

Основы Git и Github

[Введение](#)

[Git](#)

[GitHub](#)

[Создание токена доступа в GitHub](#)

[Создание репозитория в GitHub](#)

[Подключение репозитория, первый commit, первый push](#)

[Все последующие commit-ы](#)

[Ссылки на дополнительную информацию](#)

[Ссылки на источники](#)

Введение

В данном материале, рассказано о том, что такое Git и зачем он нужен, а так же о том, что такое GitHub. Разберем, как создать репозиторий и загружать на него изменения. Здесь рассказано только о базовых функциях Git и GitHub, с более подробной информацией можно ознакомиться в разделе [Ссылки на дополнительную информацию](#).

Git

Начнем с того, что такое Git. Git – это система контроля версий. Что такое система контроля версий? Это система, записывающая изменения в файл или набор файлов в течение времени и позволяющая вернуться позже к определённой версии. СКВ не обязательно использовать для контроля изменений в коде, ее можно использовать для контроля версий файлов практически любого типа.

СКВ позволяет вернуть файлы к состоянию, в котором они были до изменений, вернуть проект к исходному состоянию, увидеть изменения, увидеть, кто последний менял что-то и вызвал проблему, кто поставил задачу и когда и многое другое. Использование СКВ также значит в целом, что, если вы сломали что-то или потеряли файлы, вы спокойно можете всё исправить, откатив версию программы до работающей.

Каждый раз, когда вы делаете commit, то есть сохраняете состояние своего проекта в Git, система запоминает, как выглядит каждый файл в этот момент, и сохраняет ссылку на этот снимок. Для увеличения эффективности, если файлы не были изменены, Git не запоминает эти файлы вновь, а только создаёт ссылку на предыдущую версию идентичного файла, который уже сохранён. Это очень важное отличие между Git и почти любой другой СКВ.

У Git есть три основных состояния, в которых могут находиться ваши файлы: изменён (modified), индексирован (staged) и зафиксирован (committed):

- К изменённым относятся файлы, которые поменялись, но ещё не были зафиксированы;
- Индексированный - это изменённый файл в его текущей версии, отмеченный для включения в следующий commit;
- Зафиксированный значит, что файл уже сохранён в вашей локальной базе;

Далее будет упрощение, для лучшего понимания того, что происходит.

Когда мы говорим о Git и GitHub, следует различать локальный репозиторий и удаленный. Локальный репозиторий хранит изменения(commit-ы) на вашем ПК, удаленный репозиторий это например сервер GitHub, на который вы отправляете изменения с помощью команды push. Соответственно, после внесения каких-либо изменений в файлы проекта, мы сначала фиксируем изменения в нашем локальном репозитории с помощью commit, а затем отправляем этот commit(эти изменения) на наш удаленный репозиторий с помощью push. Дальше на примере еще раз посмотрим, как это происходит.

Скачать Git можно [здесь](#).

GitHub

Немного о том, что такое GitHub. 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. Сервис бесплатен для проектов с открытым исходным кодом и (с 2019 года) небольших частных проектов, предоставляя им все возможности, а для крупных корпоративных проектов предлагаются различные платные тарифные планы.

Слоган сервиса — «Social Coding» — на русский можно перевести как «Пишем код вместе». На футболках же печатают совсем другую фразу: «Fork you!» («Ветвить тебя!»). С одной стороны, она созвучна с англоязычным ругательством и намекает на неформальную атмосферу. С другой, эти слова напоминают, что создавать новые форки с Git можно легко и безболезненно — традиционно, к созданию веток разработчики проектов с открытым исходным кодом относятся негативно.

Создатели сайта называют GitHub «социальной сетью для разработчиков». Кроме размещения кода, участники могут общаться, комментировать правки друг друга, а также следить за новостями знакомых.

Некоторые возможности/особенности GitHub:

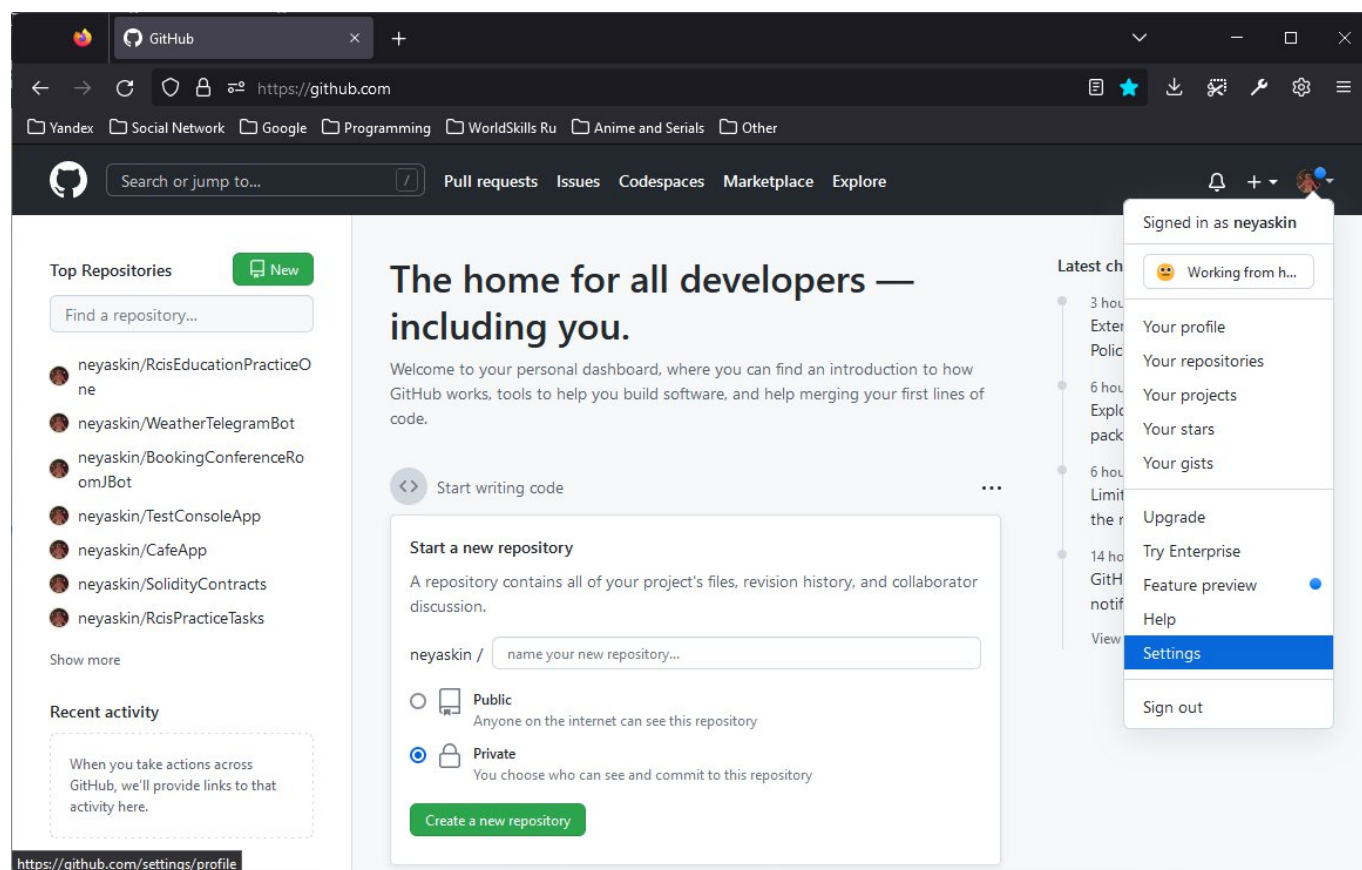
- Для проектов есть личные страницы, небольшие Вики и система отслеживания ошибок;
- Прямо на сайте можно просмотреть файлы проектов с подсветкой синтаксиса для большинства языков программирования;
- Можно создавать приватные репозитории, которые будут видны только вам и выбранным вами людям;
- Есть возможность прямого добавления новых файлов в свой репозиторий через веб-интерфейс сервиса;
- Код проектов можно не только скопировать через Git, но и скачать в виде обычных архивов с сайта;
- На сайте есть pastebin-сервис gist.github.com для быстрой публикации фрагментов кода;
- Файлы из репозитория могут автоматически публиковаться в виде статического сайта с помощью GitHub Pages;
- В 2019 году был запущен сервис GitHub Packages, позволяющий публиковать прямо на GitHub пакеты RubyGems, NuGet, npm, Maven, а также образы Docker;
- В том же году состоялся релиз системы автоматизации GitHub Actions. Помимо стандартных возможностей CI/CD, таких как сборка, тестирование и публикация кода, сервис предлагает тесную интеграцию с другими функциями GitHub, а также позволяет взаимодействовать со сторонними сервисами. Сервис предоставляется бесплатно для публичных репозиториях;

Зарегистрироваться на GitHub можно [здесь](#).

