKTOP-BB4R09R MINGW64 /d/WORK/2023/LB9/TestRepo (main)
$ git commit -m "add file Task1"
[main 2a919b4] add file Task1
1 file changed, 1 insertion(+)
create mode 100644 Task1.txt

```

Выполните команду **git push**. Эта команда берет имеющиеся локальные изменения и отправляет их в удалённый репозиторий.

```

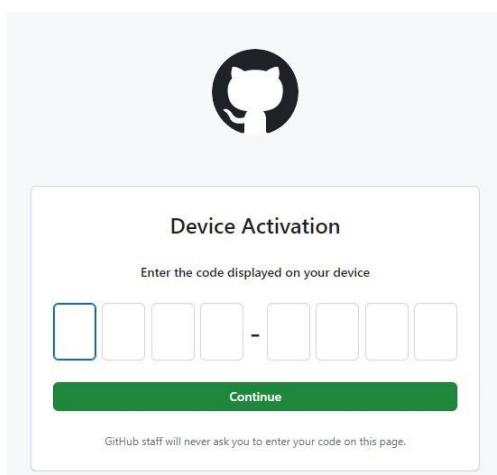
Myata@DESKTOP-BB4R09R MINGW64 /d/WORK/2023/LB9/TestRepo (main)
$ git push
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 12 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 287 bytes | 287.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/[REDACTED]/TestRepo.git
   39cd032..2a919b4  main -> main