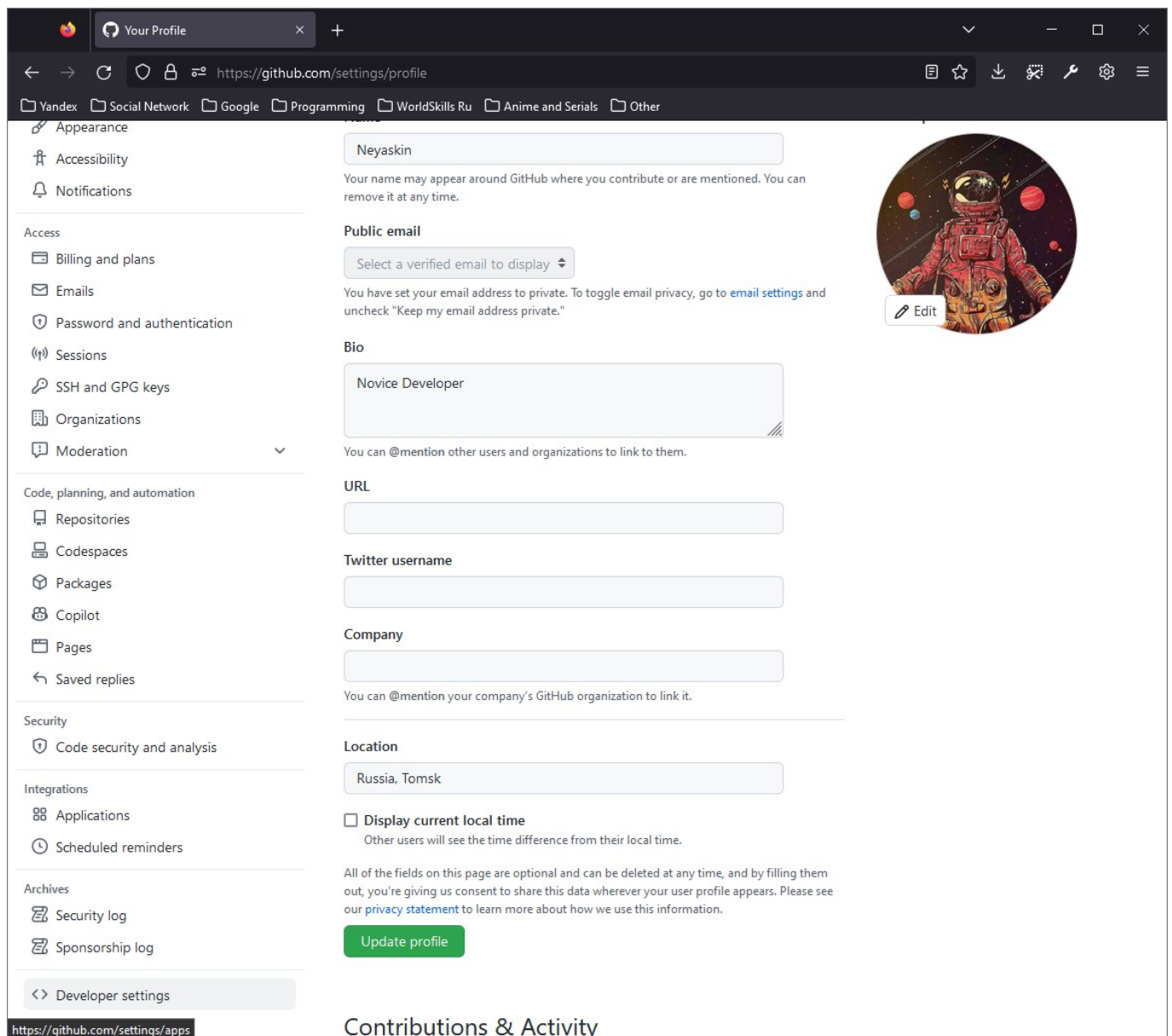
Создание токена доступа в GitHub

Данный токен позволит нам загружать изменения(commit-ы) на [GitHub](#) репозиторий, он заменит нам логин и пароль.

Заходим на сайт GitHub, вы должны быть уже зарегистрированы и авторизованны. Нажимаем на иконку профиля в правом верхнем углу, откроется выпадающий в нем нажимаем на «Settings».

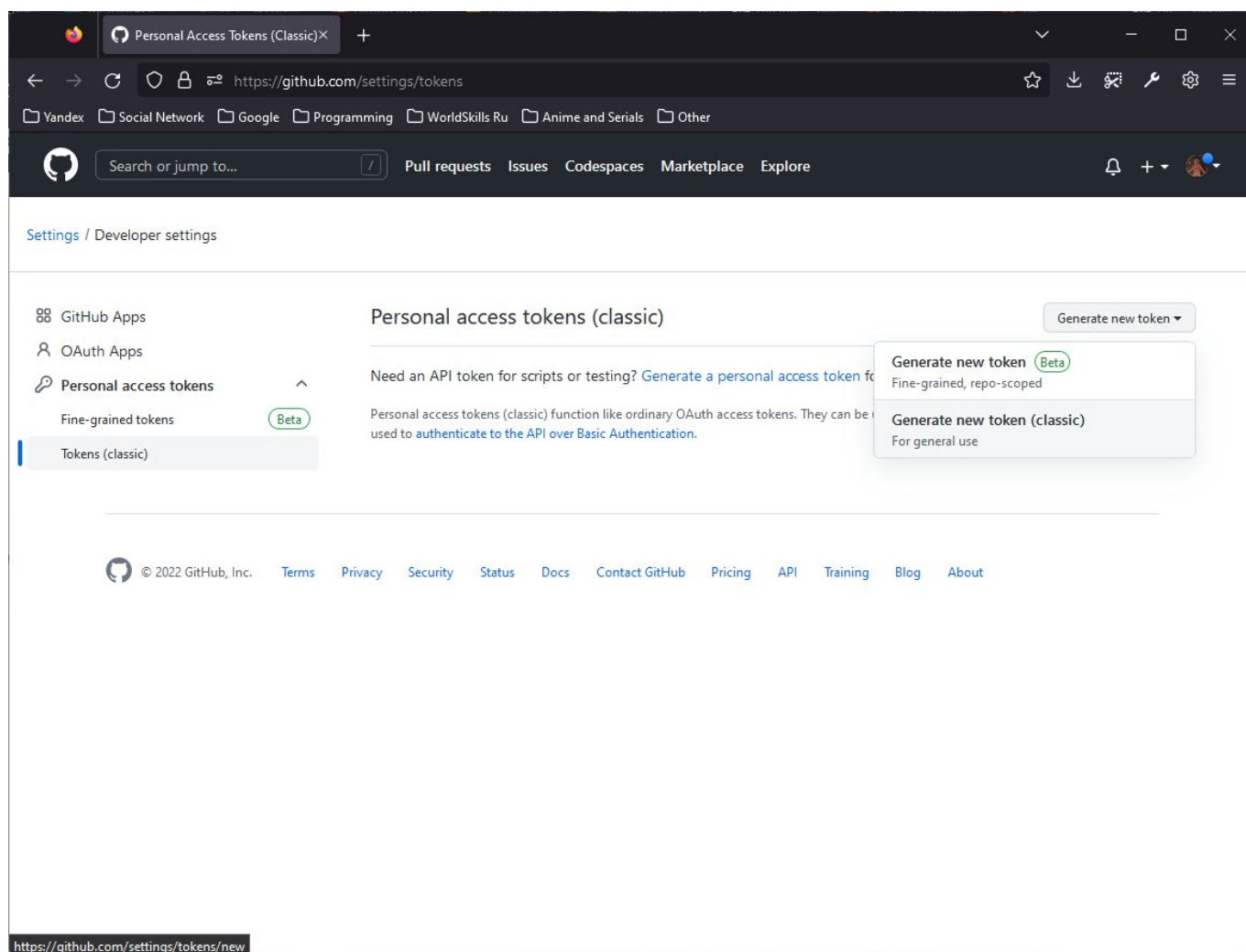


В меню настроек, с левой стороны нажимаем по «Developer settings», данный пункт находится в самом низу.



The screenshot shows the GitHub 'Your Profile' settings page. The browser address bar displays `https://github.com/settings/profile`. The left sidebar contains a list of settings categories: Appearance, Accessibility, Notifications, Access (with sub-items like Billing and plans, Emails, Password and authentication, Sessions, SSH and GPG keys, Organizations, and Moderation), Code, planning, and automation (with sub-items like Repositories, Codespaces, Packages, Copilot, Pages, and Saved replies), Security (with sub-item Code security and analysis), Integrations (with sub-items Applications and Scheduled reminders), Archives (with sub-items Security log and Sponsorship log), and Developer settings (highlighted with a '<>' icon). The main content area displays the user's profile information: Name (Neyaskin), Public email (Select a verified email to display), Bio (Novice Developer), URL, Twitter username, Company, and Location (Russia, Tomsk). A green 'Update profile' button is visible. Below the profile information, there is a section for 'Contributions & Activity'.

Далее нажимаем по «Personal access tokens», затем по «Tokens(classic)», потом по «Generate new token» и в выпадающем списке по «Generate new token(classic)».



Теперь нужно вписать название токена в поле «Note», выбрать срок действия токена в «Expiration» и указать какие права имеет токен, поставив галочку на «repo». После всего этого в самом низу нужно нажать на большую зеленую кнопку «Generate token».

Settings / Developer settings

New personal access token (classic)

Personal access tokens (classic) function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to [authenticate to the API over Basic Authentication](#).

Note

MyToken

What's this token for?

Expiration *

No expiration The token will never expire!

GitHub strongly recommends that you set an expiration date for your token to help keep your information secure. [Learn more](#)

Select scopes

Scopes define the access for personal tokens. [Read more about OAuth scopes.](#)

<input checked="" type="checkbox"/> repo	Full control of private repositories
<input checked="" type="checkbox"/> repo:status	Access commit status
<input checked="" type="checkbox"/> repo_deployment	Access deployment status
<input checked="" type="checkbox"/> public_repo	Access public repositories
<input checked="" type="checkbox"/> repo:invite	Access repository invitations
<input checked="" type="checkbox"/> security_events	Read and write security events

Скопировать токен будет возможно только 1 раз, сразу после его создания, поэтому его нужно где-нибудь сохранить/записать.

The screenshot shows the GitHub 'Personal Access Tokens (Classic)' settings page. At the top, a light blue notification bar states: 'Some of the scopes you've selected are included in other scopes. Only the minimum set of necessary scopes has been saved.' The left sidebar contains a menu with 'GitHub Apps', 'OAuth Apps', and 'Personal access tokens' (which is expanded to show 'Fine-grained tokens' with a 'Beta' badge and 'Tokens (classic)' which is selected). The main content area is titled 'Personal access tokens (classic)' and includes buttons for 'Generate new token' and 'Revoke all'. Below this, a message says: 'Tokens you have generated that can be used to access the GitHub API.' A light blue box contains the instruction: 'Make sure to copy your personal access token now. You won't be able to see it again!'. Below this, a green box displays a token: 'ghp_OvIK01cmq3yr5X5UsP5xtRMPLpg81Y1dVICK' with a copy icon and a 'Delete' button. At the bottom, a note explains: 'Personal access tokens (classic) function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to [authenticate to the API over Basic Authentication](#).' The footer includes the GitHub logo, copyright notice '© 2022 GitHub, Inc.', and links for Terms, Privacy, Security, Status, Docs, Contact GitHub, Pricing, API, Training, Blog, and About.

Personal Access Tokens (Classic) X

https://github.com/settings/tokens

Yandex Social Network Google Programming WorldSkills Ru Anime and Serials Other

Search or jump to... Pull requests Issues Codespaces Marketplace Explore

Some of the scopes you've selected are included in other scopes. Only the minimum set of necessary scopes has been saved.

Settings / Developer settings

GitHub Apps OAuth Apps Personal access tokens

Fine-grained tokens (Beta) Tokens (classic)

Personal access tokens (classic) Generate new token Revoke all

Tokens you have generated that can be used to access the GitHub API.

Make sure to copy your personal access token now. You won't be able to see it again!

ghp_OvIK01cmq3yr5X5UsP5xtRMPLpg81Y1dVICK Delete

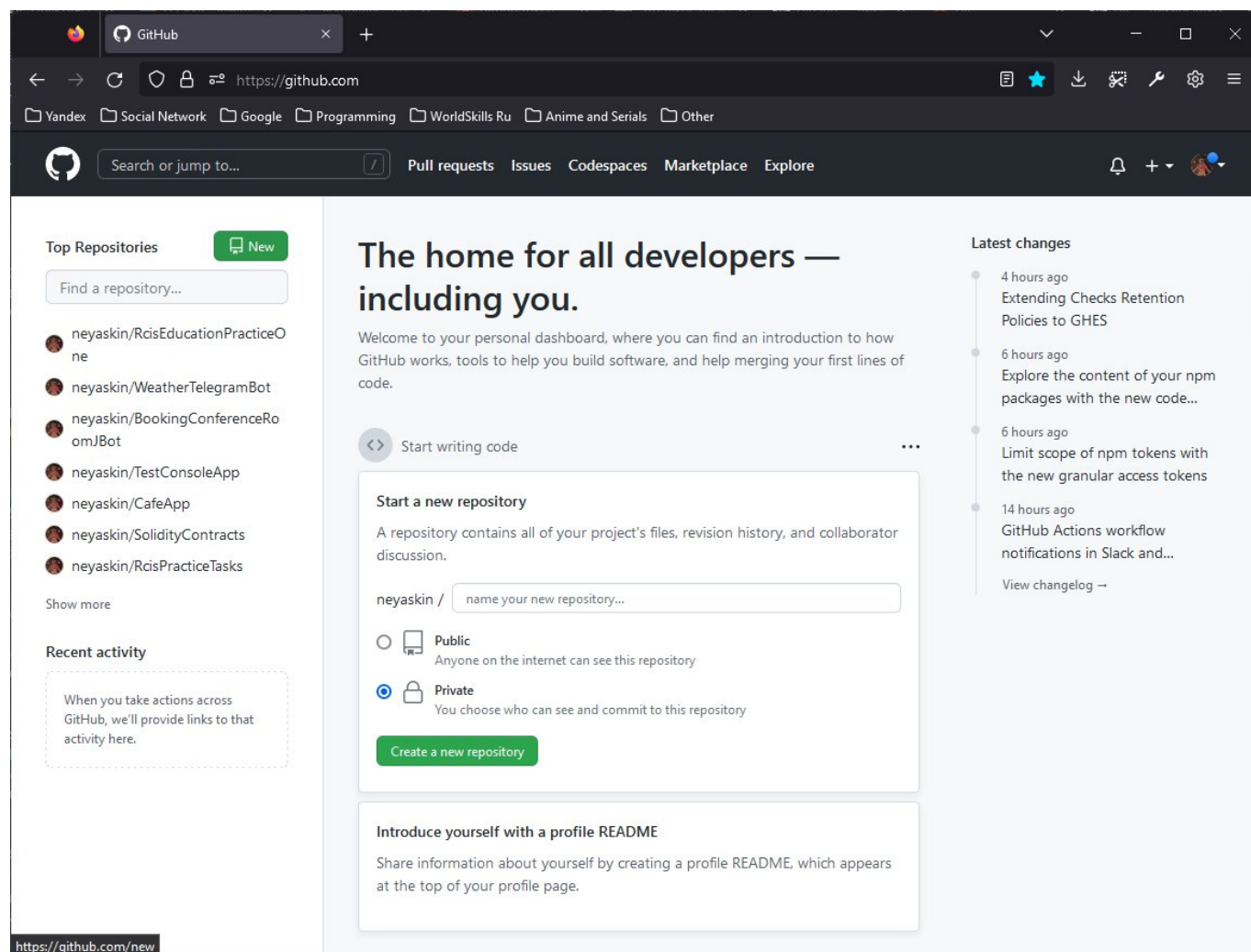
Personal access tokens (classic) function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to [authenticate to the API over Basic Authentication](#).

© 2022 GitHub, Inc. Terms Privacy Security Status Docs Contact GitHub Pricing API Training Blog About

Создание репозитория в GitHub

В репозитории будет храниться наш проект и все commit-ы(изменения), который мы будем на него загружать.

Заходим на [GitHub](https://github.com). Кликаем по зеленой кнопке «New» в левом верхнем углу.



Далее вводим название репозитория в поле «Repository name», и нажимаем на зеленую кнопку «Create repository» в самом низу.

Откроется страница с созданным репозиторием. Конечно же там нет никаких файлов, сейчас там просто небольшая инструкция, как сделать первый commit.

The screenshot shows a web browser window with the GitHub repository page for 'neyaskin/TestRep'. The browser's address bar shows the URL 'https://github.com/neyaskin/TestRep'. The repository page header includes the repository name, a 'Public' badge, and buttons for 'Pin', 'Unwatch' (1), 'Fork' (0), and 'Star' (0). Below the header is a navigation bar with links for 'Code', 'Issues', 'Pull requests', 'Actions', 'Projects', 'Wiki', 'Security', 'Insights', and 'Settings'. The main content area features a light blue box titled 'Quick setup — if you've done this kind of thing before'. This box contains a 'Set up in Desktop' button, an 'or' separator, and tabs for 'HTTPS' and 'SSH'. The 'HTTPS' tab is selected, showing the URL 'https://github.com/neyaskin/TestRep.git'. Below this, a text line says 'Get started by creating a new file or uploading an existing file. We recommend every repository include a README, LICENSE, and .gitignore.' Below the blue box are three sections with instructions for creating or pushing a repository from the command line, each with a copy icon. The first section, '...or create a new repository on the command line', contains the following commands:

```
echo "# TestRep" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/neyaskin/TestRep.git
git push -u origin main
```

 The second section, '...or push an existing repository from the command line', contains:

```
git remote add origin https://github.com/neyaskin/TestRep.git
git branch -M main
git push -u origin main
```

 The third section, '...or import code from another repository', includes the text 'You can initialize this repository with code from a Subversion, Mercurial, or TFS project.' and an 'Import code' button.