Myata@DESKTOP-BB4R09R MINGW64 /d/WORK/2023/LB9/TestRepo (main)
$ |

```

При попытке выполнения команды `git push` система попросит подтверждение прав доступа к репозиторию на Github. При первой попытке система предложит выбор между двумя вариантами: через токен и через почту.

- При выборе второго варианта (получение кода на почту) система Github отправит письмо с паролем для подтверждения на указанную при регистрации почту. Далее нужно будет просто ввести полученный код в соответствующие окошки. После система не должна больше требовать подтверждения для обновления данных.



- При выборе первого варианта (подтверждение с использованием токена), вам нужно будет создать токен (смотрите следующий пункт 5). Полученный токен вы будете использовать в командной строке вместо пароля при выполнении операций `git push` через HTTPS.

В ответ на запрос введите своё имя на Github и токен (не пароль, а **токен!!!**):



Git for Windows

Username for 'https://github.com':

OK Cancel

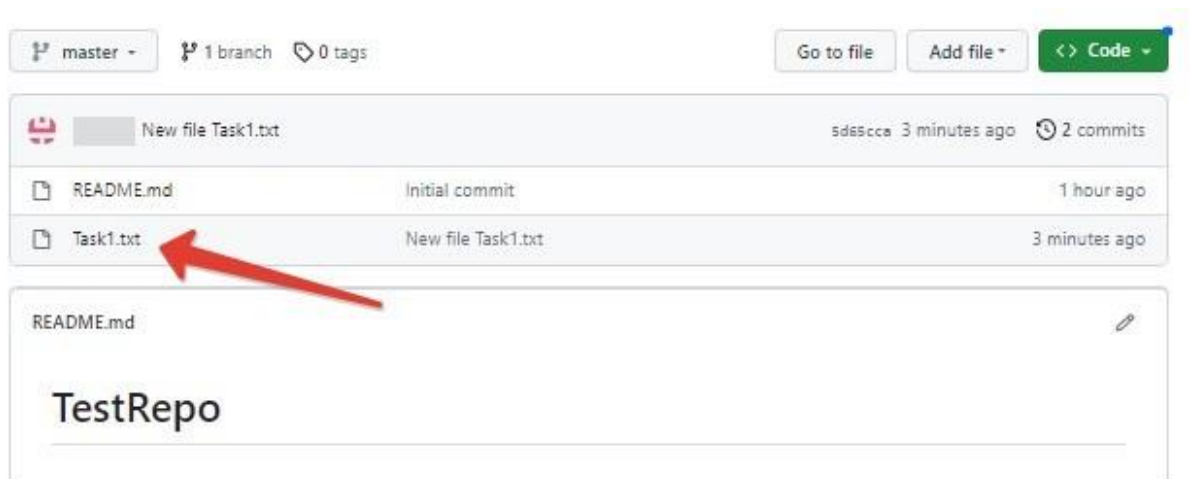


Git for Windows

Password for 'https://[redacted]@github.com':

OK Cancel

Откройте репозиторий на сайте Github, убедитесь, что файл добавился. Если вы все сделали правильно, то обновив главную страницу вашего репозитория увидите КОМИТ.



master 1 branch 0 tags Go to file Add file <> Code

File	Commit Message	Author	Time
README.md	Initial commit	sdscce	1 hour ago
Task1.txt	New file Task1.txt	sdscce	3 minutes ago

Task1.txt is highlighted with a red arrow.

README.md

TestRepo

5) Создание токена персонального доступа

Доступ к репозиториям на GitHub из командной строки можно получить двумя способами: по протоколу HTTPS или SSH. Будем использовать протокол HTTPS. Подробнее про различия протоколов можно прочитать в документации.

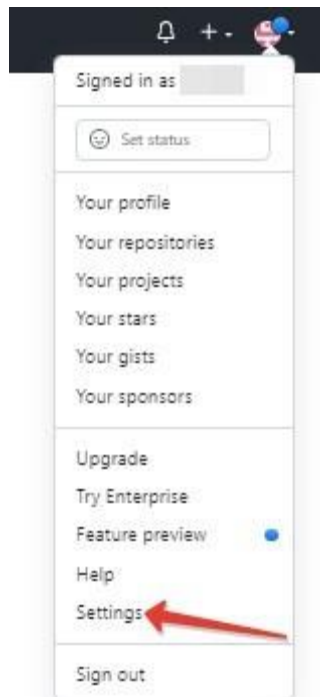