Quick setup — if you've done this kind of thing before

Set up in Desktop or HTTPS SSH `https://github.com/neyaskin/TestRep.git`

Get started by creating a new file or uploading an existing file. We recommend every repository include a README, LICENSE, and .gitignore.

...or create a new repository on the command line

```
echo "# TestRep" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/neyaskin/TestRep.git
git push -u origin main
```

...or push an existing repository from the command line

```
git remote add origin https://github.com/neyaskin/TestRep.git
git branch -M main
git push -u origin main
```

...or import code from another repository

You can initialize this repository with code from a Subversion, Mercurial, or TFS project.

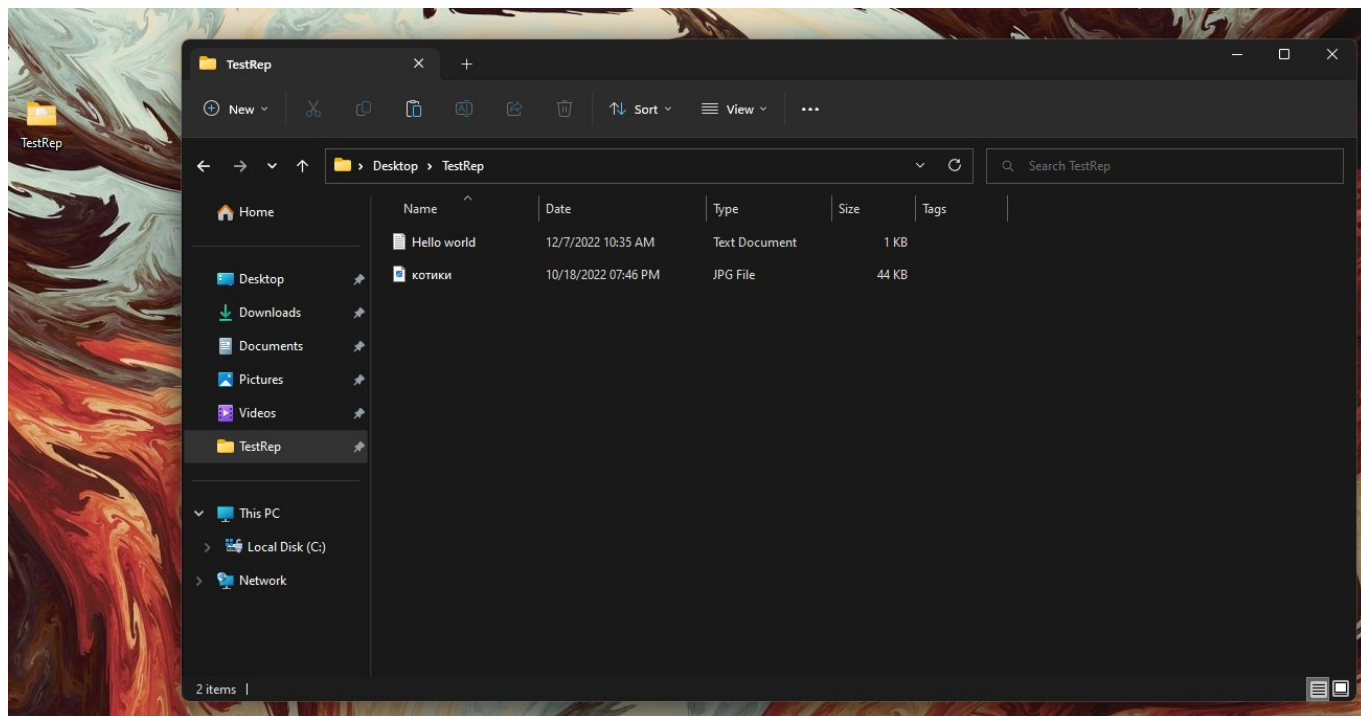
Import code

Подключение репозитория, первый commit, первый push

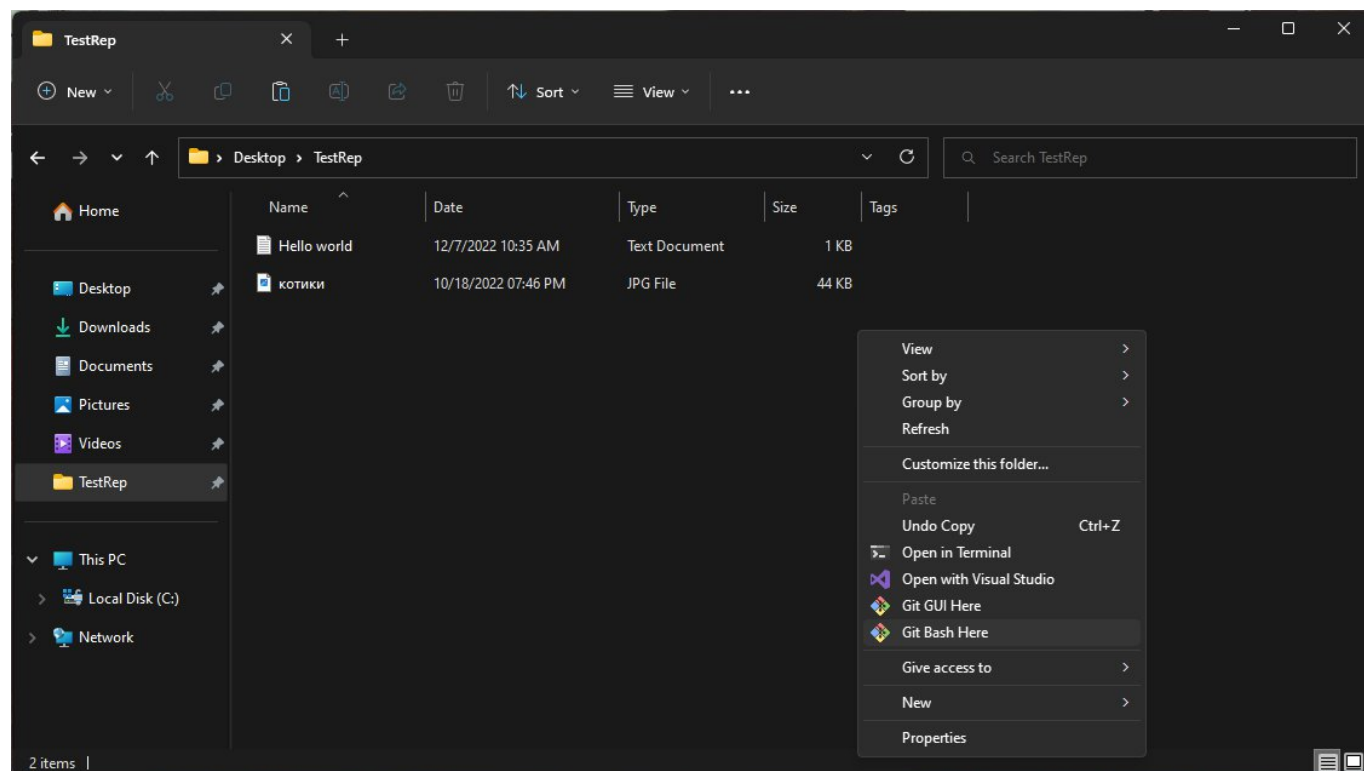
Допустим, у нас есть какая то директория(с названием «TestRep»), содержимое которой(текстовый файл и картинка) нам нужно загрузить в наш репозиторий.

Примечание

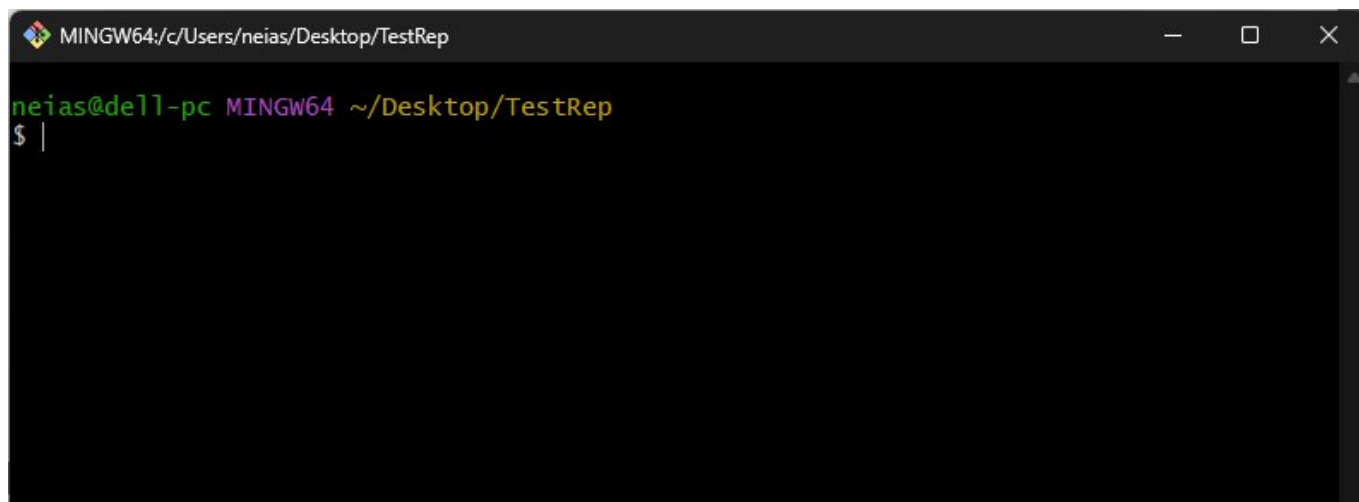
Git уже должен быть установлен.



Нажимаем правой кнопкой мыши в директории и выбираем пункт «Git Bash Here».

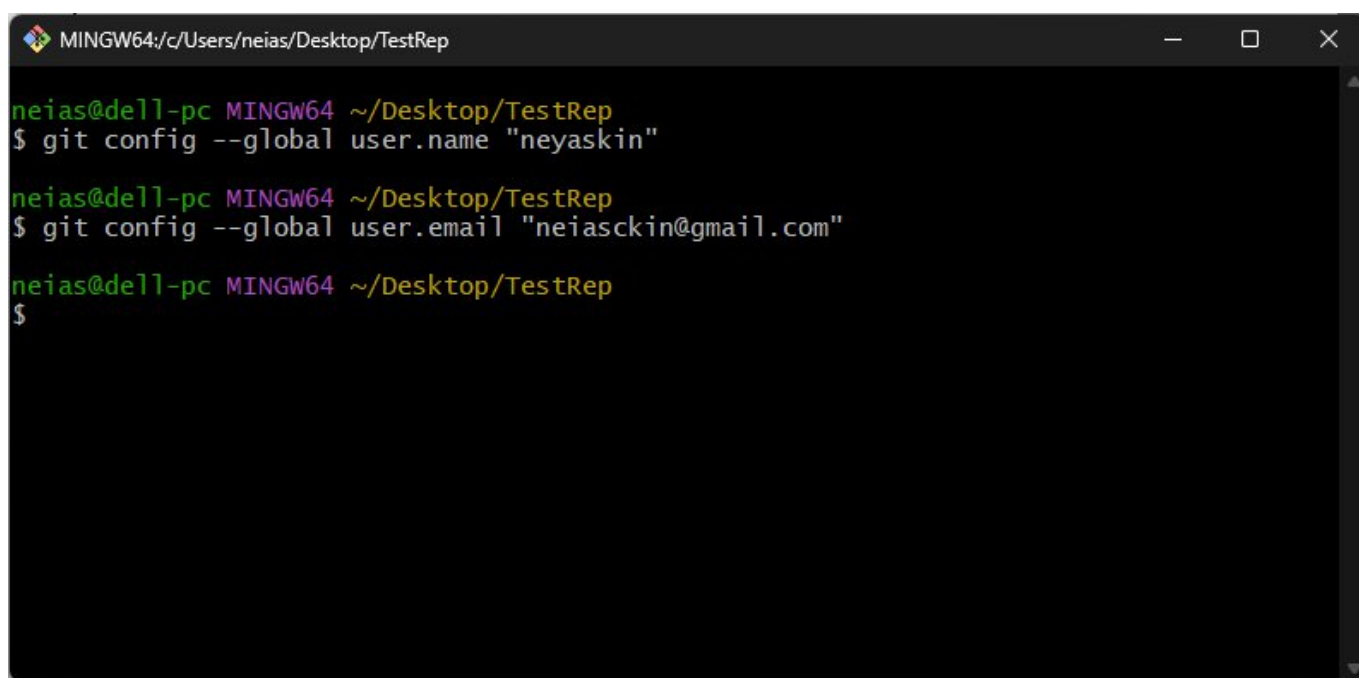


Откроется командная строка, в ней мы будем прописывать все нужные команды. После установки Git, вам нужно будет 1 раз указать ваш логин и почту GitHub. В командной строке пропишем 2 команды, логин и почту нужно подставить свои.



```
MINGW64: c:/Users/neias/Desktop/TestRep

neias@de11-pc MINGW64 ~/Desktop/TestRep
$ |
```



```
MINGW64: c:/Users/neias/Desktop/TestRep

neias@de11-pc MINGW64 ~/Desktop/TestRep
$ git config --global user.name "neyaskin"

neias@de11-pc MINGW64 ~/Desktop/TestRep
$ git config --global user.email "neiasckin@gmail.com"

neias@de11-pc MINGW64 ~/Desktop/TestRep
$
```


Начнем с команды «git init», она проинициализирует нам локальный репозиторий, в директории появится скрытый файл «.git».

Примечание

Прописывать эту команду нужно только 1 раз, во всей директории может быть только 1 «.git» файл.