С 13 августа 2021 г. GitHub отменил использование паролей для аутентификации из командной строки с помощью Git в пользу токенов персонального доступа (Personal access tokens, PAT). Эти шаги были [сделаны с целью повышения безопасности](#) пользователей.

Для доступа (авторизации) к GitHub по протоколу HTTPS необходимо создать PAT.

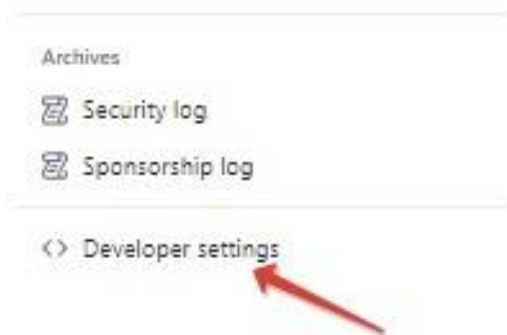
Преимущества PAT перед паролями:

- возможность генерации токена для каждого устройства, с которого необходимо получать доступ к репозиторию на GitHub
- возможность настройки ограничений для каждого токена при работе с GitHub
- возможность аннулирования токена для конкретного устройства
- возможность ограничить срок действия токена

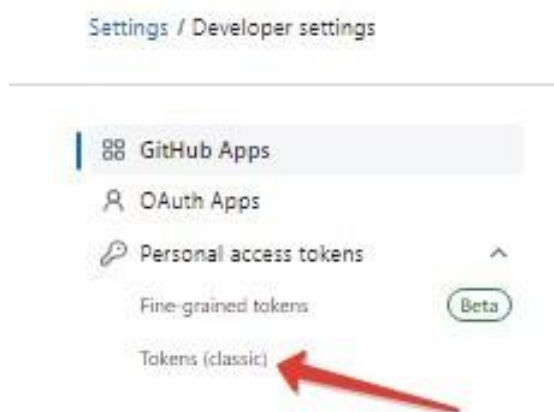
Для создания токена необходимо войти в настройки Settings вашего профиля.

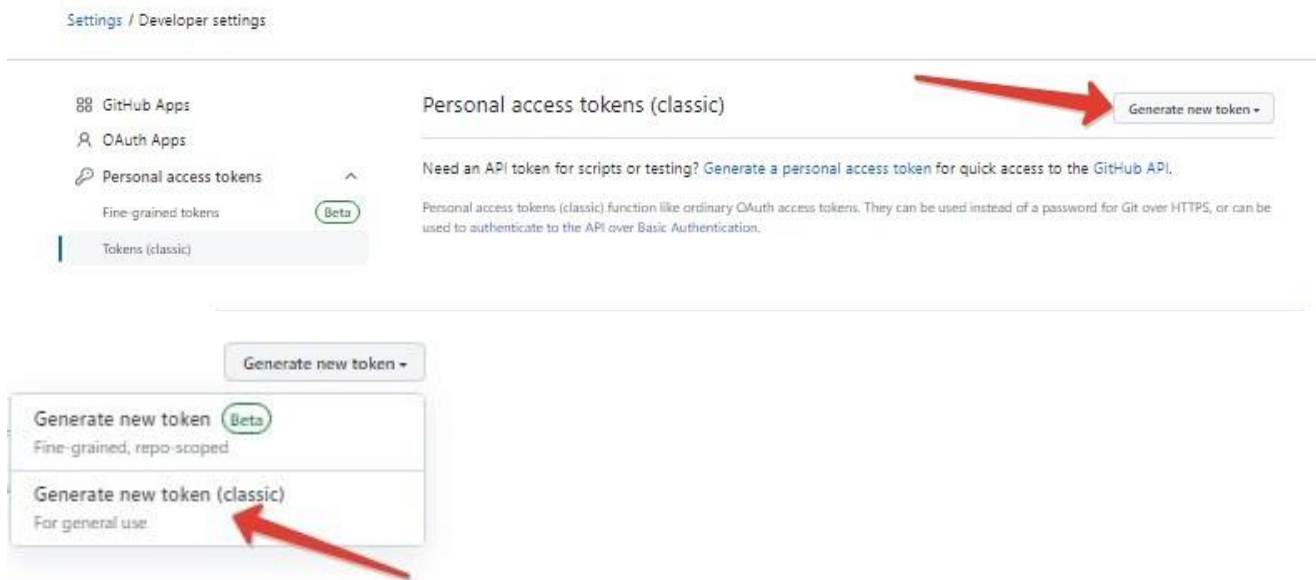


В левой панели нажмите пункт Developer settings



Также слева нажмите Personal access tokens → Tokens (classic), а затем кнопку Generate new token (classic)

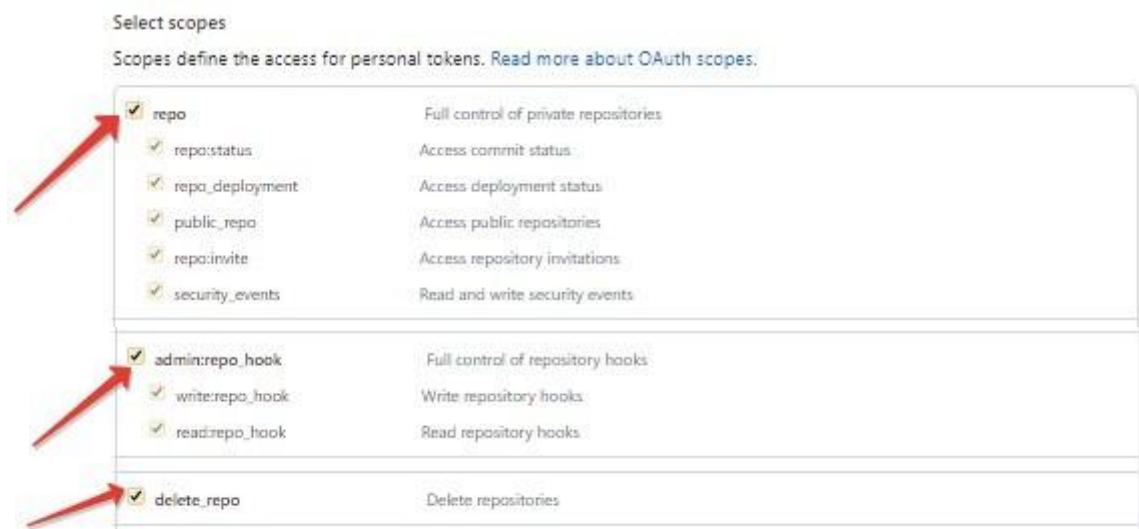




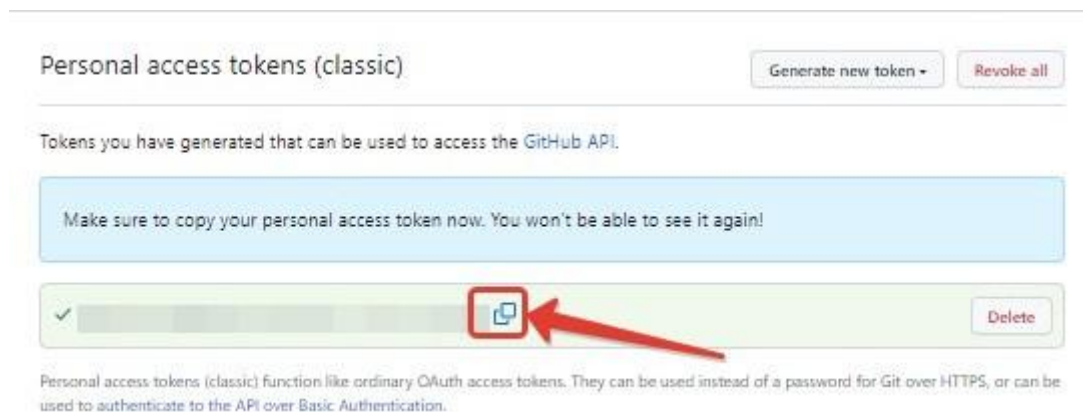
Дайте название (краткое описание) создаваемому токenu, а также срок его действия.



Выберите действия, которые будет разрешать данный токен. Чтобы использовать его для доступа к репозиториям из командной строки, отметьте пункты, где присутствует слово **repo**.



Нажмите кнопку  и скопируйте полученный токен в буфер обмена для последующего сохранения и использования. В целях безопасности, как только вы покинете страницу создания токена, вы больше не сможете просмотреть этот токен.



Предупреждение: относитесь к своим токенам как к паролям, и держите их в секрете.

Полученный токен вы будете использовать в командной строке вместо пароля при выполнении операций Git через HTTPS.

При запросе аутентификации, вам нужно будет ввести:

```
Username: ваш email
Password: ваш токен
```

При необходимости вы можете вернуться к созданному токену для его обновления или удаления.



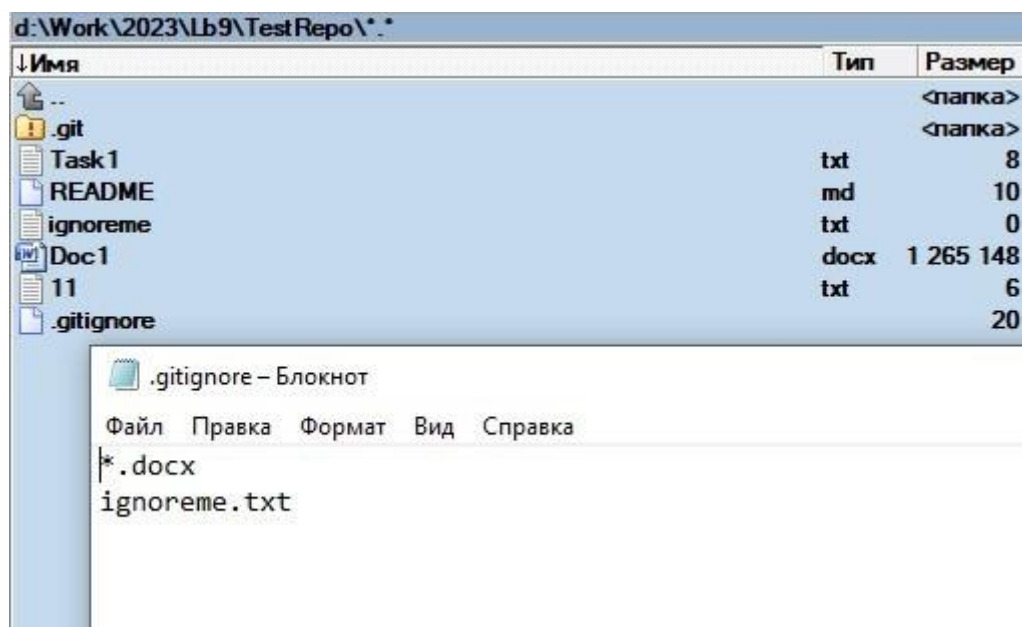
6) Добавление .gitignore

Файл .gitignore один из важнейших файлов в вашем репозитории. Он позволяет вам не отслеживать ненужные файлы.

Создание файла – для этого нужно создать файл с расширением .gitignore. **Важно!!!:** файл .gitignore может быть только один в директории.

В этом документе будут содержаться все игнорируемые файлы.

Например, будем игнорировать все файлы с расширением .docx и файл ignoreme.txt. **Важно!!!:** файл .gitignore нужно добавлять в первую очередь, пока файлов или папок, которые нужно игнорировать, еще нет на удаленном репозитории.



Проиндексируем изменения и сделаем коммит. После выполним команду `git push`.