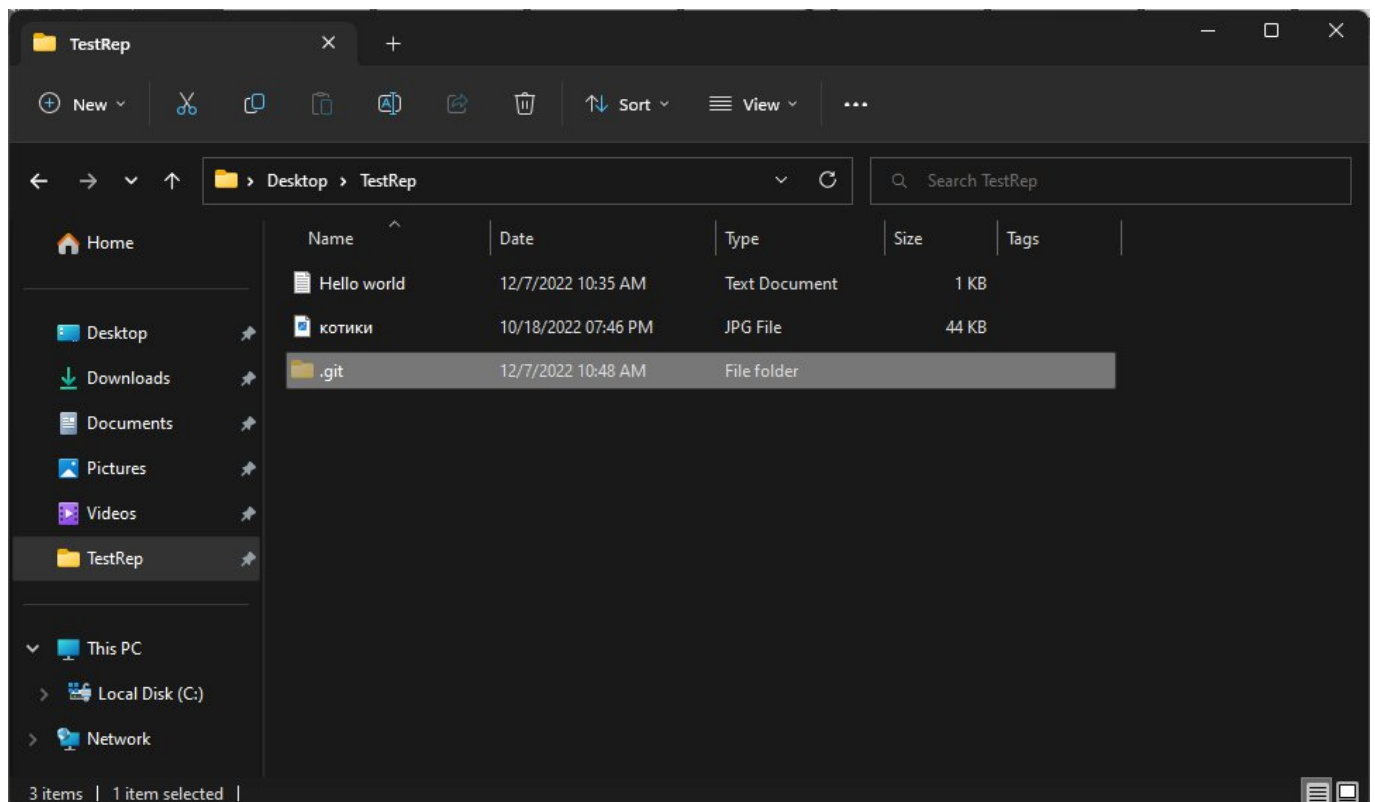
```
MINGW64:/c/Users/neias/Desktop/TestRep

neias@dell-pc MINGW64 ~/Desktop/TestRep
$ git config --global user.name "neyaskin"

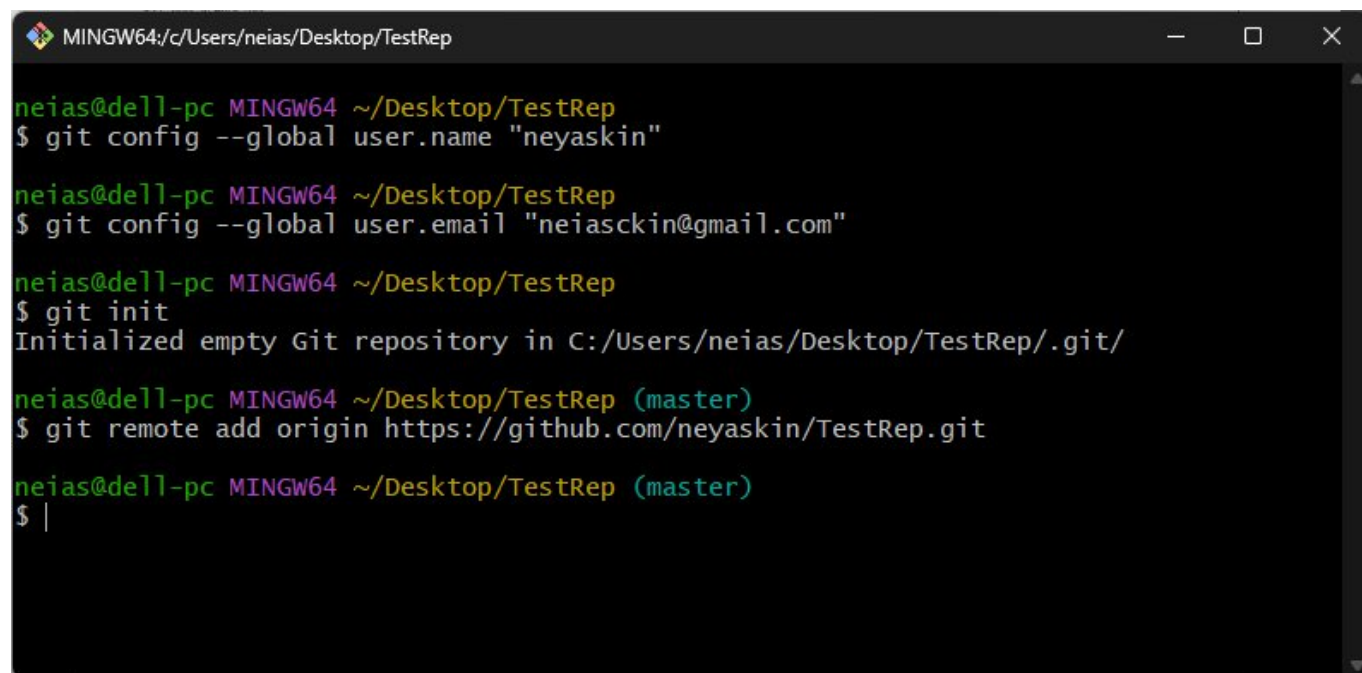
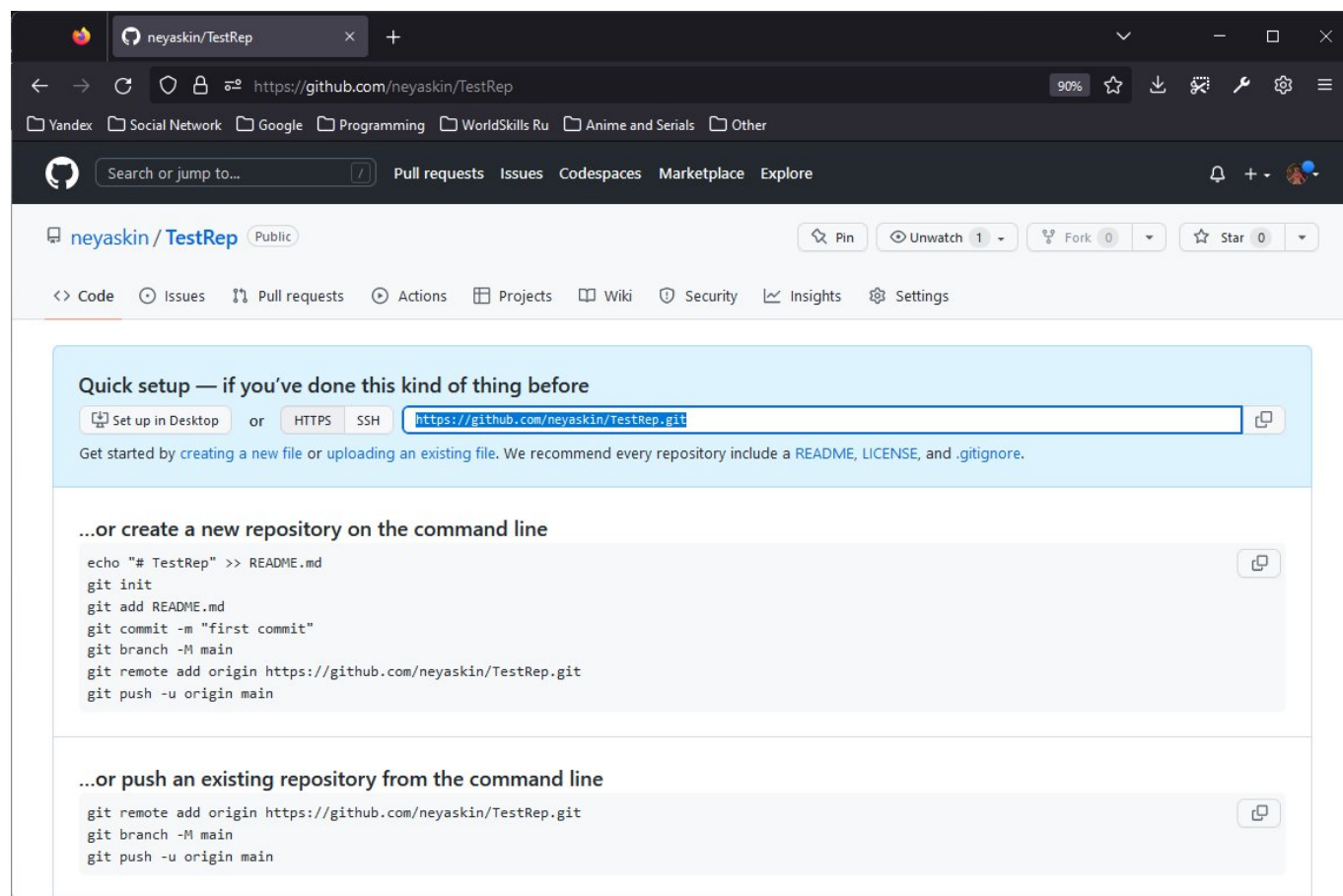
neias@dell-pc MINGW64 ~/Desktop/TestRep
$ git config --global user.email "neiasckin@gmail.com"

neias@dell-pc MINGW64 ~/Desktop/TestRep
$ git init
Initialized empty Git repository in C:/Users/neias/Desktop/TestRep/.git/

neias@dell-pc MINGW64 ~/Desktop/TestRep (master)
$ |
```



После инициализации лучше сразу подключиться к нашему удаленному репозиторию на GitHub. Нужно скопировать ссылку на него на странице репозитория. Команда «git remote add origin <ссылка>», подключит локальный репозиторий к удаленному.



Теперь нам нужно добавить все изменения, какие есть, в наш будущий локальный commit, воспользуемся командой «git add .». Точка говорит о том, что в commit добавятся все имеющиеся изменения в файлах.

```
MINGW64:/c/Users/neias/Desktop/TestRep
neias@dell-pc MINGW64 ~/Desktop/TestRep
$ git config --global user.name "neyaskin"

neias@dell-pc MINGW64 ~/Desktop/TestRep
$ git config --global user.email "neiasckin@gmail.com"

neias@dell-pc MINGW64 ~/Desktop/TestRep
$ git init
Initialized empty Git repository in C:/Users/neias/Desktop/TestRep/.git/

neias@dell-pc MINGW64 ~/Desktop/TestRep (master)
$ git remote add origin https://github.com/neyaskin/TestRep.git

neias@dell-pc MINGW64 ~/Desktop/TestRep (master)
$ git add .

neias@dell-pc MINGW64 ~/Desktop/TestRep (master)
$ |
```

Создадим первый локальный commit, напомним команду «git commit -m "first commit"». В двойных кавычках мы даем название нашему commit-у, как правило оно должно говорить о том, какие изменения были внесены, не делайте название слишком длинным, все коротко и по делу.

```
MINGW64:/c/Users/neias/Desktop/TestRep
$ git config --global user.email "neiasckin@gmail.com"

neias@dell-pc MINGW64 ~/Desktop/TestRep
$ git init
Initialized empty Git repository in C:/Users/neias/Desktop/TestRep/.git/

neias@dell-pc MINGW64 ~/Desktop/TestRep (master)
$ git remote add origin https://github.com/neyaskin/TestRep.git

neias@dell-pc MINGW64 ~/Desktop/TestRep (master)
$ git add .

neias@dell-pc MINGW64 ~/Desktop/TestRep (master)
$ git commit -m "first commit"
[master (root-commit) 545c293] first commit
 2 files changed, 1 insertion(+)
 create mode 100644 Hello world.txt
 create mode 100644 "\320\272\320\276\321\202\320\270\320\272\320\270.jpg"

neias@dell-pc MINGW64 ~/Desktop/TestRep (master)
$
```

Теперь нужно отправить(запустить) наш локальный commit с изменениями на удаленный GitHub репозиторий, введем команду «git push origin master». Откроется окно «Connect to GitHub», выбираем пункт «Token» и вставляем в поле токен, который мы создали на первых этапах, нажимаем кнопку «Sign in»

```
MINGW64:/c/Users/neias/Desktop/TestRep
neias@dell-pc MINGW64 ~/Desktop/TestRep
$ git init
Initialized empty Git repository in C:/Users/neias/Desktop/TestRep/.git/
neias@dell-pc MINGW64 ~/Desktop/TestRep (master)
$ git remote add origin https://github.com/neyaskin/TestRep.git
neias@dell-pc MINGW64 ~/Desktop/TestRep (master)
$ git add .
neias@dell-pc MINGW64 ~/Desktop/TestRep (master)
$ git commit -m "first commit"
[master (root-commit) 545c293] first commit
2 files changed, 1 insertion(+)
create mode 100644 Hello world.txt
create mode 100644 "\320\272\320\276\321\202\320\270\320\272\320\270.jpg"
neias@dell-pc MINGW64 ~/Desktop/TestRep (master)
$ git push origin master
```

Connect to GitHub

GitHub

Sign in

Browser/Device

Token

Sign in

Don't have an account?

Sign Up

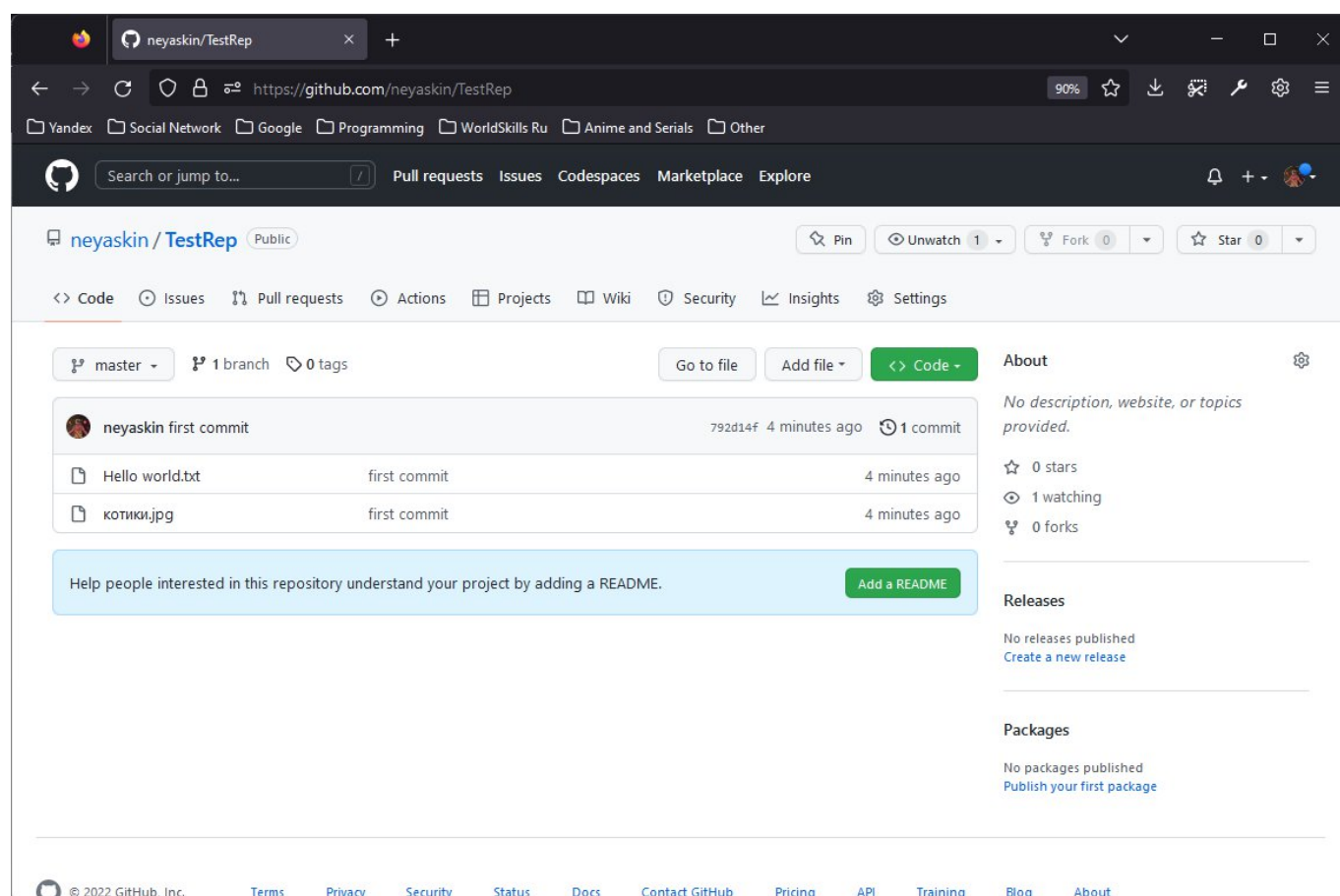
В консоли появится информация о том, что commit(изменения) успешно загружен на GitHub репозиторий. После обновления страницы репозитория, в нем появятся файлы из директории «TestRep».

```
MINGW64:/c/Users/neias/Desktop/TestRep

neias@dell-pc MINGW64 ~/Desktop/TestRep (master)
$ git commit -m "first commit"
[master (root-commit) 792d14f] first commit
2 files changed, 1 insertion(+)
create mode 100644 Hello world.txt
create mode 100644 "\320\272\320\276\321\202\320\270\320\272\320\270.jpg"

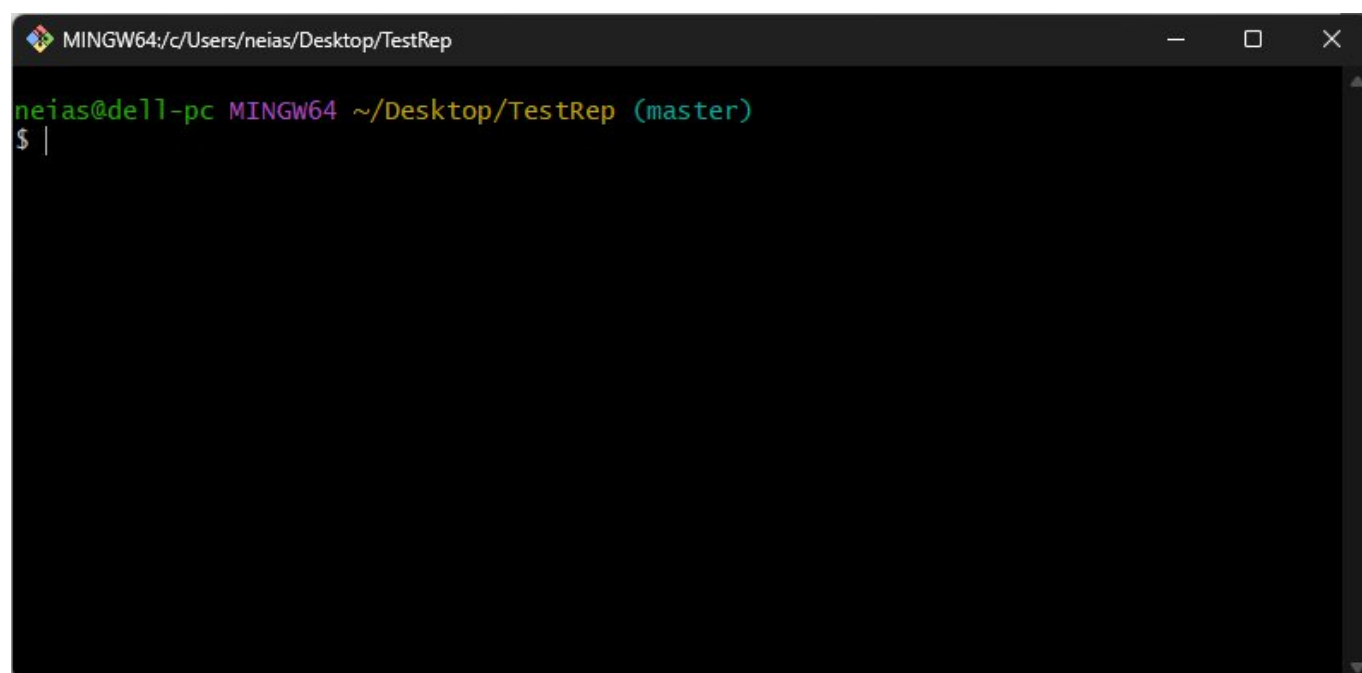
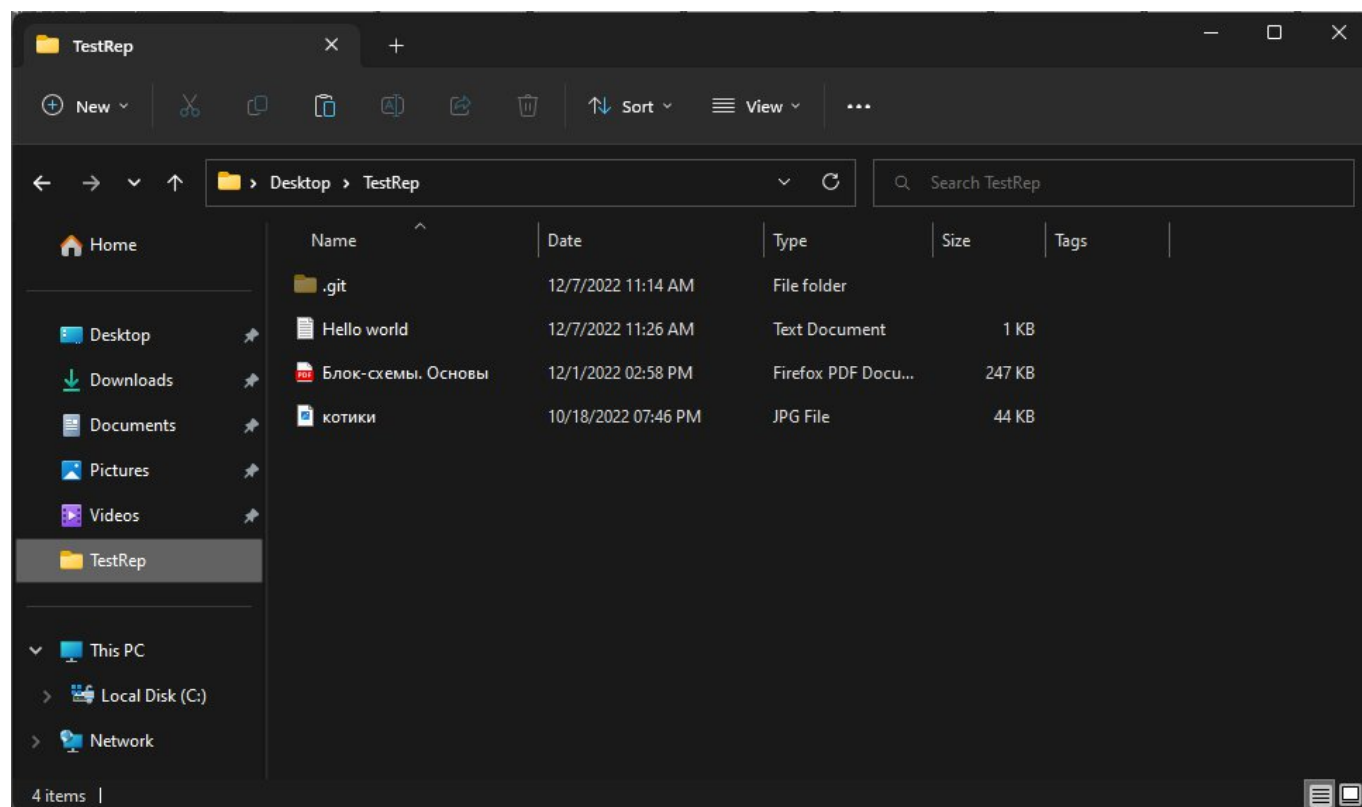
neias@dell-pc MINGW64 ~/Desktop/TestRep (master)
$ git push origin master
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 8 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (4/4), 33.04 KiB | 16.52 MiB/s, done.
Total 4 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/neyaskin/TestRep.git
* [new branch]      master -> master

neias@dell-pc MINGW64 ~/Desktop/TestRep (master)
$ |
```