```
MINGW64:/d/WORK/2023/LB9/TestRepo

Myata@DESKTOP-BB4RO9R MINGW64 /d/WORK/2023/LB9/TestRepo (main)
$ git add .

Myata@DESKTOP-BB4RO9R MINGW64 /d/WORK/2023/LB9/TestRepo (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    new file:   .gitignore
```

```

Myata@DESKTOP-BB4R09R MINGW64 /d/WORK/2023/LB9/TestRepo (main)
$ git commit -m "add gitignore"
[main e91adf3] add gitignore
1 file changed, 2 insertions(+)
create mode 100644 .gitignore

Myata@DESKTOP-BB4R09R MINGW64 /d/WORK/2023/LB9/TestRepo (main)
$ git push
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 12 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 356 bytes | 356.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/[REDACTED]/TestRepo.git
d459b97..e91adf3 main -> main

Myata@DESKTOP-BB4R09R MINGW64 /d/WORK/2023/LB9/TestRepo (main)
$

```

Проверим репозиторий на GitHub.

main

1 branch

0 tags

Go to file

Add file

<> Code

add gitignore

e91adf3 6 minutes ago

4 commits

.gitignore

add gitignore

6 minutes ago

11.txt

add 11

4 hours ago

README.md

Initial commit

5 hours ago

Task1.txt

add file Task1

4 hours ago

Как мы видим, файл ignoreme.txt и файл с расширением .docx игнорируются.

7) Получение данных

Для того чтобы получить данные с удаленного репозитория используется команда `git pull <remote>` – извлечь из указанного удаленного репозитория копию текущей ветки и немедленно объединить ее с локальной копией. Для примера создадим новую ветку и получим изменения из главной ветки, проверим командой `git log`.

```
MINGW64:/d/WORK/2023/LB9/TestRepo

Myata@DESKTOP-BB4R09R MINGW64 /d/WORK/2023/LB9/TestRepo (main)
$ git checkout -b test1
Switched to a new branch 'test1'

Myata@DESKTOP-BB4R09R MINGW64 /d/WORK/2023/LB9/TestRepo (test1)
$ git pull origin main
From https://github.com/[redacted]/TestRepo
* branch      main      -> FETCH_HEAD
Already up to date.

Myata@DESKTOP-BB4R09R MINGW64 /d/WORK/2023/LB9/TestRepo (test1)
$ git log
commit e91adf30f9b68230a3a0acbfff23397c0baef0593 (HEAD -> test1, origin/main, origin/HEAD, main)
Author: [redacted] <[redacted]@yandex.ru>
Date:   Mon May 1 22:56:14 2023 +0300

    add gitignore

commit d459b9711c0282c07ef0952dcef3363b9ca173ee
Author: [redacted] <[redacted]@yandex.ru>
Date:   Mon May 1 19:03:50 2023 +0300

    add 11

commit 2a919b4879450d66a1447b352166dfe0cf1c4b78
Author: [redacted] <[redacted]@yandex.ru>
Date:   Mon May 1 18:14:12 2023 +0300

    add file Task1

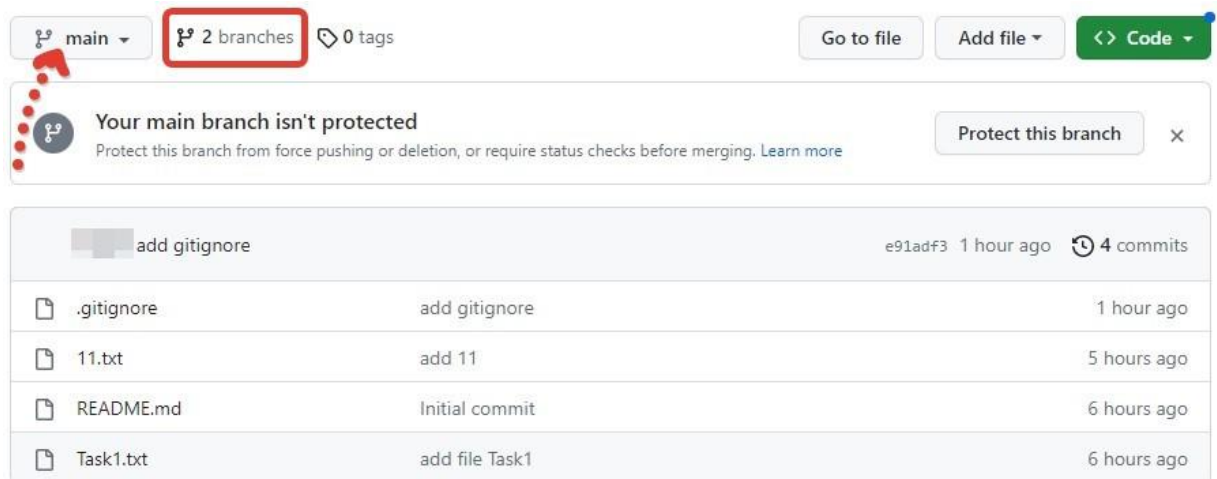
commit 39cd032c1dd05661defe41bf07504c39bcf7a06a
Author: [redacted] <132271107+[redacted]@users.noreply.github.com>
Date:   Mon May 1 17:35:53 2023 +0300
```

После отправим эту ветку на удаленный репозиторий.

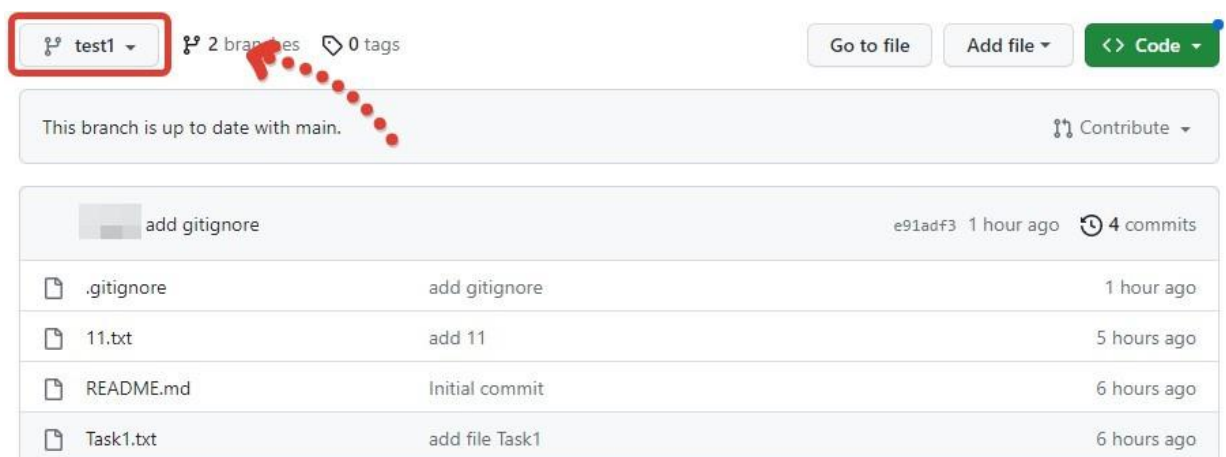
```
Myata@DESKTOP-BB4R09R MINGW64 /d/WORK/2023/LB9/TestRepo (test1)
$ git push origin test1
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: Create a pull request for 'test1' on GitHub by visiting:
remote:   https://github.com/[redacted]/TestRepo/pull/new/test1
remote:
To https://github.com/[redacted]/TestRepo.git
* [new branch]      test1 -> test1