Все последующие commit-ы

Первый commit загружен. Допустим, что в нашей директории произошли изменения, добавился новый файл и изменилось содержимое текстового документа, и теперь эти изменения нам нужно загрузить на GitHub. Открываем Git Bash в этой директории.



Посмотри какие изменения есть в директории, напомним команду «git status». Мы видим что текстовый файл изменен и добавлен новый pdf файл. Пишем «git add .», чтобы добавить все эти изменения в будущий commit.

```
MINGW64:/c/Users/neias/Desktop/TestRep

neias@dell-pc MINGW64 ~/Desktop/TestRep (master)
$ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   Hello world.txt

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        "\320\221\320\273\320\276\320\272-\321\201\321\205\320\265\320\274\321\213. \3
20\236\321\201\320\275\320\276\320\262\321\213.pdf"

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

neias@dell-pc MINGW64 ~/Desktop/TestRep (master)
$ |
```

```
MINGW64:/c/Users/neias/Desktop/TestRep

neias@dell-pc MINGW64 ~/Desktop/TestRep (master)
$ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   Hello world.txt

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        "\320\221\320\273\320\276\320\272-\321\201\321\205\320\265\320\274\321\213. \3
20\236\321\201\320\275\320\276\320\262\321\213.pdf"

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

neias@dell-pc MINGW64 ~/Desktop/TestRep (master)
$ git add .

neias@dell-pc MINGW64 ~/Desktop/TestRep (master)
$ |
```

Зафиксируем изменения создав commit, пишем команду «git commit -m "add pdf, change txt"». И отправим commit на GitHub репозиторий командой «git push origin master», в консоли увидим информацию о загрузке изменений на GitHub.

Примечание

Обратите внимание что вводить токен еще раз не пришлось.

```
MINGW64:/c/Users/neias/Desktop/TestRep
modified:  Hello world.txt

Untracked files:
  (use "git add <file>..." to include in what will be committed)
  "\320\221\320\273\320\276\320\272-\321\201\321\205\320\265\320\274\321\213. \320\236\321\201\320\275\320\276\320\262\321\213.pdf"

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

neias@dell-pc MINGW64 ~/Desktop/TestRep (master)
$ git add .

neias@dell-pc MINGW64 ~/Desktop/TestRep (master)
$ git commit -m "add pdf, change txt"
[master fbd759f] add pdf, change txt
 2 files changed, 1 insertion(+), 1 deletion(-)
 create mode 100644 "\320\221\320\273\320\276\320\272-\321\201\321\205\320\265\320\274\321\213. \320\236\321\201\320\275\320\276\320\262\321\213.pdf"

neias@dell-pc MINGW64 ~/Desktop/TestRep (master)
$ |
```

```
MINGW64:/c/Users/neias/Desktop/TestRep

neias@dell-pc MINGW64 ~/Desktop/TestRep (master)
$ git commit -m "add pdf, change txt"
[master fbd759f] add pdf, change txt
 2 files changed, 1 insertion(+), 1 deletion(-)
 create mode 100644 "\320\221\320\273\320\276\320\272-\321\201\321\205\320\265\320\274\321\213. \320\236\321\201\320\275\320\276\320\262\321\213.pdf"

neias@dell-pc MINGW64 ~/Desktop/TestRep (master)
$ git push origin master
Enumerating objects: 6, done.
Counting objects: 100% (6/6), done.
Delta compression using up to 8 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (4/4), 185.60 KiB | 20.62 MiB/s, done.
Total 4 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/neyaskin/TestRep.git
 792d14f..fbd759f  master -> master

neias@dell-pc MINGW64 ~/Desktop/TestRep (master)
$ |
```


Обновим страницу с репозиторием на GitHub. Увидим, что добавился файл и изменилось название commit-а у некоторых файлов.

The screenshot shows a web browser window displaying the GitHub repository page for 'neyaskin/TestRep'. The browser's address bar shows the URL 'https://github.com/neyaskin/TestRep'. The repository page includes a header with the repository name, a 'Public' badge, and buttons for 'Pin', 'Unwatch' (1), 'Fork' (0), and 'Star' (0). Below the header is a navigation bar with links for 'Code', 'Issues', 'Pull requests', 'Actions', 'Projects', 'Wiki', 'Security', 'Insights', and 'Settings'. The main content area shows the repository's commit history for the 'master' branch. The latest commit, 'neyaskin add pdf, change txt', is highlighted. It lists three files: 'Hello world.txt', 'Блок-схемы. Основы.pdf', and 'котики.jpg'. The commit message 'add pdf, change txt' is shown for the first two files, and 'first commit' for the third. The commit was made '1 minute ago' and has '2 commits'. To the right of the commit list, there is a section for 'About' with a description, '0 stars', '1 watching', and '0 forks'. Below this is a 'Releases' section with 'No releases published' and a link to 'Create a new release'. At the bottom, there is a 'Packages' section with 'No packages published' and a link to 'Publish your first package'. A blue banner at the bottom of the repository page encourages adding a README.

neyaskin/TestRep (Public)

Pin Unwatch 1 Fork 0 Star 0

<> Code Issues Pull requests Actions Projects Wiki Security Insights Settings

master 1 branch 0 tags

Go to file Add file <> Code

neyaskin add pdf, change txt fbd759f 1 minute ago 2 commits

Hello world.txt	add pdf, change txt	1 minute ago
Блок-схемы. Основы.pdf	add pdf, change txt	1 minute ago
котики.jpg	first commit	18 minutes ago

Help people interested in this repository understand your project by adding a README. Add a README

About

No description, website, or topics provided.

0 stars

1 watching

0 forks

Releases

No releases published

Create a new release

Packages

No packages published

Publish your first package

© 2022 GitHub, Inc. Terms Privacy Security Status Docs Contact GitHub Pricing API Training Blog About

Ссылки на дополнительную информацию

- [Документация по Git](#)
- [Документация по GitHub](#)

Ссылки на источники

- [Книга Git](#)
- [GitHub Wikipedia](#)