Myata@DESKTOP-BB4R09R MINGW64 /d/WORK/2023/LB9/TestRepo (test1)
$
```

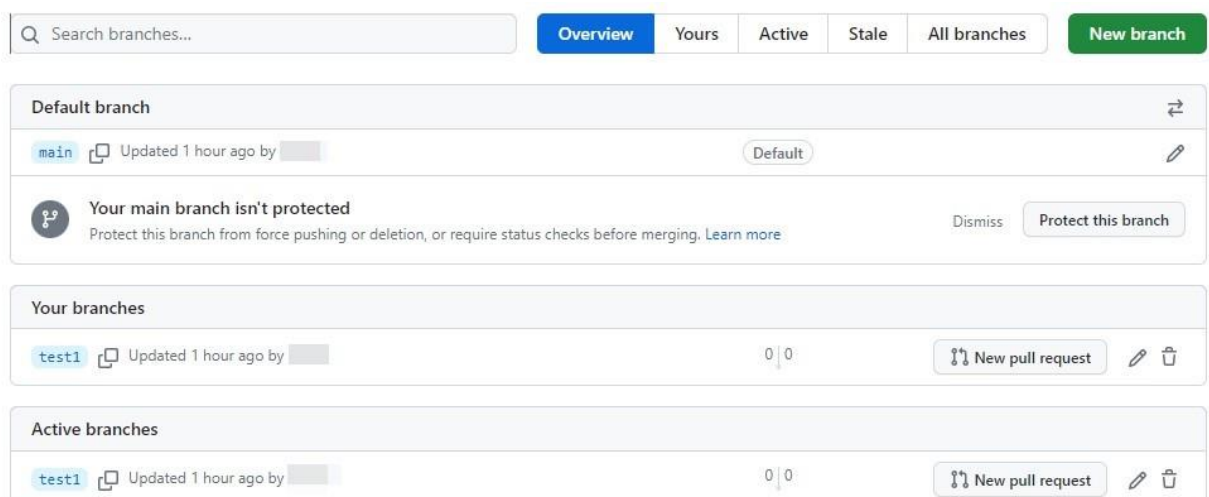
Проверим репозиторий на GitHub. Видим, что в репозитории 2 ветки. Сейчас активна ветка main. Выберем из выпадающего списка ветку test1.



Перейдя на ветку test1 видим то же самое содержимое, что и в ветке main. Кликнув на ссылку с количеством веток, перейдем на страницу с информацией о них.



Здесь можно увидеть все ветки, какая из них активна в данный момент.



Задание:

- 1) Регистрация и создание пустого репозитория, его клонирование
- 2) Добавление файла с расширением .txt
- 3) Добавление в игнорирование системой учета версий файла с названием «ignoremeplease.txt» и всех файлов с расширением «.xlsx»
- 4) Добавление новой ветки и получение данных в неё из главной

Требования к отчету:

- 1) Титульный
- 2) Создание репозитория на GitHub. Все введенные команды, ответы системы, пояснения, что было сделано между пунктами (например, отредактирован файл)